Inside a low budget consumer hardware espionage implant

Analysis of the S8 data line locator

mich @0x6d696368 (https://twitter.com/0x6d696368)

(Published: 2017-11-11, Last update: 2018-01-07)

The following analysis was performed on a S8 data line locator which replied to the hidden SMS command for version query (`*3646655*`) with:

```
Ver=MTK6261M.T16.17.01.10
build=2017/01/10 17:33
```

Introduction

A while back Joe Fitz tweeted about the *S8 data line locator*[1]. He referred to it as “Trickle down espionage” due to its reminiscence of NSA spying equipment.

The *S8 data line locator* is a GSM listening and location device hidden inside the plug of a standard USB data/charging cable. It supports the 850, 900, 1800 and 1900 MHz GSM frequencies.

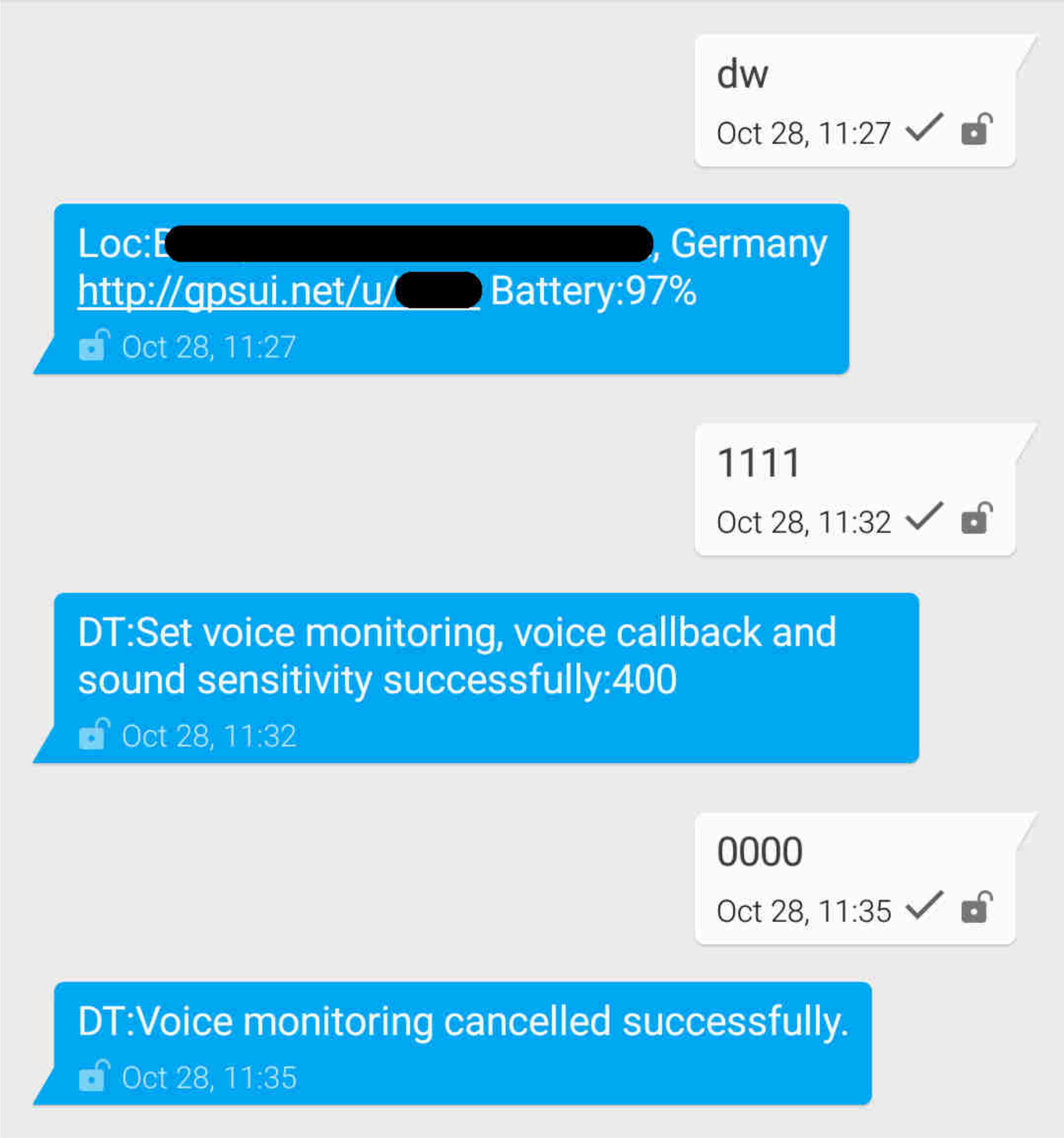
Its core idea is very similar to the COTTONMOUTH product line by the NSA/CSS [1] in which an RF device is hidden inside a USB plug. Those hidden devices are referred to as implants.

The device itself is marketed as a location tracker usable in cars, where a thief would not be able to identify the USB cable as a location tracking device. Its malicious use-cases can, however, not be denied. Especially since it features no GPS making its location reporting very coarse (1.57 km deviation in my tests). It can, e.g., be called to listen to a live audio feed from a small microphone within the device, as well as programmed to call back if the sound level surpasses a 45 dB threshold. The fact that the device can be repackaged in its sliding case, after configuring it, i.e. inserting a SIM, without any noticeable marks to the packaging suggests its use-case: covert espionage.

(images/00_packaging.jpg)

S8 data line locator capabilities

The S8 data line locator has several eavesdropping, espionage and spying capabilities. A SMS message log could look like this:



dw
Oct 28, 11:27 ✓

Loc:E[redacted], Germany http://gpsui.net/u/[redacted] Battery:97%
Oct 28, 11:27

1111
Oct 28, 11:32 ✓

DT:Set voice monitoring, voice callback and sound sensitivity successfully:400
Oct 28, 11:32

0000
Oct 28, 11:35 ✓

DT:Voice monitoring cancelled successfully.
Oct 28, 11:35

(images/10_00_cmds.jpg)

Listen in

Calling the S8 data line locator for 10 seconds establishes a call and allows you to listen to the microphone feed from the device.

Call back

Sending `1111` via SMS to the device enables voice activated call back. It is acknowledged via the following SMS reply:

```
DT: Set voice monitoring, voice callback and sound sensitivity:400
```

Once the audio level goes above 40 dB the device calls back the number that send the `1111` command.

Sending `0000` disables the audio triggered call back. It is replied by:

```
DT: Voice monitoring cancelled successfully.
```

Query location

Accoding to the manual sending ‘dw’ via SMS to the device yields a reply SMS with the location. This reply is in the form:

```
Loc:Street, ZIP City, Country
http://gpsui.net/u/xxxx Battery: 100%
```

The ‘xxxx’ are replaced with characters ‘0-9,A-Z,a-z’ and the `Street, ZIP City, Country` line with the appropriate street, ZIP, city and country. The link to http://gpsui.net can be accessed without authorization. It forwards to Google maps.

The location was never more accurate than 1.57 km off.

During the query the device will use a mobile data connection to an unknown endpoint (presumably gpsui.net). This is confirmed by a “MMS/Internet” charge by my provider. My provider does not discern MMS and Internet, but it is save to assume there is an Internet connection established during location query.

This issue was the stepping stone for this analysis. Because the device sends unknown data to an unknown third party it can not - at least with a clear conscious - be used in, e.g., a penetration test. You simply can not use a potentially pre-owned tool.

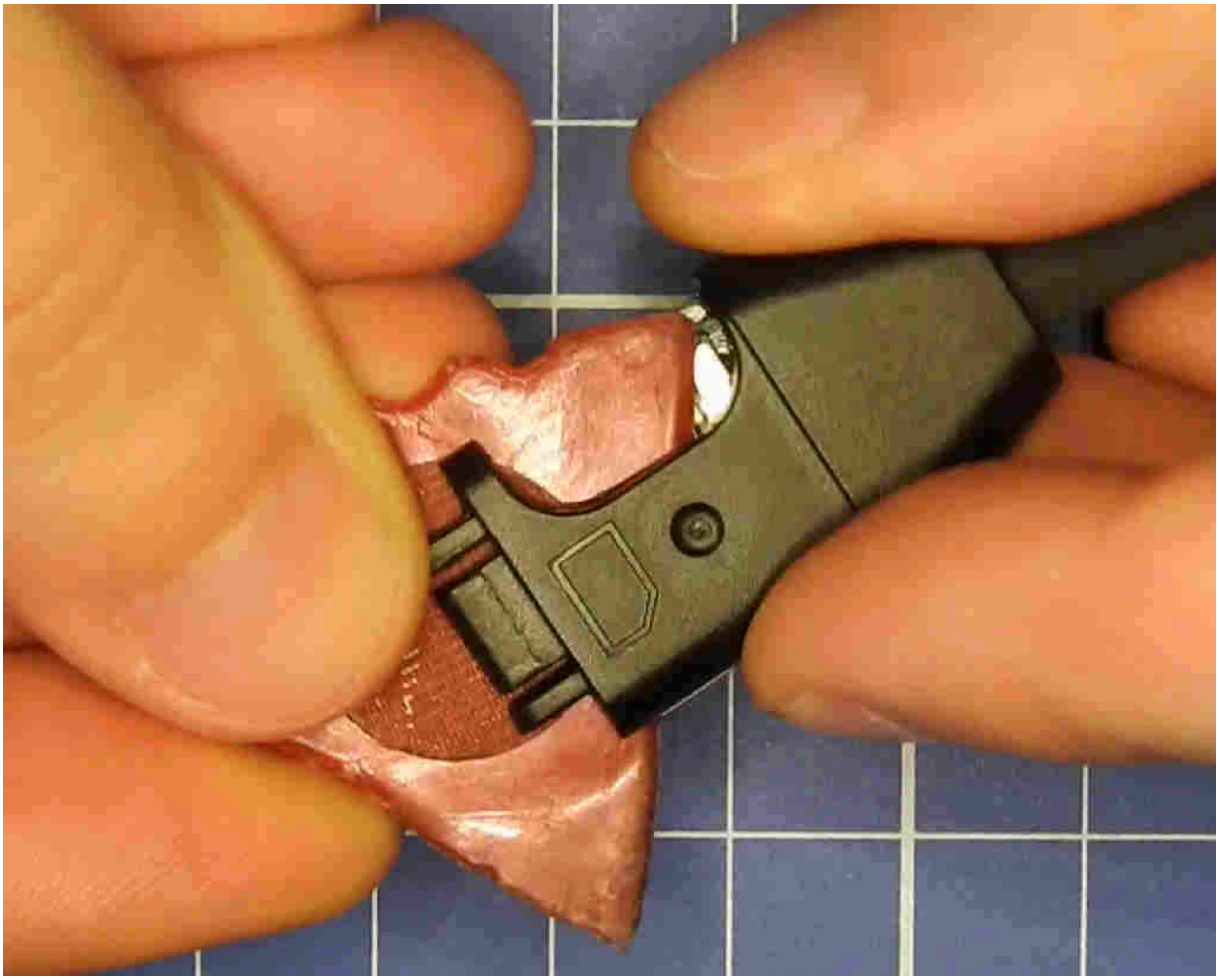
I therefore tried to analyze and eliminate this phone-home “feature”.

Hardware

To gain access to the devices innards we first tear of the metal shield of the USB connector:

(images/04_00_teardown.jpg)

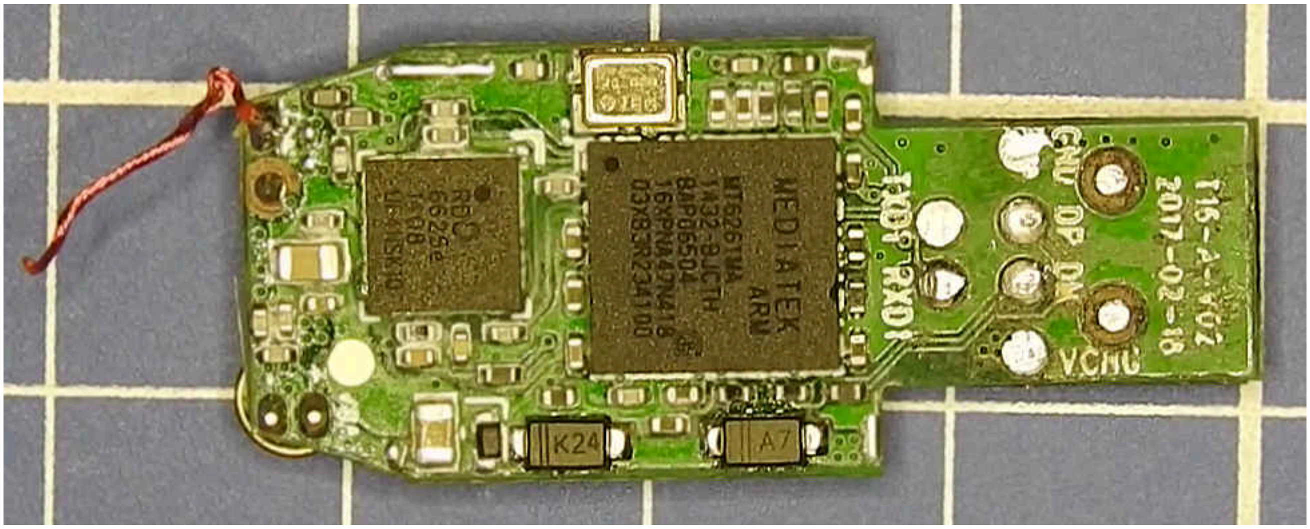
Next, we remove the plastic cover:



(images/04_03_teardown.jpg)

Chips

After opening the device we can identify the chips:



(images/06_02_chipid.jpg)

It features:

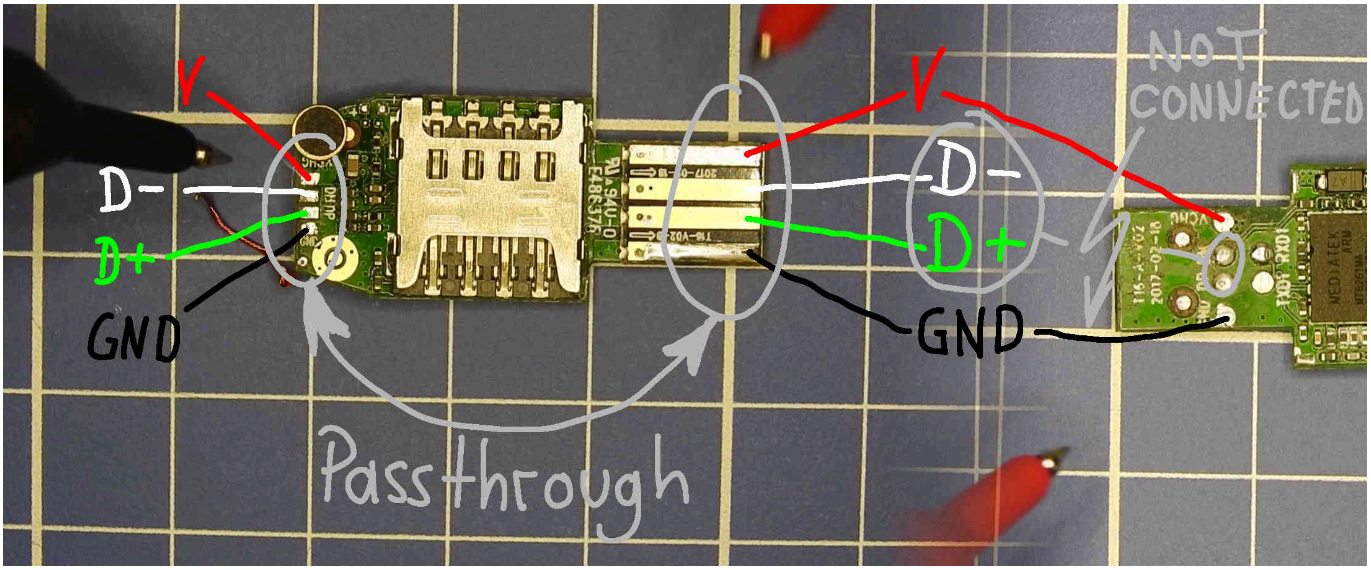
- MediaTek MT6261MA: Low budget chip often used in cheap Chinese smartwatches. No official documentation nor information about the chip is available from MediaTek.
- RDA 6626e: “a high-power, high-efficiency quad-band front-end Module […] designed for GSM850, EGSM900, DCS1800, PCS1900 handheld digital cellular equipment.”

Connections

So far I could identify 3 different avenues to connect to the device:

USB (passthrough)

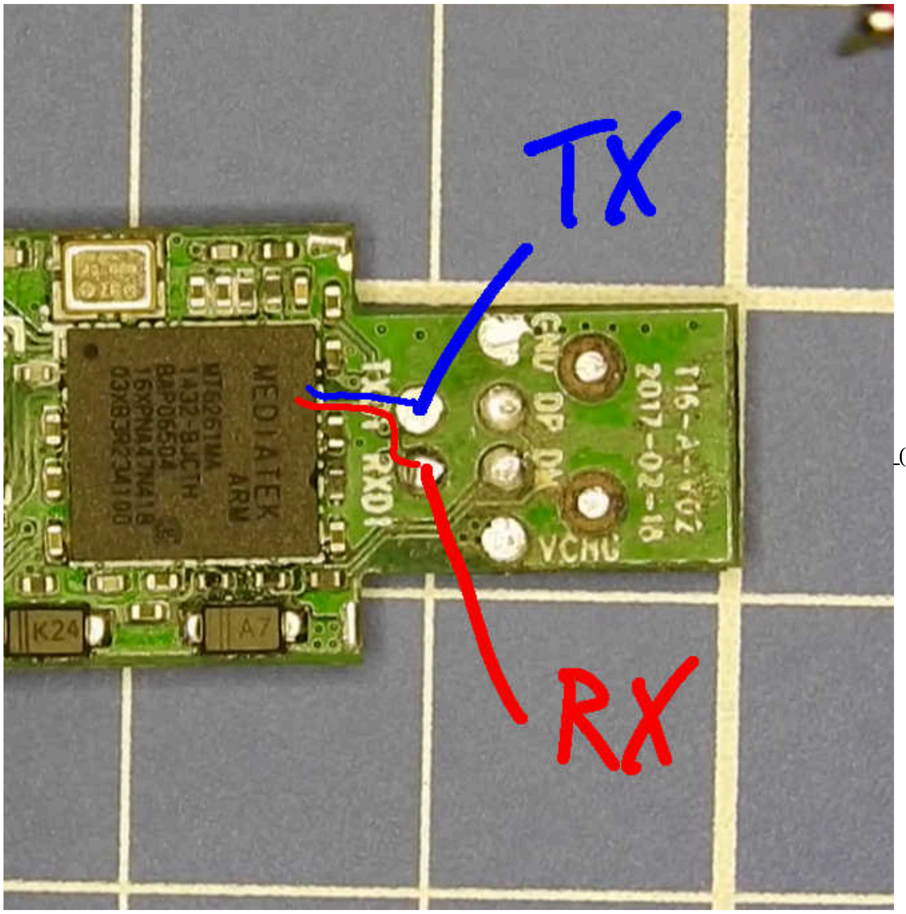
The USB A-connector and the Micro-B cable are not connected to the MT6261MA. They merely pass the signal from one to the other:



(images/07_00_meter.jpg)

UART

The next connection is a UART:



(images/07_02_meter.jpg)

Interfacing with it yields, approximately 3 seconds after booting the device:

```
screen /dev/ttyUSB0 115200 # 8N1

F1: 0000 0000
V0: 0000 0000 [0001]
00: 1029 0001
01: 0000 0000
U0: 0000 0001 [0000]
G0: 0002 0000 [0000]
T0: 0000 0C73
Jump to BL




~~~ Welcome to MTK Bootloader V005 (since 2005) ~~~
**===================================================**



Bye bye bootloader, jump to=0x1000a5b0
```

However, the output stops there. Input to the device is ignored.

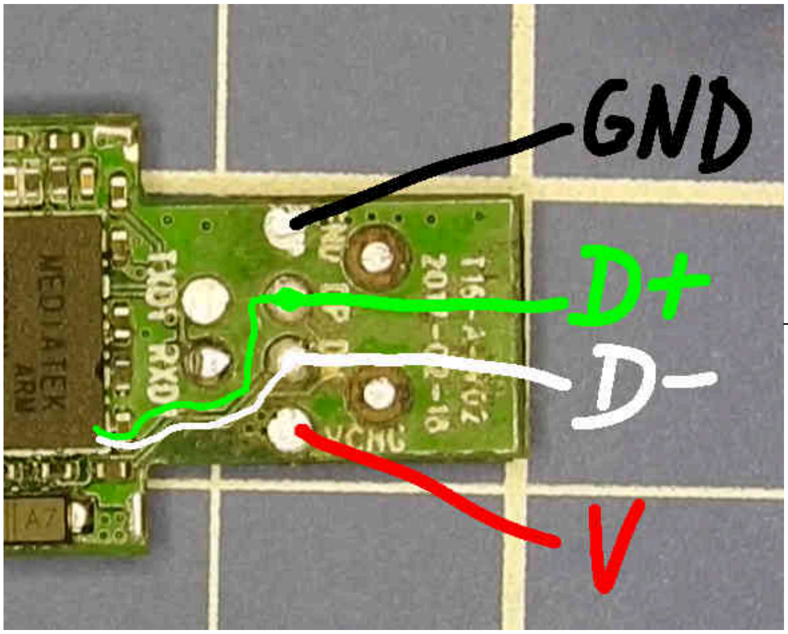
It is likely there exist a different firmware version that accepts AT modem commands[2]. The boot banner of that alternate firmware references “ZhiPu”[3] (some file names of the FAT12 in the firmware flash of my device contain this string as well, so the device firmware is likely related to that other firmware).

Alternative cable

```

F1: 0000 0000
V0: 0000 0000 [0001]
00: 1029 0002
01: 0000 0000
U0: 0000 0001 [0000]
G0: 0002 0000 [0000]
T0: 0000 0C73
Jump to BL





~~~ Welcome to MTK Bootloader V005 (since 2005) ~~~
**===================================================**



Bye bye bootloader, jump to=0x1000a5b0

LOG: RegisterSn:
LOG: ZhiPu_sock_buf_init malloc= 217780, 217180, 216940
LOG: ZhiPu_mmi_get_imsi_request
LOG: ZhiPu_system_init VERSION= MTK6261M.T16.17.01.10 , build date is 2017/01/10 17:33, curtime 2004-01-01 00:00
LOG: g_zhipu_imei= \
LOG: ----- 0 -----  ----- -268081676 -----  ----- 2 -----
LOG: ZhiPu_sms_ready_sync
LOG: ZhiPu System Language: English
LOG: service_availability= 0,ChargerConnected= 1,poweron_mode= 0
LOG: sim invalid, 4 minutes later reboot
LOG: ----- 0 -----  ----- 83 -----  ----- 2 -----
LOG: idle_screen_network_name:Same IMEI
```

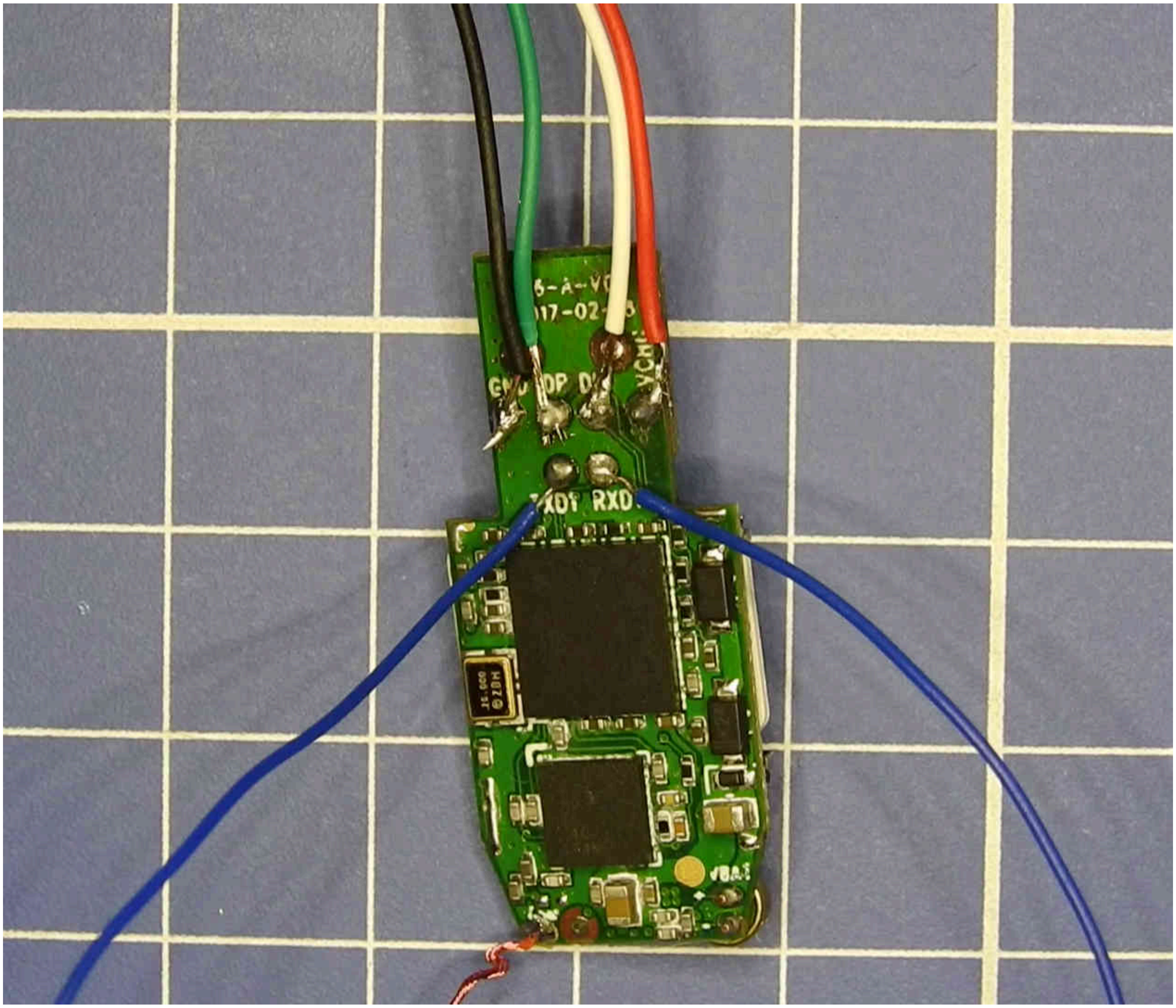
USB (MTK)

The DP and DM pads on the USB connector are not connected to the D+ and D- lines of the USB connector. However, the V and GND pads are. The DP and DM pads are instead routed to the MT6261MA processor as illustrated here:



(images/07_01_meter.jpg)

Next, a USB cable must be soldered to these connectors as follows:



(images/09_usb.jpg)

The device will then be recognized as an MediaTek phone USB endpoint with the following data:

```
ID 0e8d:0003 MediaTek Inc. MT6227 phone
```

This is often called the “MTK boot repair”, “MTK DM DP flash”, etc. It will allow us to interface with the device and dump the firmware ROM and flash.

Dumping firmware

To dump the firmware I use the open source Fernvale research OS [2]. It was initially targeted for the MT6260 processor. It has, however, been ported to the MT6261 and also works on the MT6261MA.

Obtaining and building fernly’s MT6261 branch

A suitable fork of fernly by Urja “urjaman” Rannikko can be obtained and build as follows:

```
git clone https://github.com/urjaman/fernly
git clone https://github.com/robertfoss/setup_codesourcery.git
sudo setup_codesourcery/setup.sh
/usr/local/bin/codesourcery-arm-2014.05.sh
cd fernly
git checkout fernly6261
make CROSS_COMPILE=arm-none-eabi-
exit
cp 95-fernvale-simple.rules /etc/udev/rules.d/.
```

Dumping ROM

To dump the flash we run:

```
echo "data = [" > rom.py
fernly/build/fernly-usb-loader /dev/fernvale fernly/build/dump-rom-usb.bin >> rom.py
echo "
]
f = open('rom.bin','wb')
for s in data:
  f.write(chr(int(s,16)))
f.close()
" >> rom.py
python rom.py
```

The file `rom.bin` will now (at least according to the fernly repository documentation) contain the devices ROM.

Dumping flash

To dump the flash of the device we need to patch flashrom as follows:

```
git clone https://github.com/flashrom/flashrom
cd flashrom/
git checkout c8305e1dee66cd69bd8fca38bff2c8bf32924306
patch -p0 < ../fernly/flashrom-fernvale.patch
# manually fix Makefile.rej to complete patching
```

The patch does not cleanly apply so you need to fix the rejected `Makefile` (`Makefile.rej`) manually yourself.

Once this was done we can first load the fernly firmware into the devices RAM via:

```
fernly/build/fernly-usb-loader -w /dev/fernvale fernly/build/stage1.bin fernly/build/firmware.bin
```

Next, we can use the `fernvale_spi` programmer we patched into flashrom.

We first let it recognize the flash via:

```
flashrom/flashrom --programmer fernvale_spi:dev=/dev/fernvale
```

And then read the flash via:

```
flashrom/flashrom --programmer fernvale_spi:dev=/dev/fernvale -c "MX25L3205(A)" --read flash.dat
```

`flash.dat` will now contain the devices flash memory.

Writing flash attempt

Writing the flash can be performed via:

```
flashrom/flashrom --programmer fernvale_spi:dev=/dev/fernvale -c "MX25L3205(A)" --write flash.dat
```

However, the flash seems to be block protected and the block protect bits can not be disabled by flashrom. I have not (yet) found a way to disable the block protect.

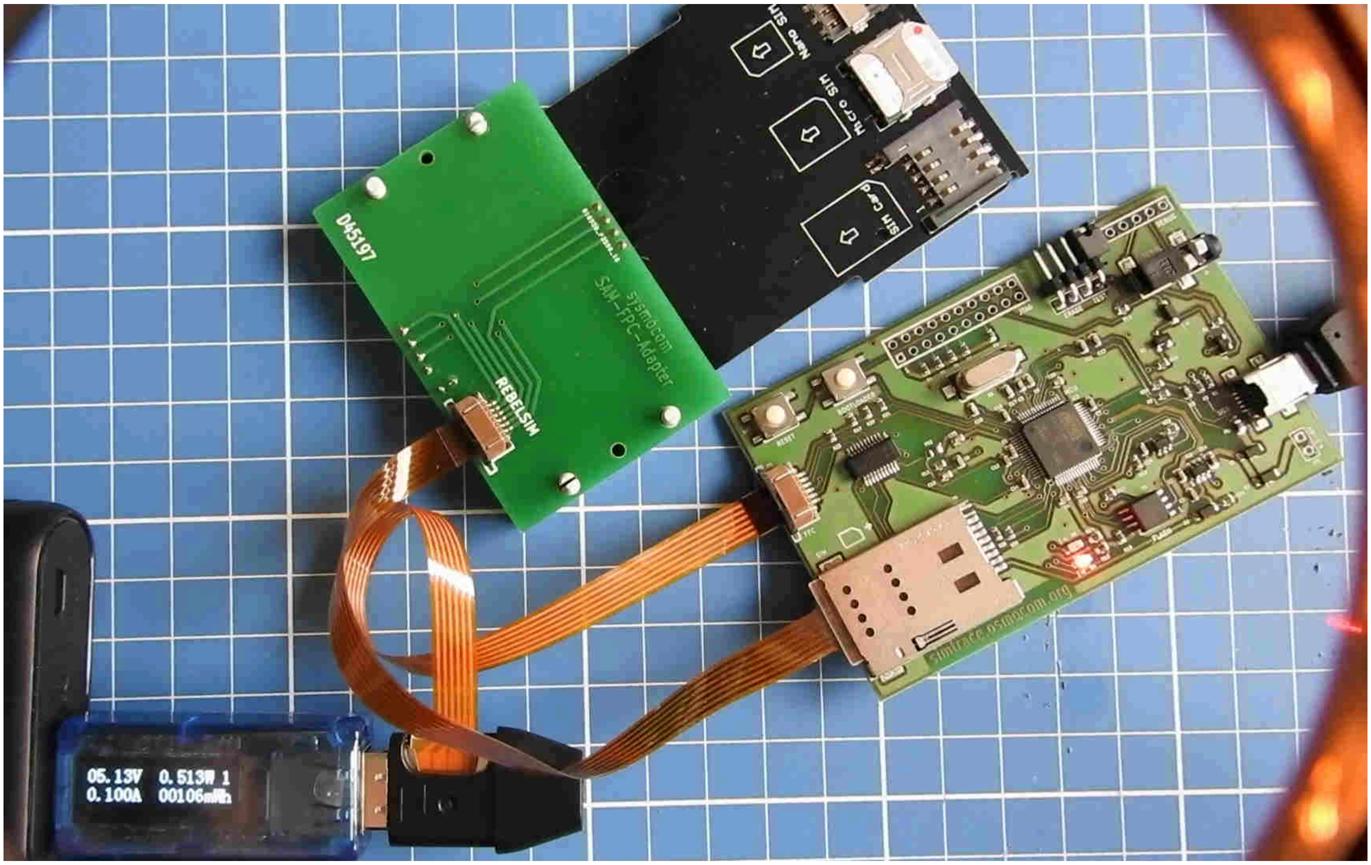
Alternative device

```
$ flashrom/flashrom --programmer fernvale_spi:dev=/dev/fernvale
flashrom v0.9.9-86-ge1a960e-dirty on Linux 4.13.2-1.el7.elrepo.x86_64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Found GigaDevice flash chip "GD25LQ32" (4096 kB, SPI) on fernvale_spi.
No operations were specified.
```

Analysis

Mostly for my personal education I did some more analysis then the obligatory firmware dump.

SIM sniffing (via SIMtrace)



(images/02_simtrace.jpg)

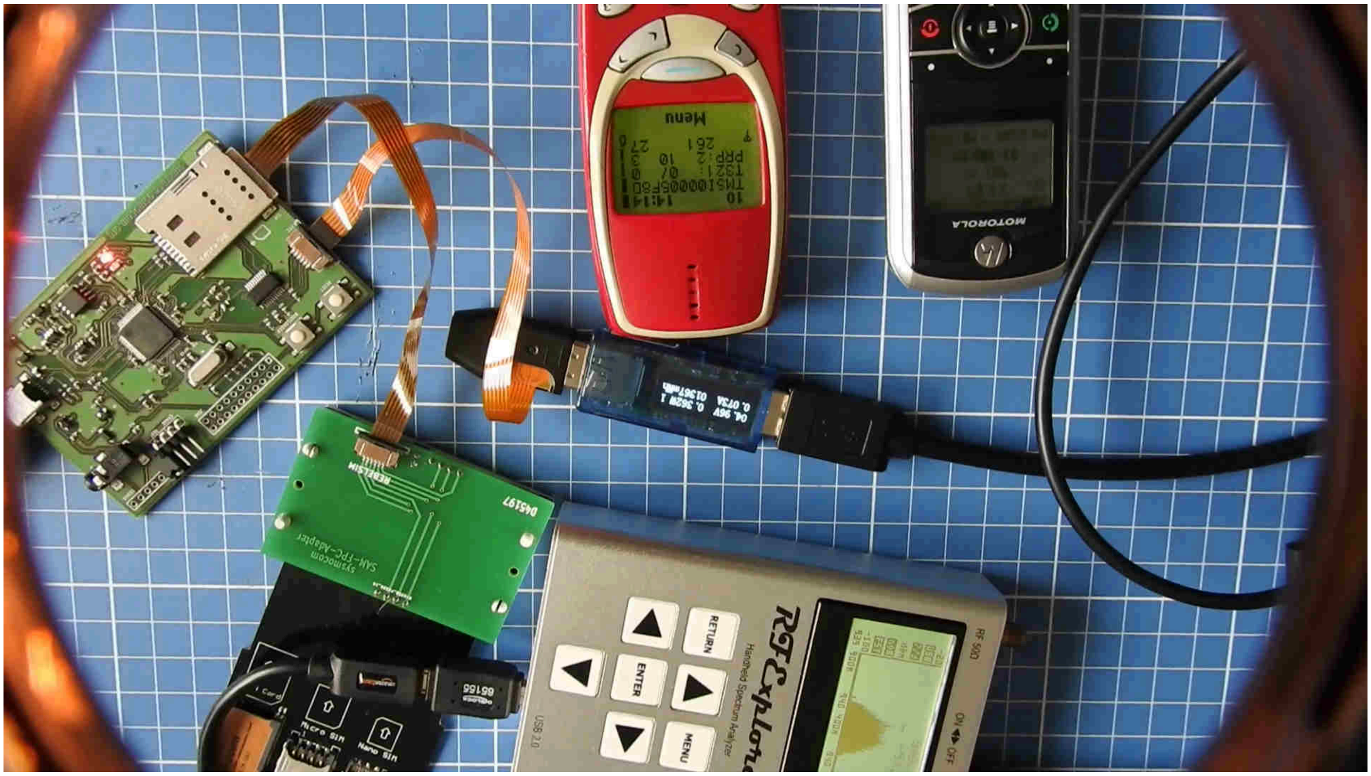
First, I sniffed the communication between the device and the SIM. Interestingly, it accessed all records of the telephone book and SMS storage. More specifically it accesses the following files, which are not needed to provide the services rendered by the device itself:

- ADF
  - EF(ECC)
  - EF(EXT2)
  - EF(SMS)
- DF(TELECOM)
  - DF(PHONEBOOK)
    - EF(ADN)
    - EF(ANRA1)
    - EF(SMS)

Other SIM accesses seems to be normal.

This is probably not an elaborate scheme to harvest phone numbers and send them to China, but rather the way the default manufactured SIM code was implemented and it was never trimmed down to the needs of this device. Nevertheless, I found it interesting seeing how the device is accessing virtually everything on the SIM.

GPRS sniffing attempt (via OpenBTS)



(images/03_000_openbts.jpg)

Next, I tried to sniff the Internet traffic to figure out what is send to whom via the mobile data connection. To this end, I used a Ettus B100 with OpenBTS.

Unfortunately, the S8 data line locator did not connect to the GPRS. This caused the following alternative response to the `dw` location command:

```
Loc:Please link:http://gpsui.net/smap.php?lac=1000&cellid=10&c=901&n=70&v=7100 Battery:67%
```

Flash contents

The most interesting things could be found in the dumped flash.

OS

Strings in the `flash.dat` suggest the device is probably running Nucleus RTOS:

```
$ strings -a flash.dat
Copyright (c) 1993-2000 ATI - Nucleus PLUS - Version ARM 7/9  1.11.19
```

Other strings that may help identify the OS are:

```
$ strings -a flash.dat | grep "\.c"
psss\components\src\bl_Secure_v5.c
psss\components\src\SSS_secure_shared_v5.c
hal\system\bootloader\src\bl_Main.c
hal\system\bootloader\src\bl_Main.c
hal\system\bootloader\src\bl_FTL.c
hal\system\bootloader\src\bl_FTL.c
hal\system\bootloader\src\bl_FTL.c
hal\storage\flash\mtd\src\flash_disk.c
hal\system\bootloader\src\bl_Main.c
hal\peripheral\src\dcl_pmu6261.c
hal\system\cache\src\cache.c
hal\peripheral\src\dcl_rtc.c
hal\peripheral\src\dcl_pmu6261.c
hal\system\bootloader\src\bl_FTL.c
hal\system\bootloader\src\bl_FTL.c
hal\peripheral\src\rtc.c
hal\peripheral\src\rtc.c
hal\peripheral\src\rtc.c
hal\peripheral\src\rtc.c
hal\peripheral\src\rtc.c
hal\peripheral\src\rtc.c
hal\storage\flash\mtd\src\flash_mtd_sf_dal.c
hal\peripheral\src\dcl_pmu_common.c
hal\peripheral\src\dcl_f32k_clk.c
hal\peripheral\src\dcl_f32k_clk.c
hal\peripheral\src\dcl_gpio.c
hal\peripheral\src\dcl_pmu_common.c
hal\system\cache\src\cache.c
hal\peripheral\src\dcl_f32k_clk.c
hal\peripheral\src\dcl_gpio.c
hal\peripheral\src\gpio.c
hal\system\bootloader\src\bl_FTL.c
hal\peripheral\src\rtc.c
hal\peripheral\src\bmt_hw.c
hal\peripheral\src\dcl_pmu6261.c
hal\storage\flash\mtd\src\flash_mtd.c
hal\peripheral\src\gpio.c
custom\common\hal\combo_flash_nor.c
hal\peripheral\src\dcl_rtc.c
hal\peripheral\src\dcl_rtc.c
hal\storage\flash\mtd\src\flash_disk.c
custom\common\hal\combo_flash_nor.c
hal\storage\flash\mtd\src\flash_mtd_sf_dal.c
hal\system\emi\src\emi.c
sss\components\src\SSS_secure_shared_common.c
alice.c
ddload.c
plutommi\Framework\GDI\gdisrc\gdi.c
C.cKi
hal\audio\src\v1\audio_service.c
ddload.c
ddload.c
plutommi\Framework\GDI\gdisrc\gdi_image_hwjpg_v2.c
```

```
plutommi\Framework\GDI\gdisrc\gdi_image_hwjpg_v2.c
plutommi\Framework\GDI\gdisrc\gdi_util.c
plutommi\Framework\GDI\gdisrc\gdi_util.c
hal\audio\src\v1\audio_service.c
ddload.c
```

FAT12 filesystems (?)

Update: In my original write up I missed the flash translation layer, hence the FAT12 filesystem was corrupted. Thanks to Bjoern Kerler (@viperbjk (https://twitter.com/viperbjk)) for pointing this out to me. I will work on descrambling the flash and update once I have it working.

For the time being here is my original try at extracting the files from the file system:

Searching the `flash.dat` for the FAT12 file systems that are supposedly present in on MediaTek phones, we get:

```
$ hexdump -C flash.dat
002c1e20  00 00 00 00 00 00 00 00  00 ef cd 4e 4f 20 4e 41  |...........NO NA|
002c1e30  4d 45 20 20 20 20 46 41  54 31 32 20 20 20 00 00  |ME    FAT12   ..|
002c1e40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
002c1e50  00 00 00 00 00 00 00 00  00 00 4d 4d 4d 4d 4d 4d  |..........MMMMMM|
002c1e60  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d  |MMMMMMMMMMMMMMMM|
*
002c1ff0  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 55 aa  |MMMMMMMMMMMMMMU.|
[...]
002d8400  eb 58 90 46 69 6c 65 53  79 73 20 00 02 01 01 00  |.X.FileSys .....|
002d8410  01 80 00 9b 01 f8 02 00  01 00 01 00 01 00 00 00  |................|
002d8420  9b 01 00 00 80 00 29 00  00 21 30 4e 4f 20 4e 41  |......)..!0NO NA|
002d8430  4d 45 20 20 20 20 46 41  54 31 32 20 20 20 00 00  |ME    FAT12   ..|
002d8440  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
002d8450  00 00 00 00 00 00 00 00  00 00 4d 4d 4d 4d 4d 4d  |..........MMMMMM|
002d8460  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d  |MMMMMMMMMMMMMMMM|
*
002d85f0  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 55 aa  |MMMMMMMMMMMMMMU.|
[...]
002dbc00  ff ff ff ff ff de 9e 68  00 00 00 00 00 50 ba ff  |.......h.....P..|
002dbc10  55 93 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |U...............|
002dbc20  00 00 00 00 00 00 00 00  00 ef cd 4e 4f 20 4e 41  |...........NO NA|
002dbc30  4d 45 20 20 20 20 46 41  54 31 32 20 20 20 00 00  |ME    FAT12   ..|
002dbc40  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
002dbc50  00 00 00 00 00 00 00 00  00 00 4d 4d 4d 4d 4d 4d  |..........MMMMMM|
002dbc60  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 4d 4d  |MMMMMMMMMMMMMMMM|
*
002dbdf0  4d 4d 4d 4d 4d 4d 4d 4d  4d 4d 4d 4d 4d 4d 55 aa  |MMMMMMMMMMMMMMU.|
```

However, two of the partitions do not appear to be valid FAT12 file systems:

```
$ fls -o 5646 flash.dat -f fat12
Invalid magic value (Not a FATFS file system (magic))
$ fls -o 5826 flash.dat -f fat12
v/v 6531:   $MBR
v/v 6532:   $FAT1
v/v 6533:   $FAT2
d/d 6534:   $OrphanFiles
$ fls -o 5853 flash.dat -f fat12
Invalid magic value (Not a FATFS file system (magic))
```

And the middle FAT12 block seems to be corrupted as well, i.e. only orphan files:

```
$ fls -o 5826 flash.dat -rp -f fat12
v/v 6531:   $MBR
v/v 6532:   $FAT1
v/v 6533:   $FAT2
d/d 6534:   $OrphanFiles
-/r * 469:  $OrphanFiles/MP0B_001
-/r * 470:  $OrphanFiles/ST33A004
-/r * 471:  $OrphanFiles/ST33B004
[...]
```

An attempt was made to extract the files:

```
fls -o 5826 flash.dat -Frp -f fat12 | while read line; do
    path=$(echo "$line" | awk -F':' '{print $2}')
    mkdir -p $(dirname $path);
    icat -o 5826 flash.dat $(echo "$line" | grep -oE "[0-9]+" | head -n1) > $path
done
```

But most files are empty. The results are also very inconsistent, i.e., when changing SIM cards there are significant changes to the files listed by The Sleuthkit. This indicates that those are either not FAT12 partitions or a modified FAT12 variant.

Again as stated at the start of this section I’m missing the flash translation layer (FTL). There are proprietary tools which can do the FTL descrambling for you.

But to keep this whole writeup open source, further analysis was done using hexdump.

Configuration data

The flash also contained some configuration data. First, the IMSI of the inserted SIM and the number that is used to remote control the device could be found in the flash:

```
$ hexdump -C flash.dat
002e2ad0  00 00 00 00 32 xx xx xx  xx xx xx xx xx xx xx xx  |....2xxxxxxxxxxx|
002e2ae0  xx xx 37 00 00 01 00 01  00 00 00 00 00 00 00 00  |xx7.............|
002e2af0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
002e2b10  00 00 00 00 00 00 00 00  00 00 00 2b 34 39 31 xx  |...........+491x|
002e2b20  xx xx xx xx xx xx xx xx  00 00 00 00 00 00 00 00  |xxxxxxxx........|
002e2b30  00 00 00 00 00 00 00 00  00 00 00 00 00 67 70 73  |.............gps|
002e2b40  75 69 2e 6e 65 74 00 00  00 00 00 00 00 00 00 00  |ui.net..........|
002e2b50  00 00 00 00 00 00 00 00  00 00 00 00 00 67 70 73  |.............gps|
002e2b60  75 69 2e 6e 65 74 00 00  00 00 00 00 00 00 00 00  |ui.net..........|
002e2b70  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

In the above flash segment you can also see a reference to `gpsui.net`. This is presumably the remote server which is contacted to turn the MCC, MNC, LAI and CID codes into street, city and country information as well as the link to gpsui.net which forwards to Google maps. However, because writing to the flash could not be achieved this hypothesis could not be confirmed.

Hidden commands

Eventually, there was a small find potentially making this effort worthwhile. Searching the `flash.dat` for the `dw`,`1111` and `0000` commands reveals more hidden commands:

```
$ hexdump -C flash.dat
00069530  8c ae 00 00 8d ae 8e ae  72 65 73 74 6f 72 65 00  |........restore.|
00069540  00 00 00 00 00 00 00 00  01 00 00 00 68 68 68 00  |............hhh.|
00069550  00 00 00 00 00 00 00 00  00 00 00 00 02 00 00 00  |................|
00069560  69 6d 73 69 00 00 00 00  00 00 00 00 00 00 00 00  |imsi............|
00069570  03 00 00 00 74 69 6d 65  7a 6f 6e 65 00 00 00 00  |....timezone....|
00069580  00 00 00 00 04 00 00 00  74 69 6d 65 00 00 00 00  |........time....|
00069590  00 00 00 00 00 00 00 00  05 00 00 00 61 71 65 00  |............aqe.|
000695a0  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
000695b0  61 71 63 00 00 00 00 00  00 00 00 00 00 00 00 00  |aqc.............|
000695c0  07 00 00 00 73 65 72 76  65 72 00 00 00 00 00 00  |....server......|
000695d0  00 00 00 00 08 00 00 00  64 64 64 00 00 00 00 00  |........ddd.....|
000695e0  00 00 00 00 00 00 00 00  09 00 00 00 72 65 67 00  |............reg.|
000695f0  00 00 00 00 00 00 00 00  00 00 00 00 0a 00 00 00  |................|
00069600  61 71 62 00 00 00 00 00  00 00 00 00 00 00 00 00  |aqb.............|
00069610  0b 00 00 00 71 71 71 00  00 00 00 00 00 00 00 00  |....qqq.........|
00069620  00 00 00 00 0c 00 00 00  64 77 00 00 00 00 00 00  |........dw......|
00069630  00 00 00 00 00 00 00 00  0d 00 00 00 6c 6f 63 00  |............loc.|
00069640  00 00 00 00 00 00 00 00  00 00 00 00 0e 00 00 00  |................|
00069650  66 61 61 00 00 00 00 00  00 00 00 00 00 00 00 00  |faa.............|
00069660  0f 00 00 00 66 66 66 00  00 00 00 00 00 00 00 00  |....fff.........|
00069670  00 00 00 00 10 00 00 00  31 31 31 31 00 00 00 00  |........1111....|
00069680  00 00 00 00 00 00 00 00  11 00 00 00 30 30 30 30  |............0000|
00069690  00 00 00 00 00 00 00 00  00 00 00 00 12 00 00 00  |................|
000696a0  72 70 74 00 00 00 00 00  00 00 00 00 00 00 00 00  |rpt.............|
000696b0  13 00 00 00 67 62 72 70  74 00 00 00 00 00 00 00  |....gbrpt.......|
000696c0  00 00 00 00 14 00 00 00  74 72 61 63 6b 00 00 00  |........track...|
000696d0  00 00 00 00 00 00 00 00  15 00 00 00 6d 6f 6e 69  |............moni|
000696e0  74 6f 72 00 00 00 00 00  00 00 00 00 16 00 00 00  |tor.............|
000696f0  73 6f 73 6f 6e 00 00 00  00 00 00 00 00 00 00 00  |soson...........|
00069700  17 00 00 00 73 6f 73 6f  66 66 00 00 00 00 00 00  |....sosoff......|
00069710  00 00 00 00 18 00 00 00  73 6f 73 00 00 00 00 00  |........sos.....|
00069720  00 00 00 00 00 00 00 00  19 00 00 00 71 63 73 6f  |............qcso|
00069730  73 00 00 00 00 00 00 00  00 00 00 00 1a 00 00 00  |s...............|
00069740  6c 65 64 6f 6e 00 00 00  00 00 00 00 00 00 00 00  |ledon...........|
00069750  1b 00 00 00 6c 65 64 6f  66 66 00 00 00 00 00 00  |....ledoff......|
00069760  00 00 00 00 1c 00 00 00  66 6c 69 67 68 74 6f 6e  |........flighton|
00069770  00 00 00 00 00 00 00 00  1d 00 00 00 66 6c 69 67  |............flig|
00069780  68 74 6f 66 66 00 00 00  00 00 00 00 1e 00 00 00  |htoff...........|
00069790  65 73 69 6f 6e 6f 77 00  00 00 00 00 00 00 00 00  |esionow.........|
000697a0  1f 00 00 00 65 73 69 6f  61 64 64 72 00 00 00 00  |....esioaddr....|
000697b0  00 00 00 00 20 00 00 00  68 62 74 6f 6e 00 00 00  |.... ...hbton...|
000697c0  00 00 00 00 00 00 00 00  21 00 00 00 68 62 74 6f  |........!...hbto|
000697d0  66 66 00 00 00 00 00 00  00 00 00 00 22 00 00 00  |ff.........."...|
000697e0  65 73 69 6f 6c 6f 63 61  74 65 74 79 70 65 00 00  |esiolocatetype..|
000697f0  23 00 00 00 65 65 65 00  00 00 00 00 00 00 00 00  |#...eee.........|
00069800  00 00 00 00 24 00 00 00  73 6e 64 73 74 6f 70 00  |....$...sndstop.|
00069810  00 00 00 00 00 00 00 00  25 00 00 00 64 64 65 00  |........%...dde.|
00069820  00 00 00 00 00 00 00 00  00 00 00 00 26 00 00 00  |............&...|
00069830  66 6f 72 6d 61 74 74 66  00 00 00 00 00 00 00 00  |formattf........|
00069840  27 00 00 00 68 65 6c 70  00 00 00 00 00 00 00 00  |'...help........|
00069850  00 00 00 00 28 00 00 00  2a 65 38 31 2a 00 00 00  |....(...*e81*...|
00069860  00 00 00 00 00 00 00 00  29 00 00 00 2a 65 38 30  |........)...*e80|
00069870  2a 00 00 00 00 00 00 00  00 00 00 00 2a 00 00 00  |*...........*...|
00069880  2a 72 65 62 6f 6f 74 2a  00 00 00 00 00 00 00 00  |*reboot*........|
00069890  2b 00 00 00 2a 33 36 34  36 36 35 35 2a 00 00 00  |+...*3646655*...|
000698a0  00 00 00 00 2c 00 00 00  69 6d 65 69 73 65 74 00  |....,...imeiset.|
```

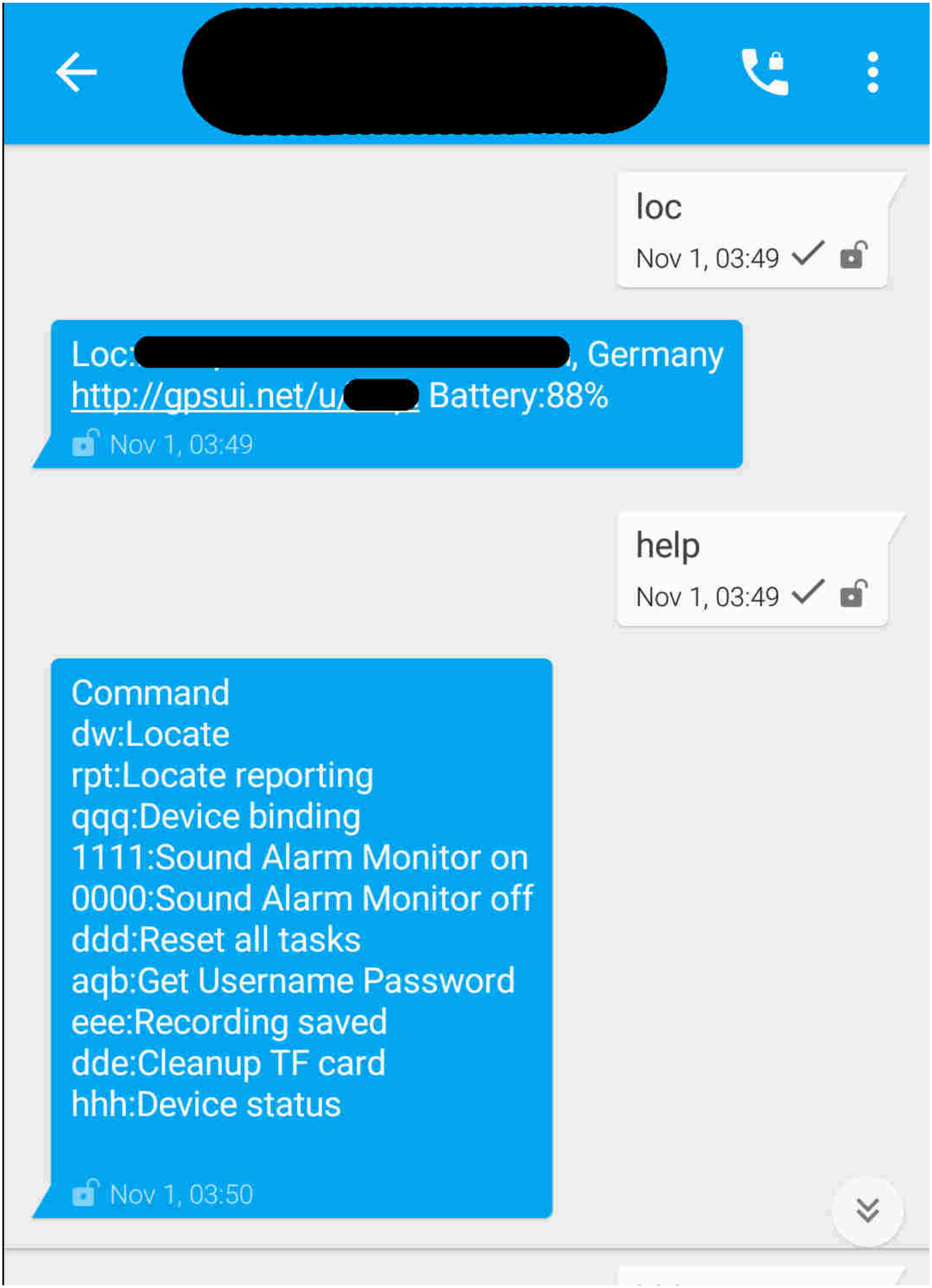
However, most of those commands do not function correctly. It seems the devices firmware is shared among several such location tracking and listening devices, e.g., there are commands referring to LEDs and a TF card, both of which this device do no feature, however, other devices available online do.

An incomplete list of the found commands and there replies is:

- `help`: replies with the following commands:
    - `dw`: Locate
    - `qqq`: Device binding
    - `1111`: Sound Alarm Monitor on
    - `0000`: Sound Alarm Monitor off
    - `ddd`: Reset all tasks
    - `aqb`: Get Username Password
    - `eee`: Recording saved
    - `dde`: Cleanup TF card
    - `hhh`: Device status
- `loc`: same as `dw`
- `imsi`: Query IMEI and IMSI
- `faa`: “DTMG: Set voice monitoring, SMS reply and sound sensitivity successfully:40”, “DTMG: Unusual sound detected”
- `fff`: “DT: Set voice monitoring, voice callback and sound sensitivity successfully:40”
- `1111`: “DT: Set voice monitoring, voice callback and sound sensitivity successfully:400”
- `0000`: “DT: Voice monitoring cancelled successfully.”
- `gbrpt`: “Report:Location the continuous escalation has been closed.”
- `track`: “Track:Caller answer mode the device is set to reply location.”
- `hbton`: “Hbt:Device is turned on real-time online”
- `hbtoff`: “Hbt: Device online has been closed”
- `esionow`: “…” ?
- `esioaddr`: “Setting esio addr and port fail!”
- `esiolocatetype`: “Esio:Reporting location type has been updated to 0.”
- `server`: “Setting server addr and port fail!”
- `reg`: “…” ?
- `monitor`: “Monitor:Caller answer mode the device is set to automatically answer.”
- `eee`: “Tf-Card check fails of is insufficient free space!”
- `sndstop`: “Cam:No task is running, cancel failed!”
- `*e81*`: “…” ?
- `*e80*`: “…” ?
- `soson`, `sosoff`, `sos`, `qcsos`: ?
- `ledon`, `ledoff`: ?
- `flighton`, `flightoff`: ?
- `aqe`: “Setting apn fail!”
- `imeiset`: “…” does not seem to set the IMEI
- `restore`: “Restore ok!”
- `formattf`: ?
- `time`: “…” ?
- `timezone`: “Setting time zone ok. Current time zone 0”
- `age`: “…” ?
- `*3646655*`: queries for version information
- `*reboot*`: reboots the device

Interestingly the reply strings could not be found in the flash in plaintext. This suggests that some of the data is compressed.

The message log of me trying some of the found hidden commands to populate the above list is as follows:



[redacted]

loc

Nov 1, 03:49 ✓

Loc:[redacted], Germany
http://gpsui.net/u/[redacted] Battery:88%

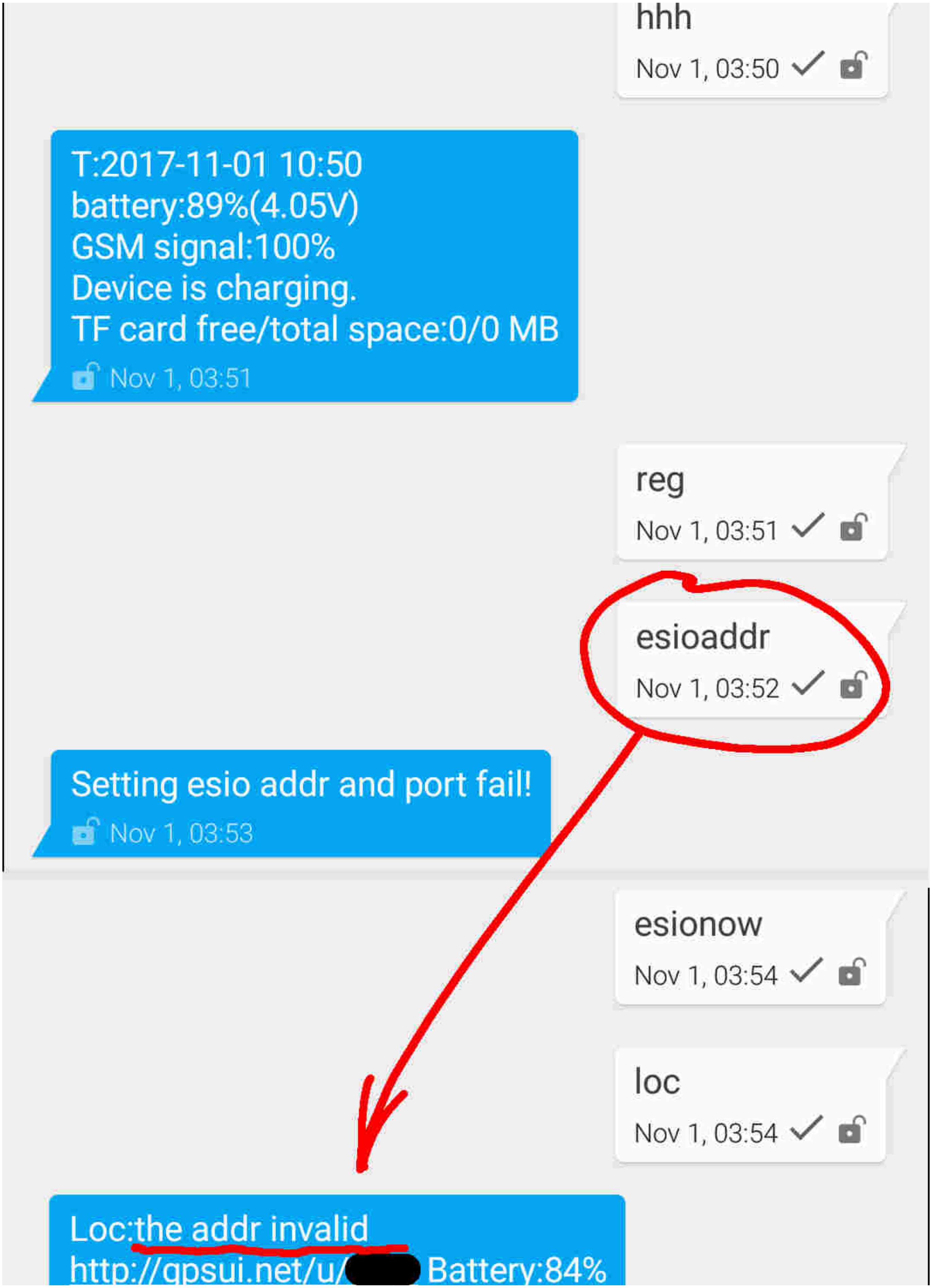
Nov 1, 03:49

help

Nov 1, 03:49 ✓

Command
dw:Locate
rpt:Locate reporting
qqq:Device binding
1111:Sound Alarm Monitor on
0000:Sound Alarm Monitor off
ddd:Reset all tasks
aqb:Get Username Password
eee:Recording saved
dde:Cleanup TF card
hhh:Device status

Nov 1, 03:50



hhh
Nov 1, 03:50 ✓

T:2017-11-01 10:50
battery:89%(4.05V)
GSM signal:100%
Device is charging.
TF card free/total space:0/0 MB
Nov 1, 03:51

reg
Nov 1, 03:51 ✓

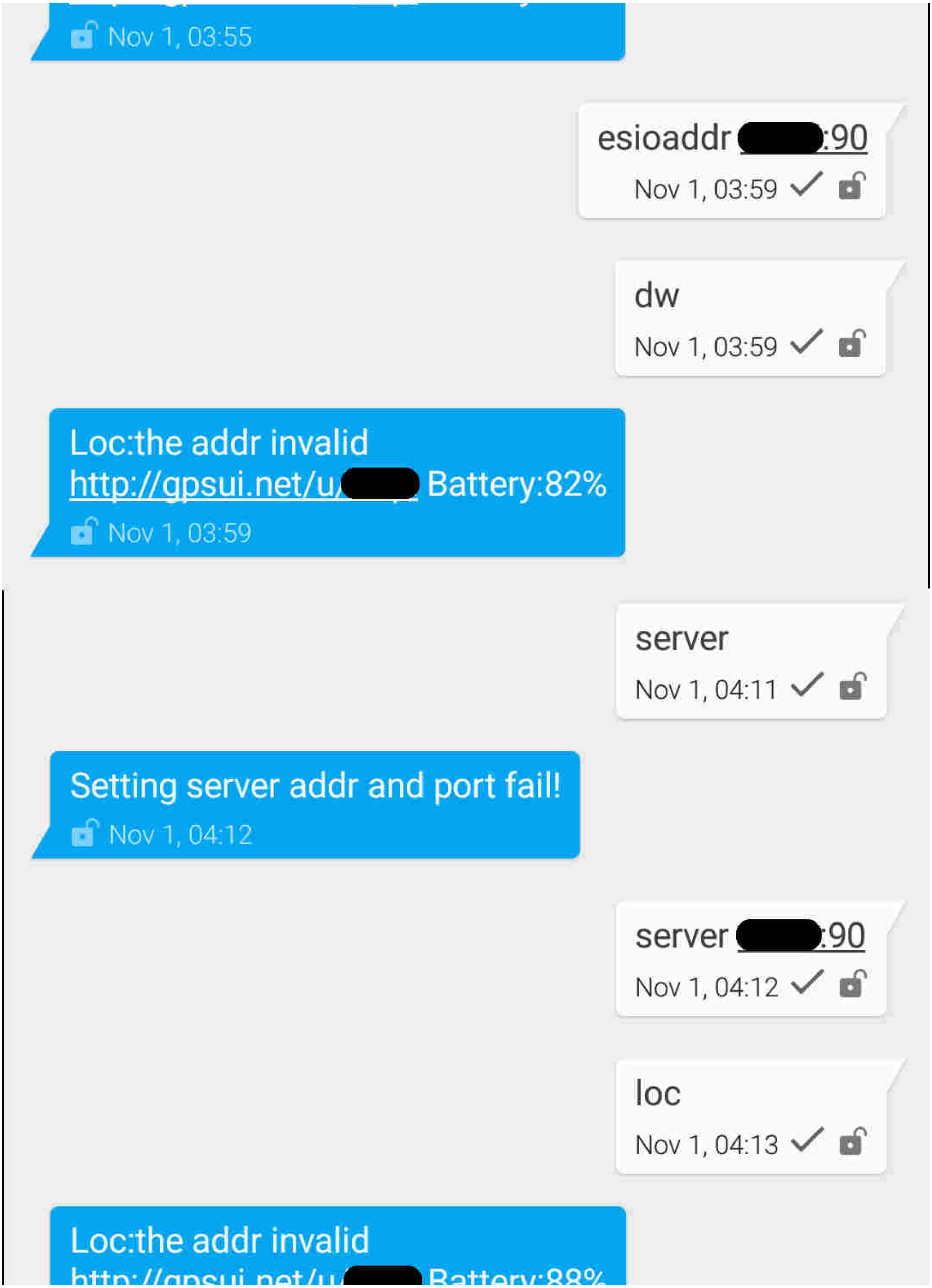
esioaddr
Nov 1, 03:52 ✓

Setting esio addr and port fail!
Nov 1, 03:53

esionow
Nov 1, 03:54 ✓

loc
Nov 1, 03:54 ✓

Loc:the addr invalid
http://gpsui.net/u/[redacted] Battery:84%



Nov 1, 03:55

esioaddr [redacted]:90

Nov 1, 03:59 ✓

dw

Nov 1, 03:59 ✓

Loc:the addr invalid
http://gpsui.net/u/[redacted] Battery:82%

Nov 1, 03:59

server

Nov 1, 04:11 ✓

Setting server addr and port fail!

Nov 1, 04:12

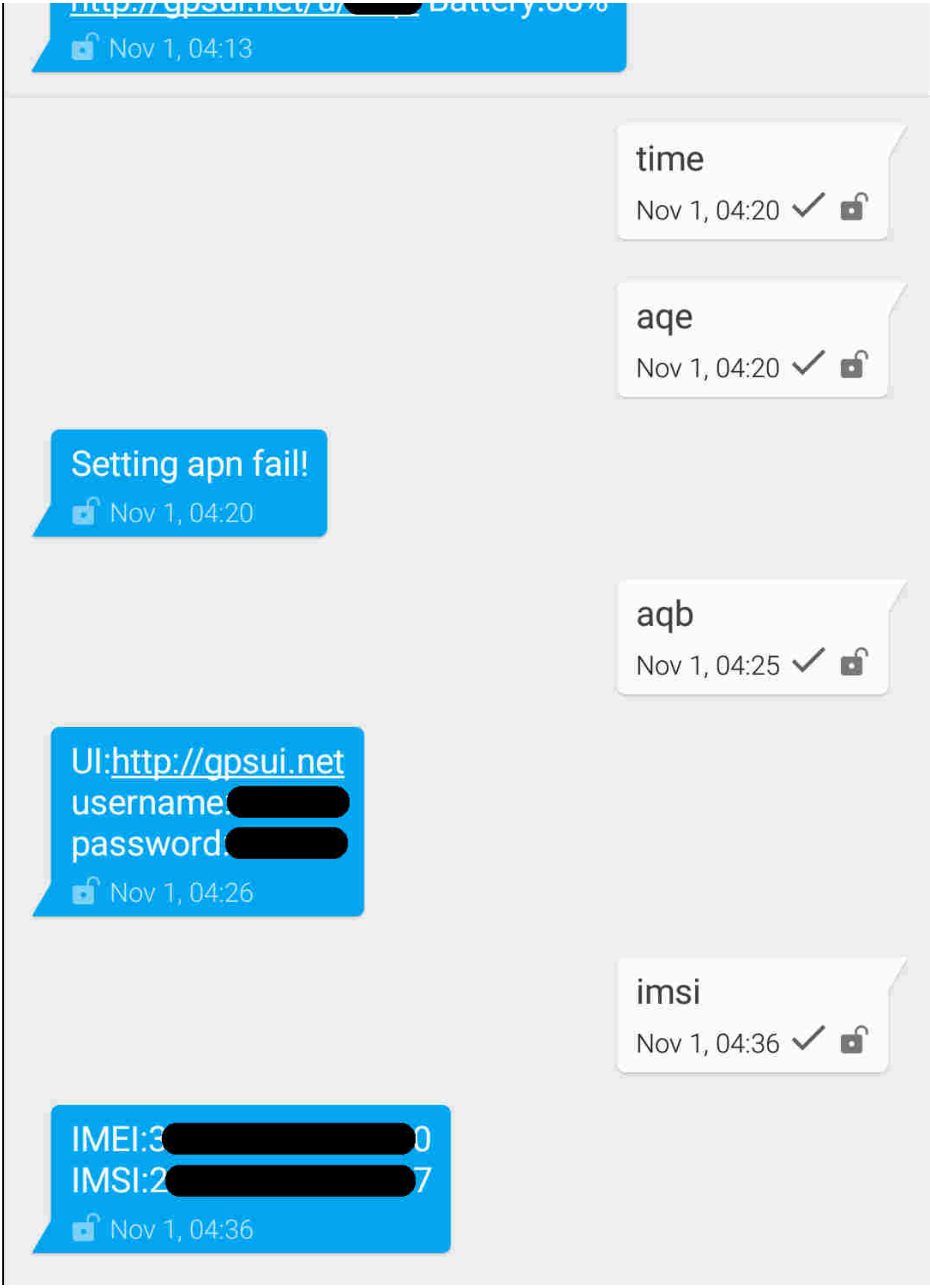
server [redacted]:90

Nov 1, 04:12 ✓

loc

Nov 1, 04:13 ✓

Loc:the addr invalid
http://gpsui.net/u/[redacted] Battery:88%



http://gpsui.net/u/[redacted] Battery:88%

Nov 1, 04:13

time

Nov 1, 04:20 ✓

aqe

Nov 1, 04:20 ✓

Setting apn fail!

Nov 1, 04:20

aqb

Nov 1, 04:25 ✓

UI:http://gpsui.net
username:[redacted]
password:[redacted]

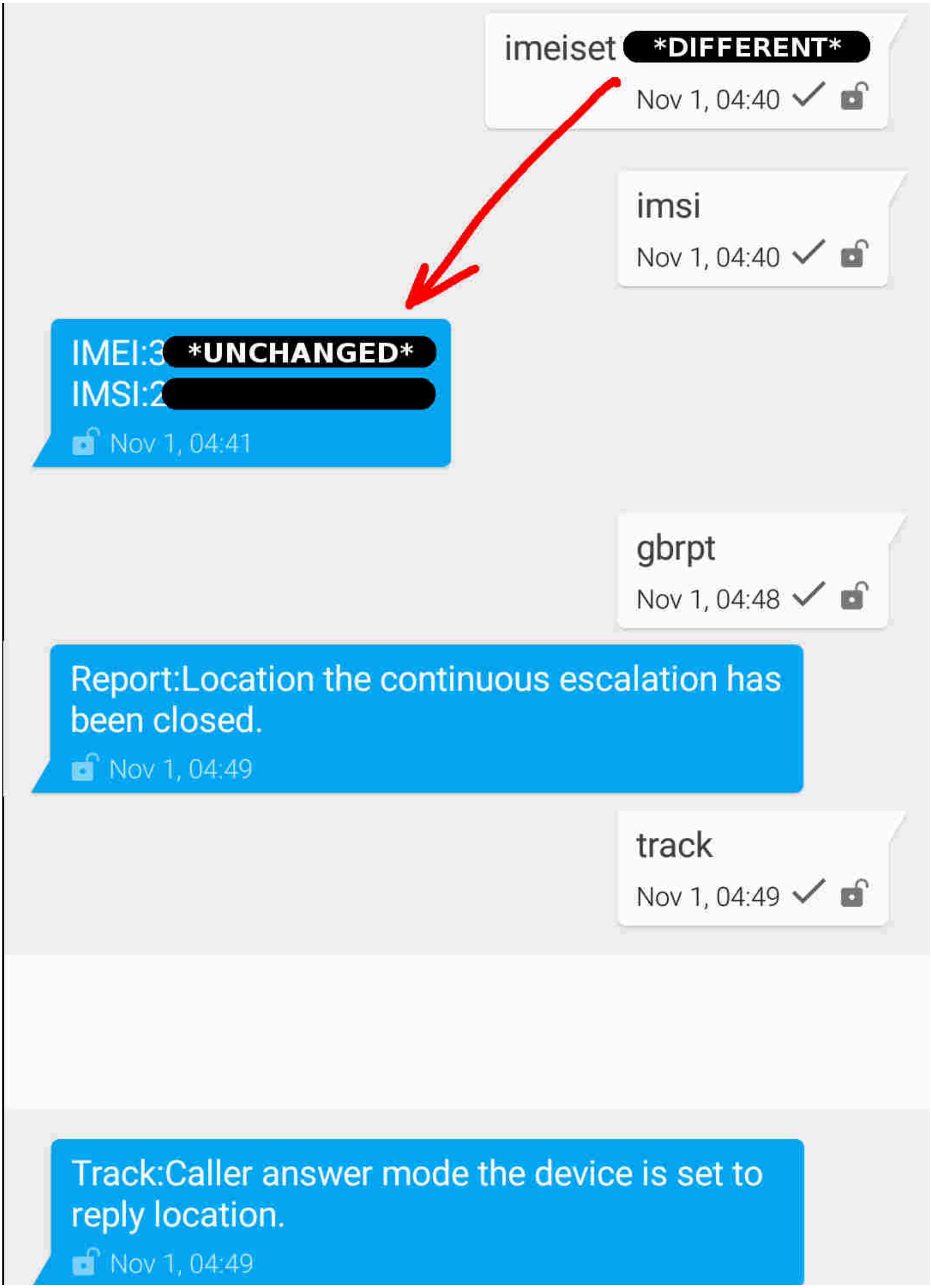
Nov 1, 04:26

imsi

Nov 1, 04:36 ✓

IMEI:3[redacted]0
IMSI:2[redacted]7

Nov 1, 04:36



imeiset *DIFFERENT*

Nov 1, 04:40 ✓

imsi

Nov 1, 04:40 ✓

IMEI:3*UNCHANGED*
IMSI:2[redacted]

Nov 1, 04:41

gbrpt

Nov 1, 04:48 ✓

Report:Location the continuous escalation has been closed.

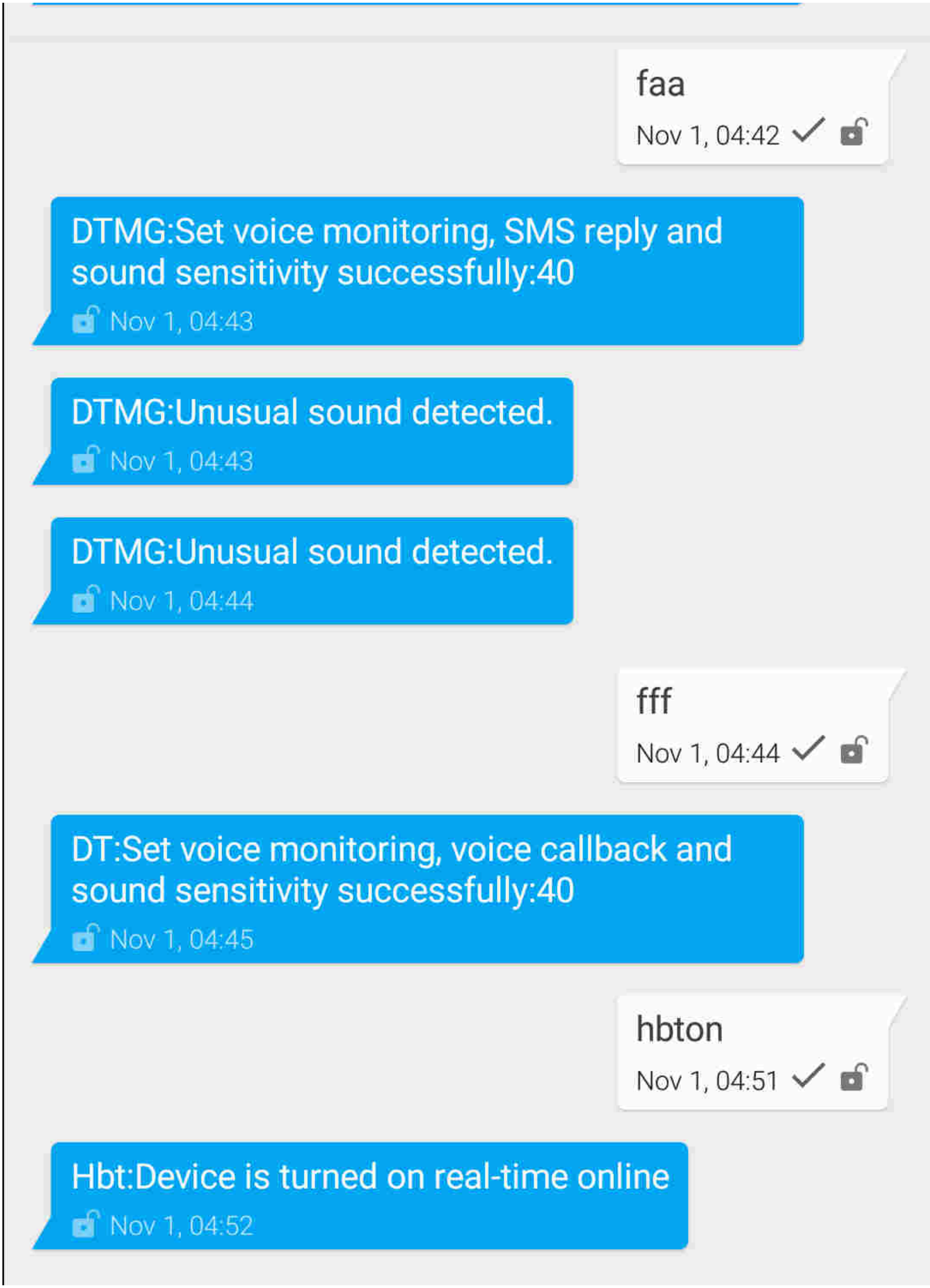
Nov 1, 04:49

track

Nov 1, 04:49 ✓

Track:Caller answer mode the device is set to reply location.

Nov 1, 04:49



faa

Nov 1, 04:42 ✓

DTMG:Set voice monitoring, SMS reply and sound sensitivity successfully:40

Nov 1, 04:43

DTMG:Unusual sound detected.

Nov 1, 04:43

DTMG:Unusual sound detected.

Nov 1, 04:44

fff

Nov 1, 04:44 ✓

DT:Set voice monitoring, voice callback and sound sensitivity successfully:40

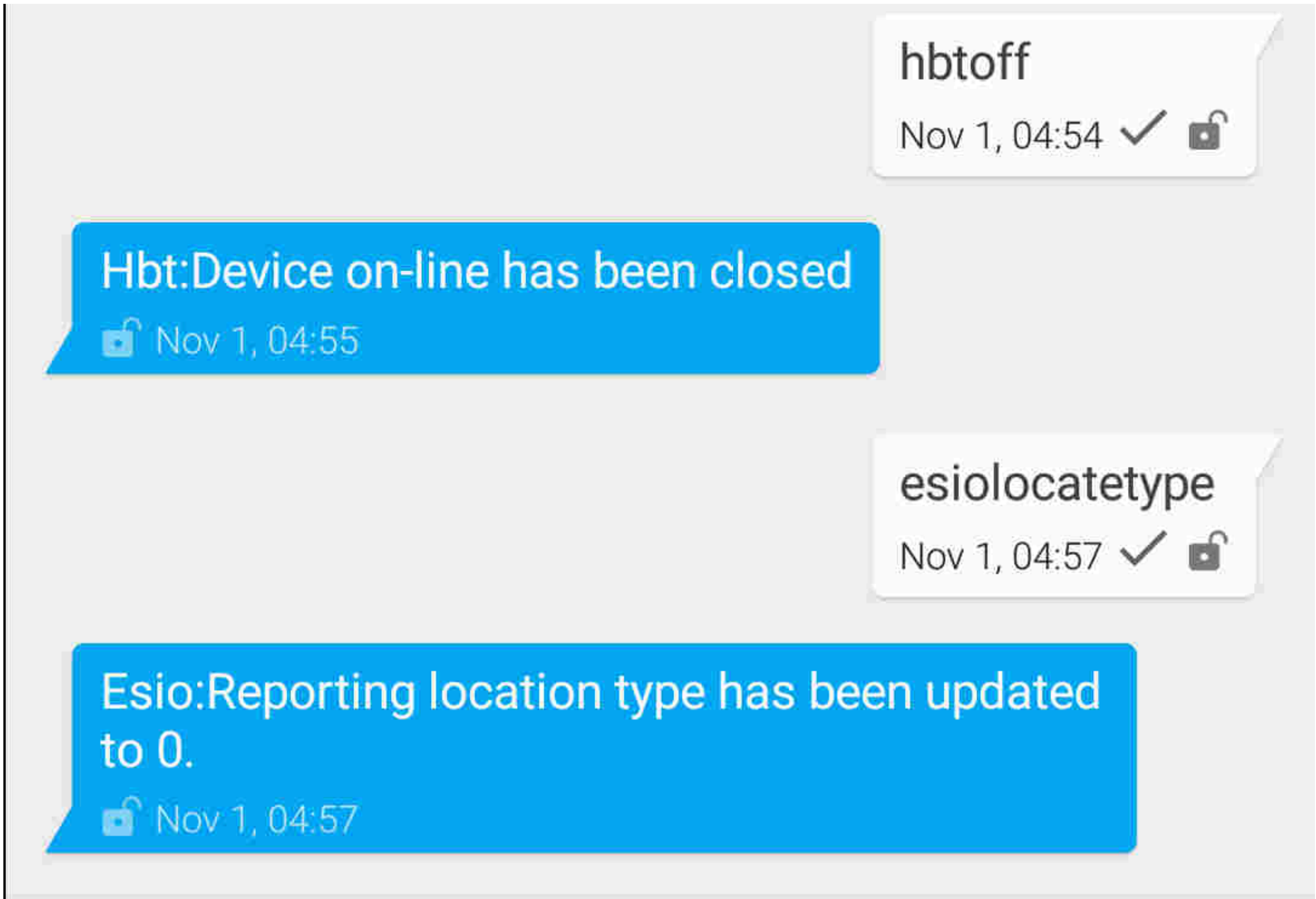
Nov 1, 04:45

hbton

Nov 1, 04:51 ✓

Hbt:Device is turned on real-time online

Nov 1, 04:52

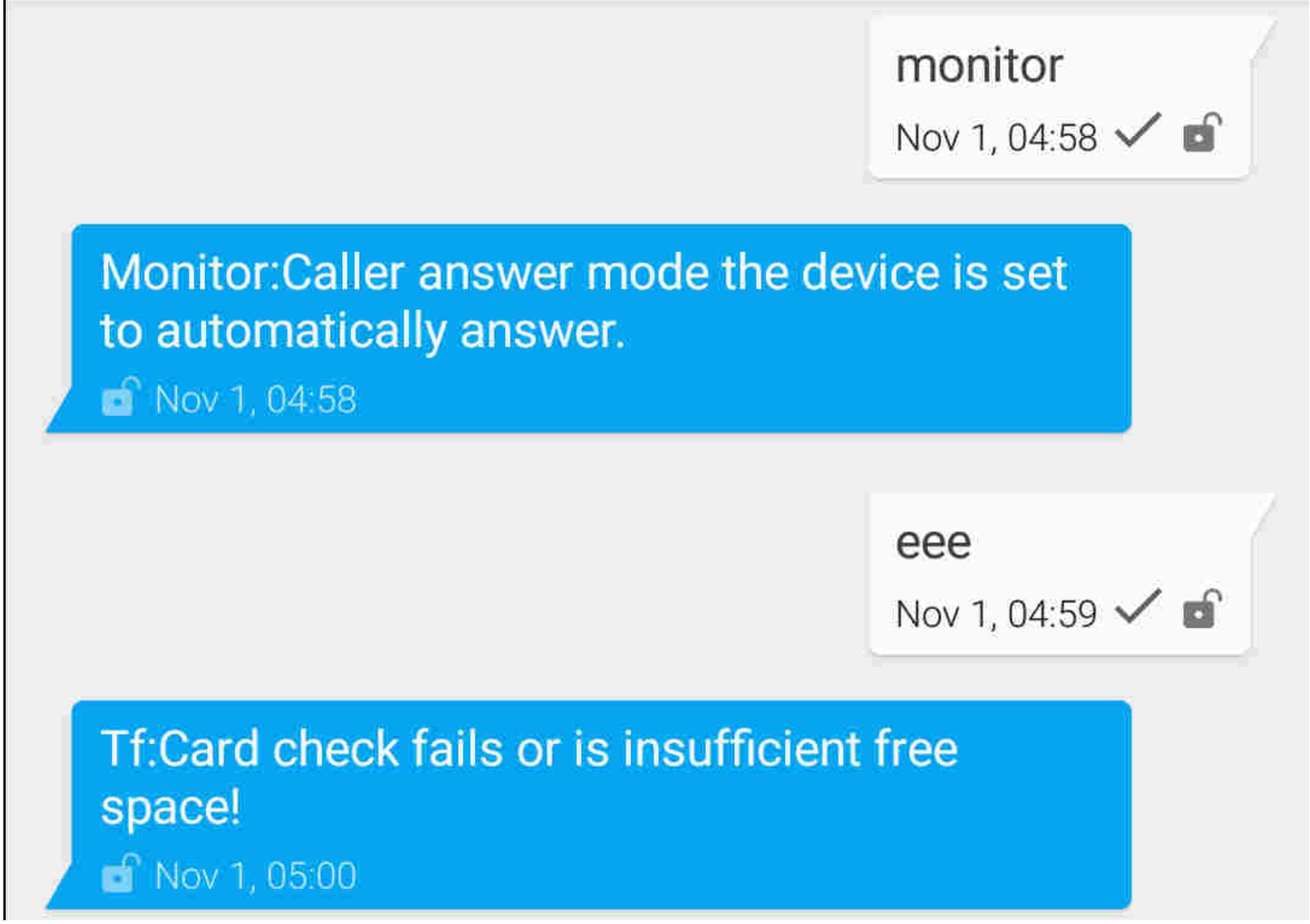


hbtoff
Nov 1, 04:54 ✓ 🔓

Hbt:Device on-line has been closed
🔓 Nov 1, 04:55

esiolocatetype
Nov 1, 04:57 ✓ 🔓

Esio:Reporting location type has been updated to 0.
🔓 Nov 1, 04:57

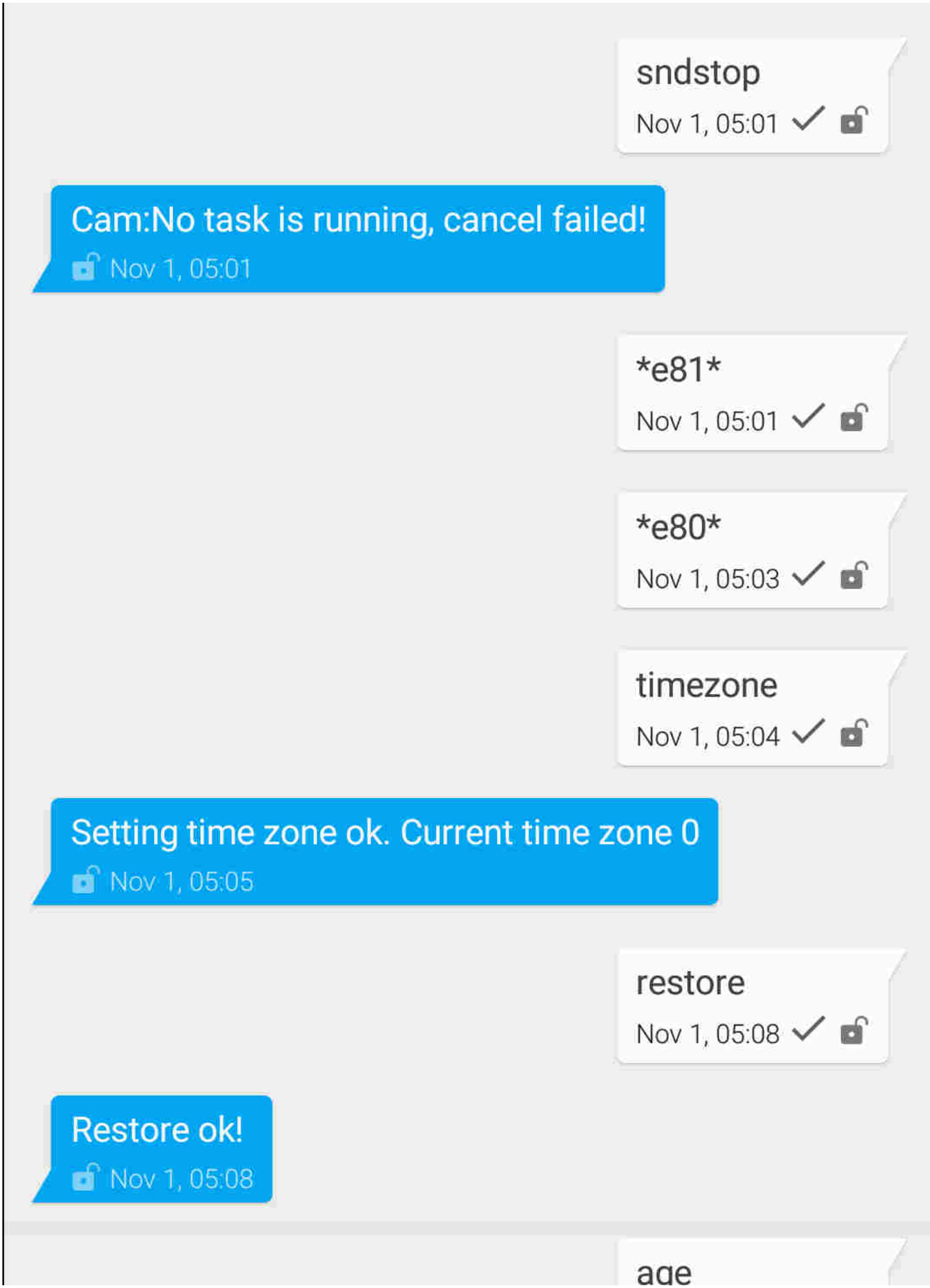


monitor
Nov 1, 04:58 ✓ 🔓

Monitor:Caller answer mode the device is set to automatically answer.
🔓 Nov 1, 04:58

eee
Nov 1, 04:59 ✓ 🔓

Tf:Card check fails or is insufficient free space!
🔓 Nov 1, 05:00



sndstop
Nov 1, 05:01 ✓

Cam:No task is running, cancel failed!
Nov 1, 05:01

e81
Nov 1, 05:01 ✓

e80
Nov 1, 05:03 ✓

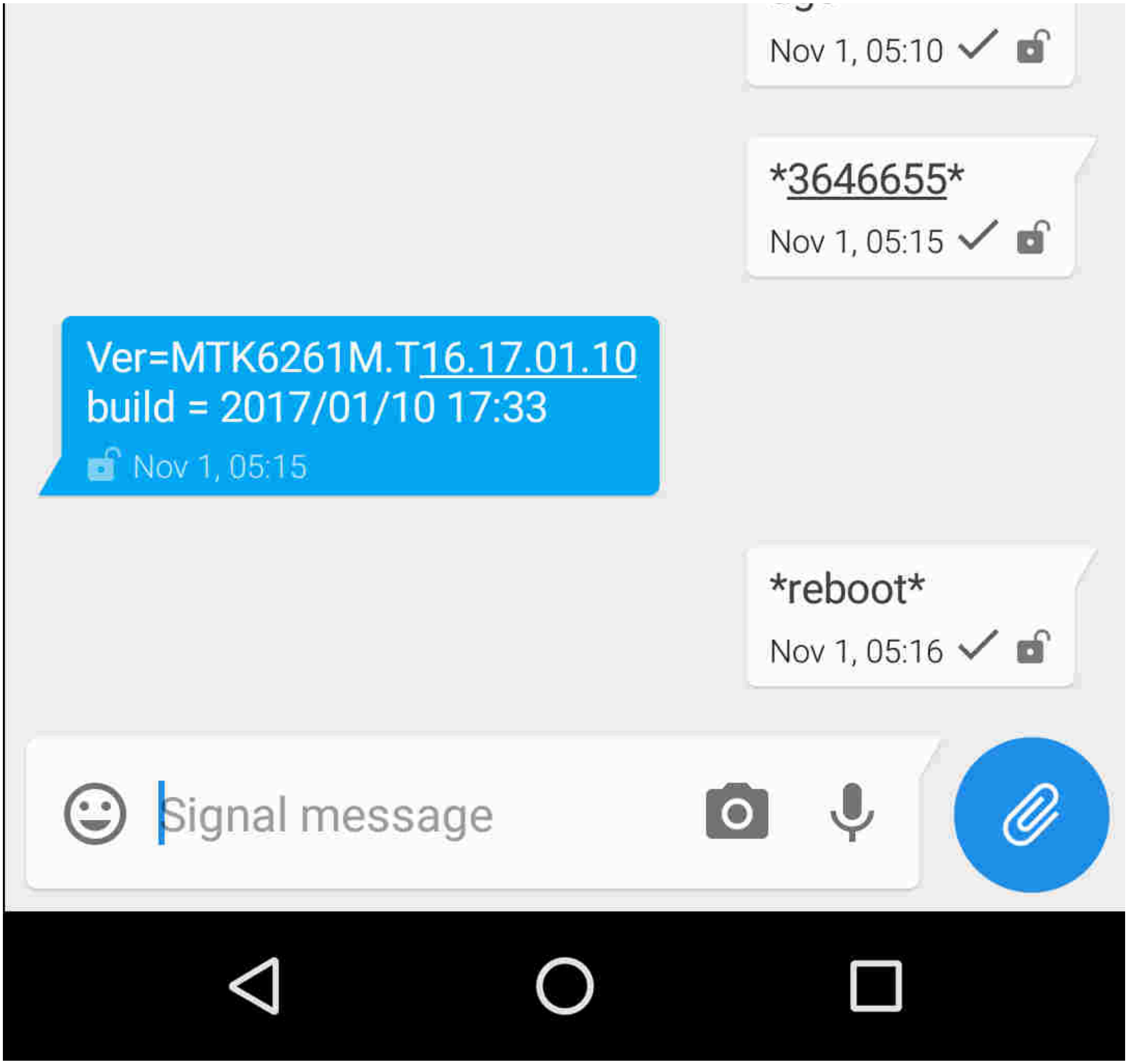
timezone
Nov 1, 05:04 ✓

Setting time zone ok. Current time zone 0
Nov 1, 05:05

restore
Nov 1, 05:08 ✓

Restore ok!
Nov 1, 05:08

age



Nov 1, 05:10 ✓

3646655

Nov 1, 05:15 ✓

Ver=MTK6261M.T16.17.01.10
build = 2017/01/10 17:33

Nov 1, 05:15

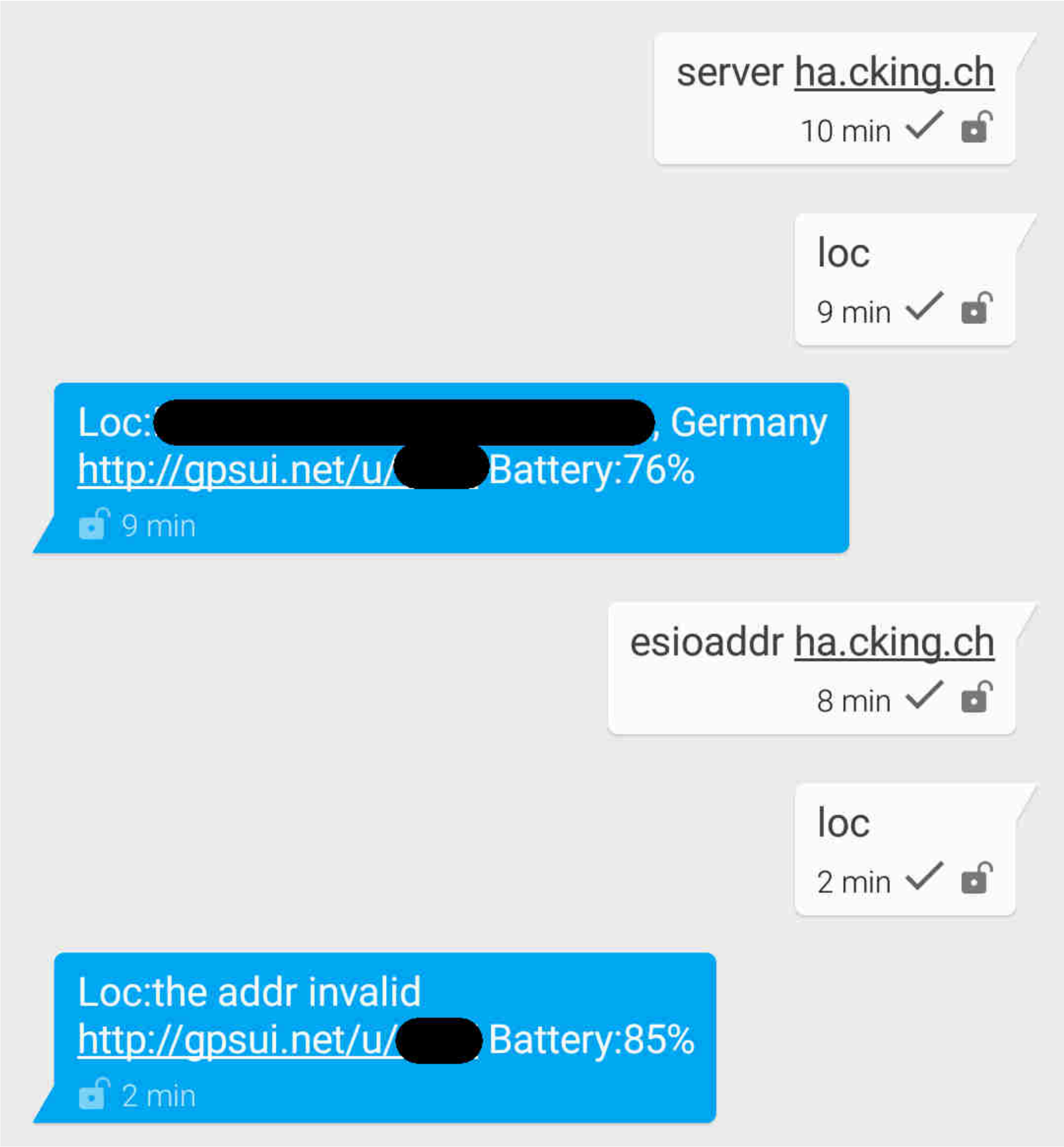
reboot

Nov 1, 05:16 ✓

(images/10_01_cmds.jpg)

It seems that we can use `esioaddr` to change the address used to lock up the location information. However, no connection to a given domain nor IP is actually made. The device will simply report `the addr invalid` in the location report.

The `server` command sets a different server. Changing it does not result in `the addr invalid` responses, as can be seen from this second message log:



server ha.cking.ch

10 min ✓

loc

9 min ✓

Loc:[redacted], Germany
http://gpsui.net/u/[redacted] Battery:76%

9 min

esioaddr ha.cking.ch

8 min ✓

loc

2 min ✓

Loc:the addr invalid
http://gpsui.net/u/[redacted] Battery:85%

2 min

(images/10_14_cmds.jpg)

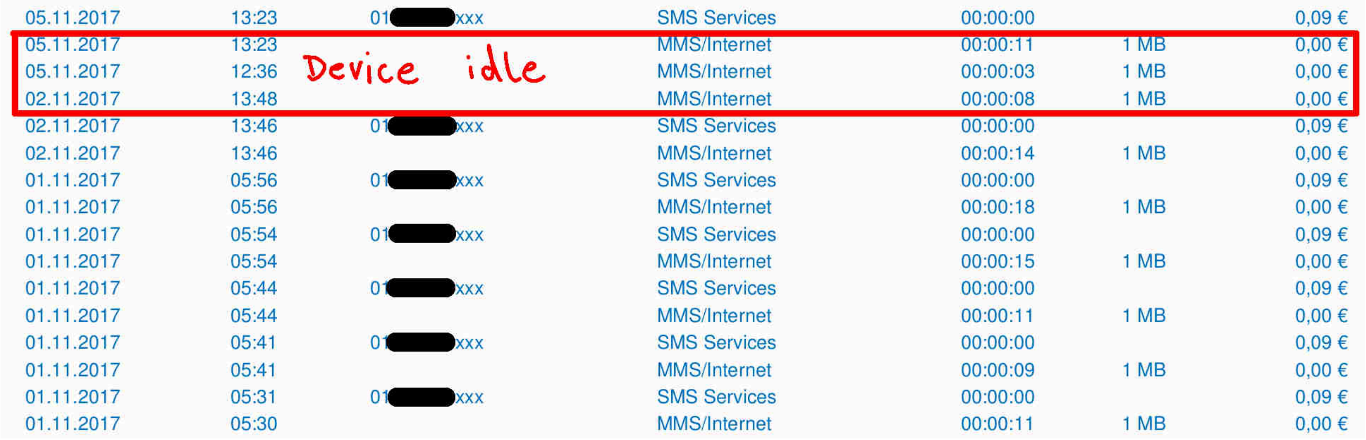
Provider call logs and itemized bill

Because the GPRS sniffing failed I resort to the billing of my provider to further analyze the communication habits of the S8 data line locator.

Obviously reply SMS are billed. More interesting is the Internet access patterns.

`dw` or `loc` commands and during idle

During location queries the device will use “MMS/Internet” service. The following is a segment in which first repeated location queries were performed, then the devices was left idle:



05.11.2017	13:23	01[redacted]xxx	SMS Services	00:00:00		0,09 €
05.11.2017	13:23		MMS/Internet	00:00:11	1 MB	0,00 €
05.11.2017	12:36	Device idle	MMS/Internet	00:00:03	1 MB	0,00 €
02.11.2017	13:48		MMS/Internet	00:00:08	1 MB	0,00 €
02.11.2017	13:46	01[redacted]xxx	SMS Services	00:00:00		0,09 €
02.11.2017	13:46		MMS/Internet	00:00:14	1 MB	0,00 €
01.11.2017	05:56	01[redacted]xxx	SMS Services	00:00:00		0,09 €
01.11.2017	05:56		MMS/Internet	00:00:18	1 MB	0,00 €
01.11.2017	05:54	01[redacted]xxx	SMS Services	00:00:00		0,09 €
01.11.2017	05:54		MMS/Internet	00:00:15	1 MB	0,00 €
01.11.2017	05:44	01[redacted]xxx	SMS Services	00:00:00		0,09 €
01.11.2017	05:44		MMS/Internet	00:00:11	1 MB	0,00 €
01.11.2017	05:41	01[redacted]xxx	SMS Services	00:00:00		0,09 €
01.11.2017	05:41		MMS/Internet	00:00:09	1 MB	0,00 €
01.11.2017	05:31	01[redacted]xxx	SMS Services	00:00:00		0,09 €
01.11.2017	05:30		MMS/Internet	00:00:11	1 MB	0,00 €

(images/11_00_provider_logs.jpg)

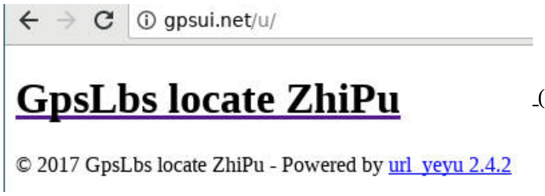
Even during idle the device sometimes uses the “MMS/Internet” service.

Even though I deactivated all tracking features that I may have activated during my previous testing, I can not be 100 % sure that this is not something that I activated, maybe while stumbling through the gpsui.net website. However, I regardless of whether I activated this “feature” or not, I do not want it and would like to know what data is actually send and how to deactivate it.

gpsui.net

Going deeper into the gpsui.net website would probably result in a new writeup in itself. It is a very big surveillance hub, just replace the xxxx in http://gpsui.net/u/xxxx with some letters and numbers and you can see random people’s locations.

The website also makes a reference to ZhiPu:



gpsui.net/u/

GpsLbs locate ZhiPu

© 2017 GpsLbs locate ZhiPu - Powered by url_yeyu 2.4.2

(images/12_gpsui.net.jpg)

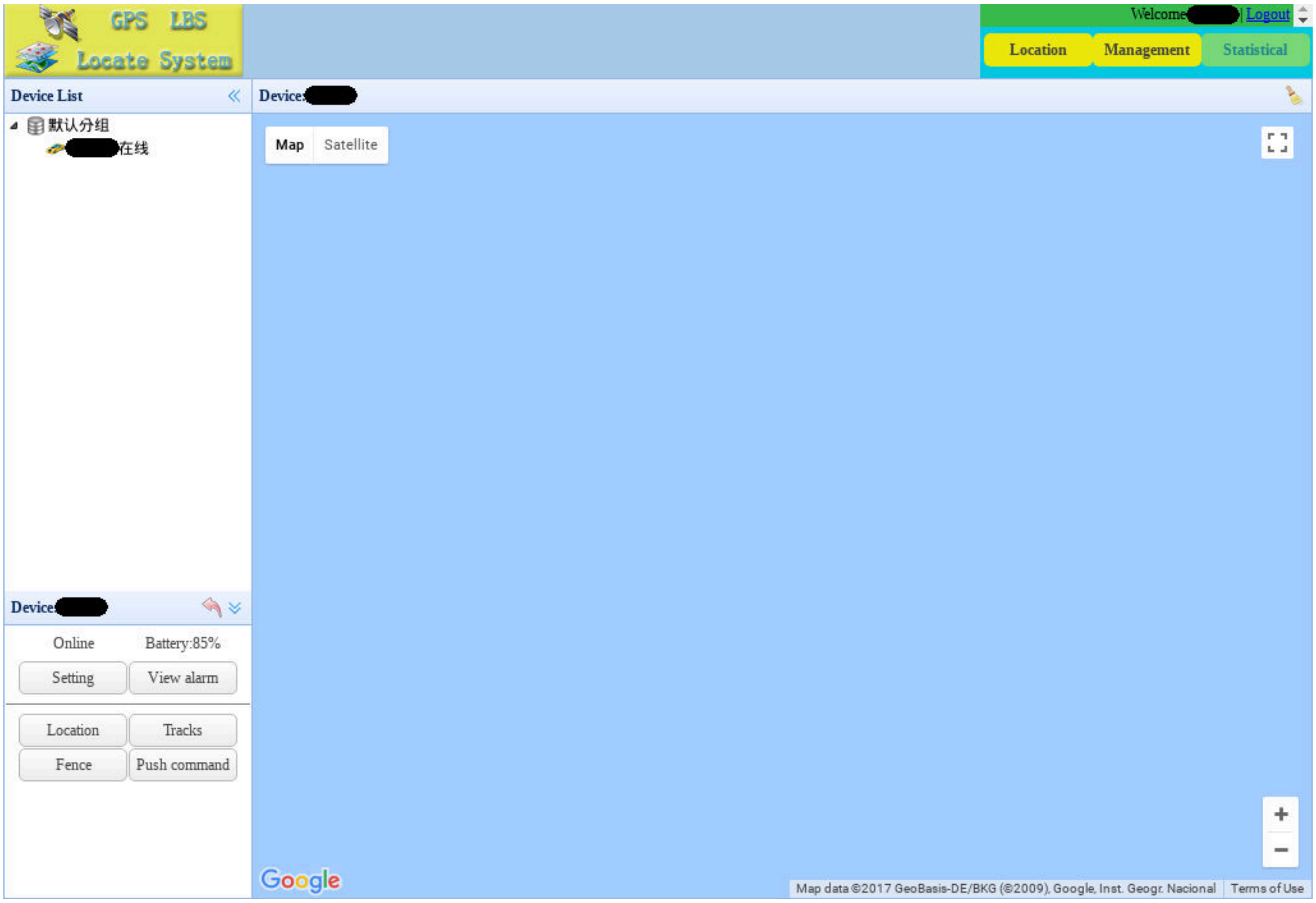
It seems this is the company that makes these trackers.

You can get your credentials for login by texting `aqb` to your S8 data line monitor. The username as well as the password are 6 digit numbers. They are also located in the flash right before the IMSI.

The web interface allows access to several features, which may or may not work as I’m not interested in them and hence not tested all of them:

Interface

The plain interface after login looks like this:



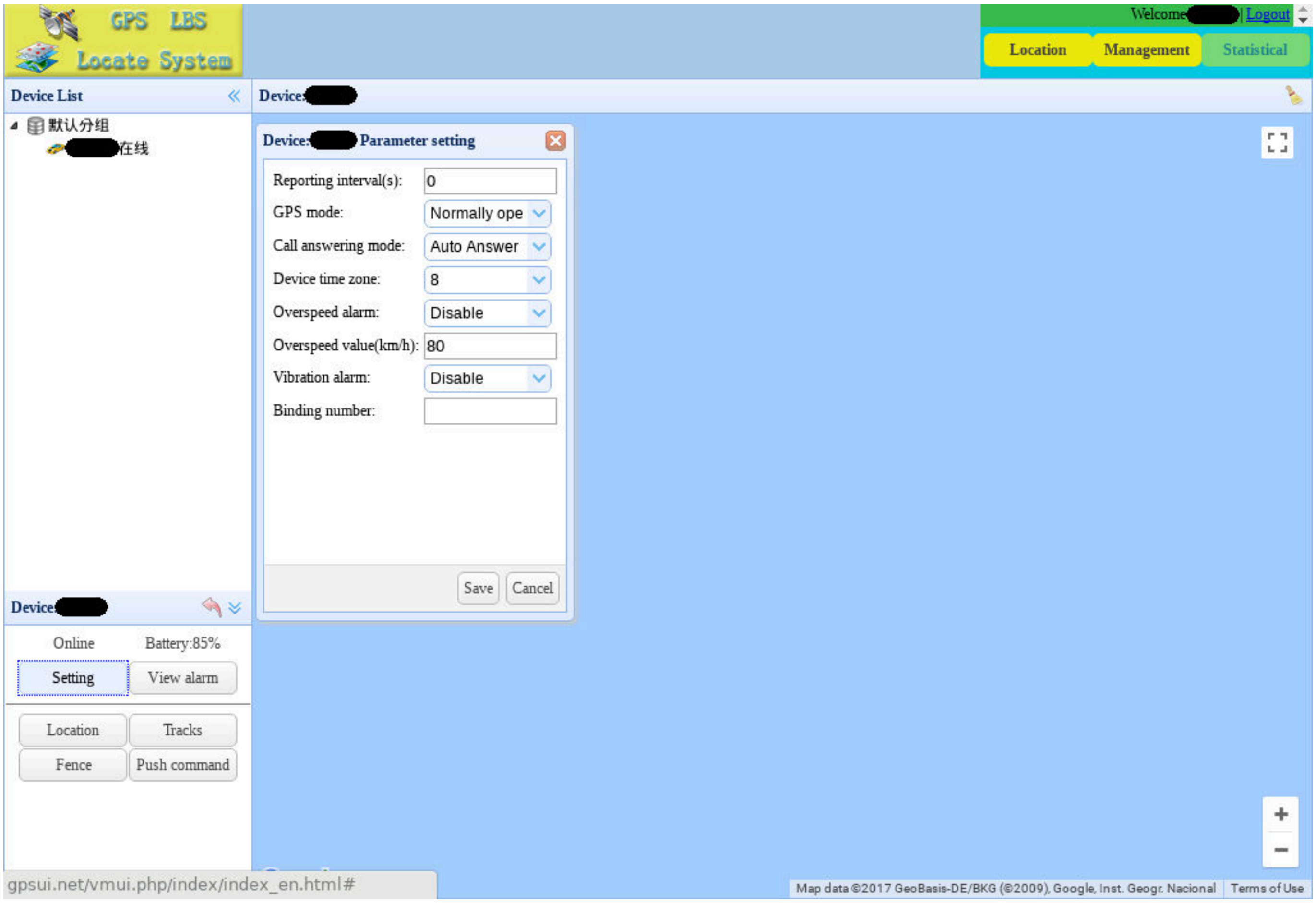
Online Battery:85%

(images/13_00_interface.jpg)

I panned the map into the ocean to hide my location. The location is pinpointed on the map by a little car and a pop up containing the GPS data and date of the last location update.

Settings

You can change various settings:



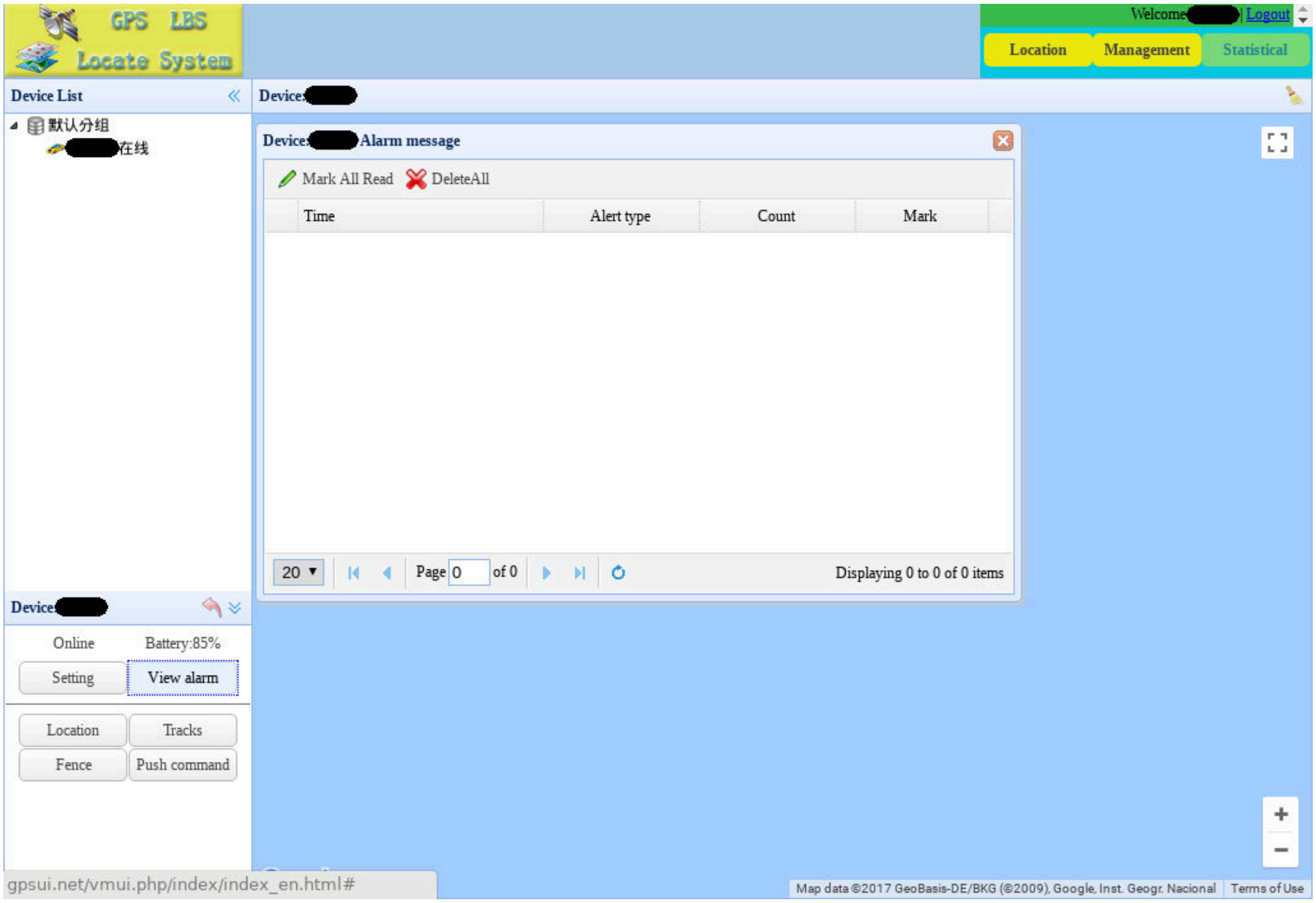
Device:[redacted] Parameter setting

Reporting interval(s):	0
GPS mode:	Normally ope
Call answering mode:	Auto Answer
Device time zone:	8
Overspeed alarm:	Disable
Overspeed value(km/h):	80
Vibration alarm:	Disable
Binding number:	

(images/13_01_setting.jpg)

Alarms

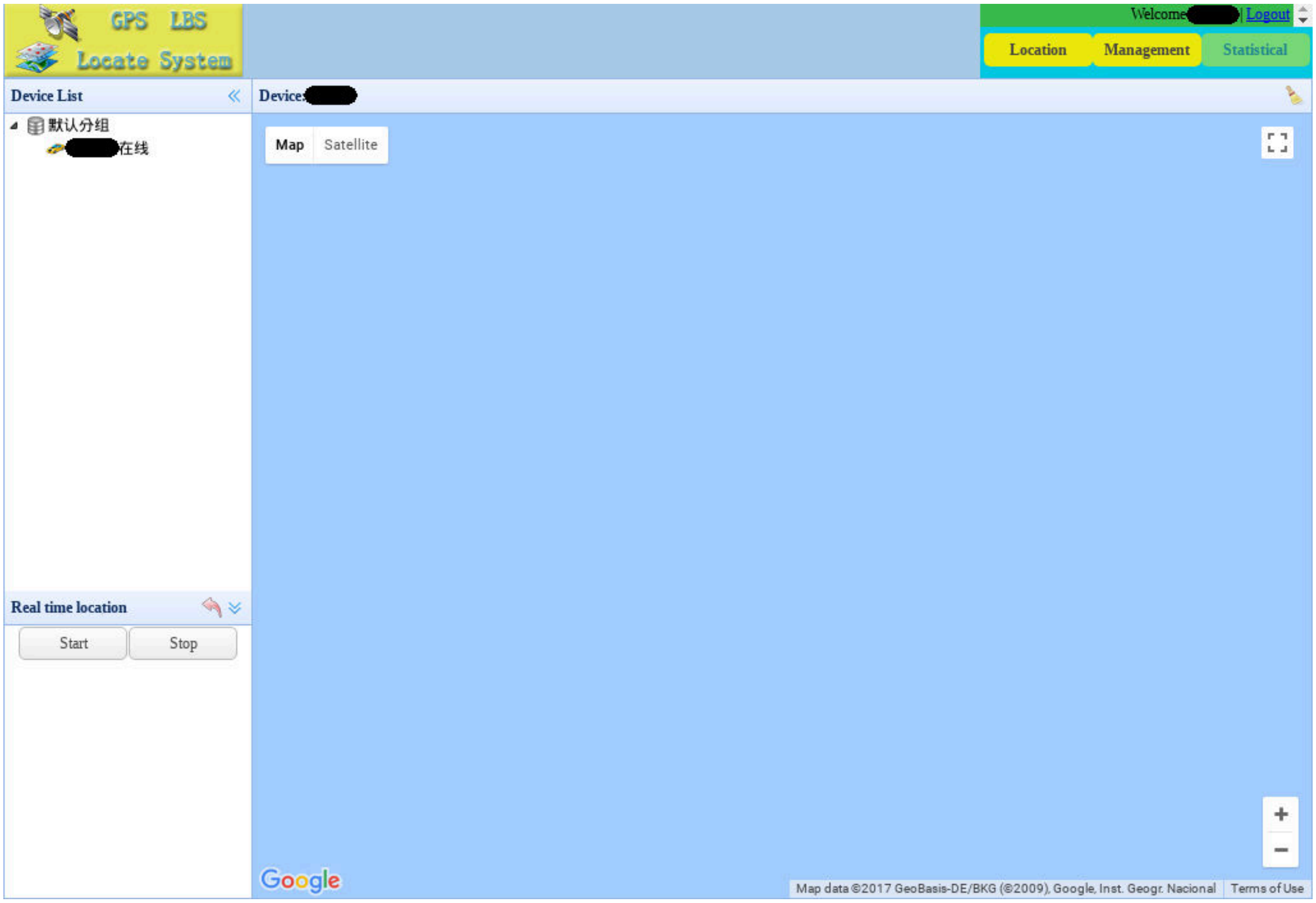
You can setup alarms:



(images/13_02_alarm.jpg)

Real time location tracking

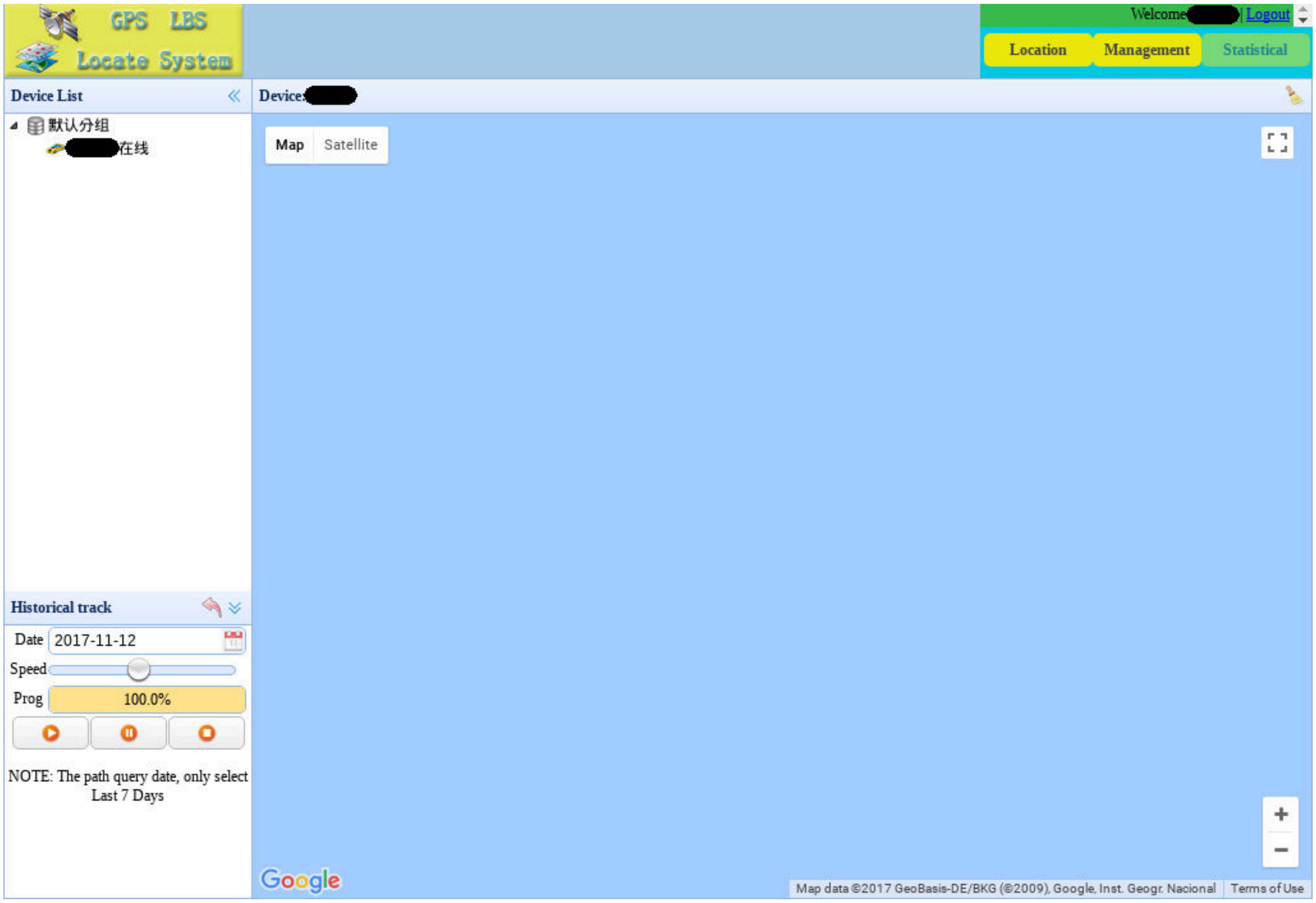
You can enable real time location tracking:



(images/13_03_rtloc.jpg)

History

Interestingly, the web interface allows to playback past location queries, as can be seen here:



Date 2017-11-12

Speed

Prog 100.0%

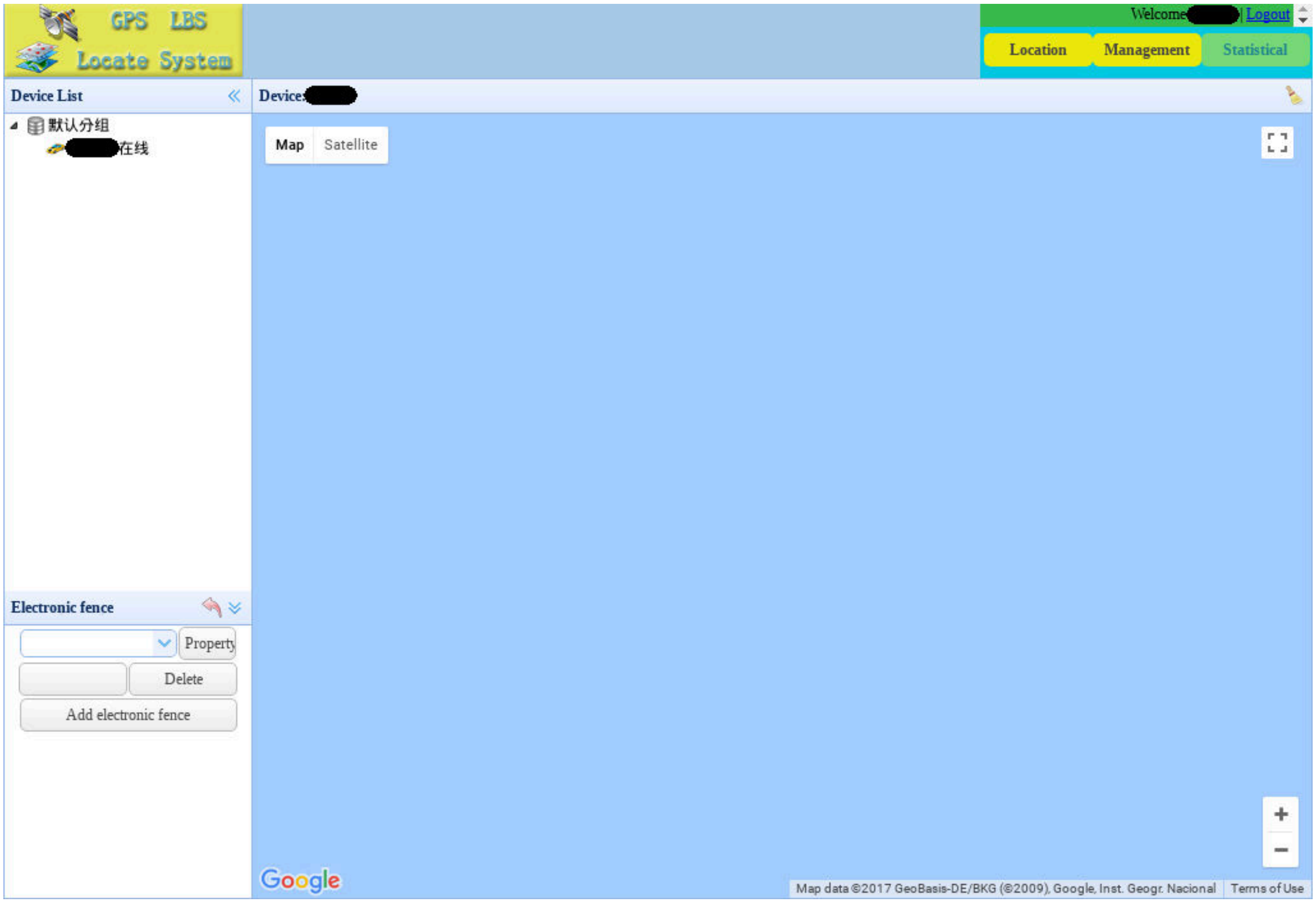
NOTE: The path query date, only select Last 7 Days

(images/13_04_history.jpg)

I find this particularity disturbing because in the original manual of the S8 data line locator there is no mention of the login credentials nor a way to get them. Also I did not expect a location query history to be stored somewhere.

Fence

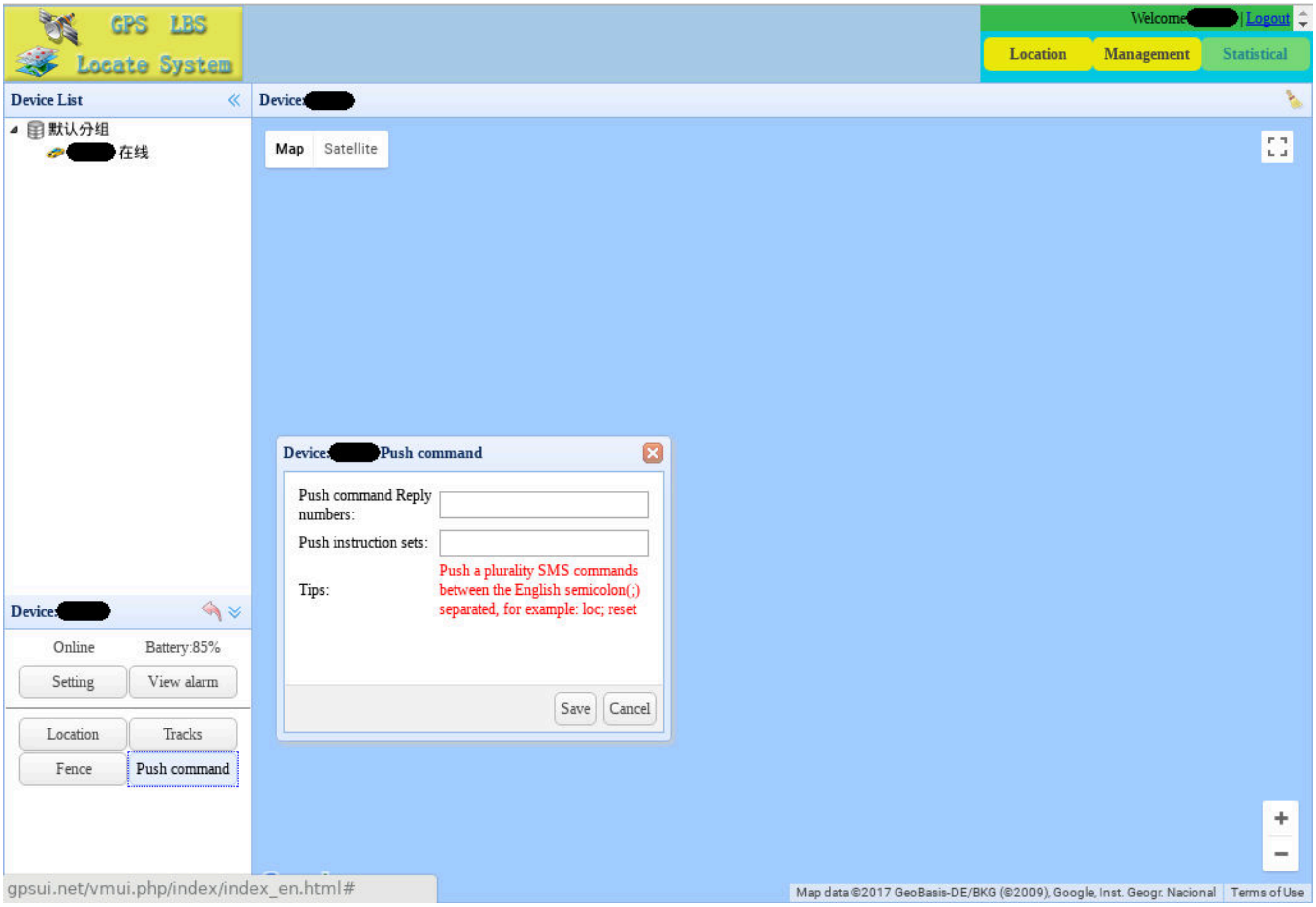
It also allows to set a geo fence:



(images/13_05_fence.jpg)

Push commands

Next, it allows you to push command to the device:



Device:[redacted]Push command

Push command Reply numbers:

Push instruction sets:

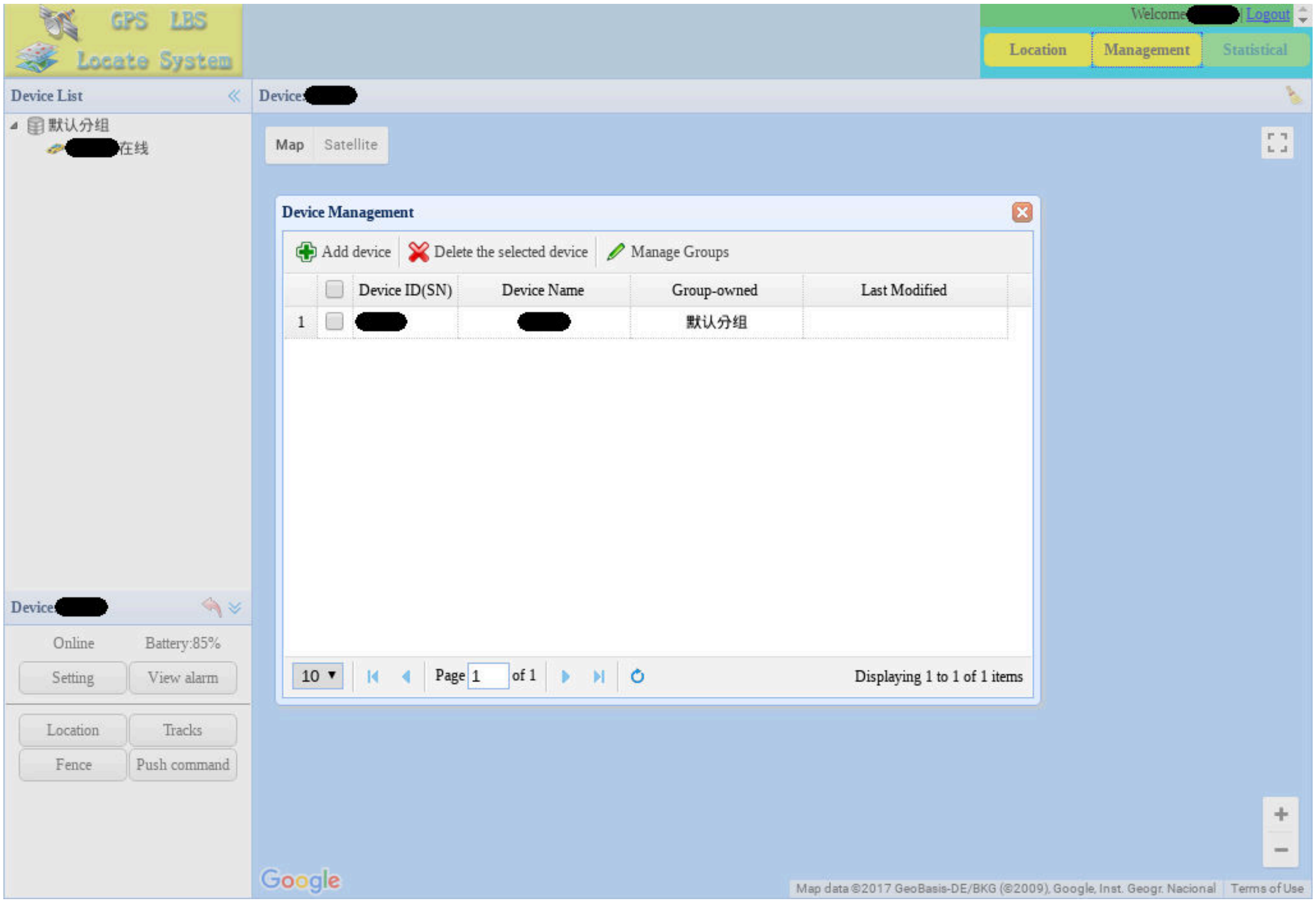
Tips: Push a plurality SMS commands between the English semicolon(;) separated, for example: loc; reset

(images/13_06_push_cmd.jpg)

This means anyone with access to your gpsui.net login credentials can control your device. A device which original packaging nor manual make any reference to said website.

Management

You can also add your fleet of devices into one account for simplified management:



Device Management

	□	Device ID(SN)	Device Name	Group-owned	Last Modified
1	□	[redacted]	[redacted]	默认分组	

Displaying 1 to 1 of 1 items

(images/13_07_mgnmt.jpg)

Vulnerabilities

After publishing this write up, Vangelis Stykas (@evstykas (https://twitter.com/evstykas)) found a bunch of Insecure Direct Object References with Authorization bypass through user-controlled key vulnerabilities leading to Horizontal escalation of privilege (one user can view/modify information of all other 615,817 accounts) in gpsui.net.

So my initial suspicions about gpsui.net were correct: you do not want your data to be send there.

We then also looked at some other (GPS) location tracking web services. What we found was mildly concerning and you can read about it here: https://0x0.li/trackmageddon/ (https://0x0.li/trackmageddon/)

I just want to point out here that a normal user is (at least as far as I know) never informed about the login credentials to the above website.

Neither is the user informed that the location queries are logged and stored there. Nor that the device can be remote controlled from there.

Detection

When sending data the S8 data line locator can be detected with a CC308+ (a cheap Chinese RF detector):

(images/01_cc308+.jpg)

The S8 data line locator seems to be badly shielded. A location request via the `dw` command causes noticeable electronic noise by the device. It seems in general to cause all sorts of RF interference.

Future work

While I did not (yet) succeed with my original goal to disable the mobile data phone home “feature’, it was nevertheless a fun exercise and hopefully someone finds this useful or at least educational.

Future work needs to be done on several things:

Issues

I was not able yet to write new firmware via flashrom because I was not able to disable block protection on the flash, yet. Maybe a different avenue for flashing new firmware could be the SPFlash tool[4] and/or the Flash tool. However, that would not be open source. **If you are able to flash your S8 data line locator please contact me with details!**

Further, I tried to capture the GPRS data connection of the device, but was unable to do so. It would not use GPRS when connected to my network. Currently, I do not know how the APN is configured. The SIM trace does not indicate that the EF(ACL) is ever accessed. However, as I found the correct APN configuration stored in the devices flash, this suggests the device acquires this information via a setup SMS by the service provider.

Ideas

Dremel the board smaller, e.g., you don't need the USB connector. This way the S8 data line locator could be turned into a "modular" bug that could be placed where ever there is a 5 V 1 A power source.

Other people working on this

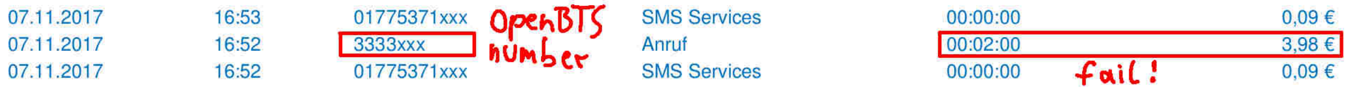
@dmxinajeansuit (https://twitter.com/dmxinajeansuit): http://n0.lol/em/s8 (http://n0.lol/em/s8)

Appendix: Fuck up

No writeup would be complete without at least one fuck up. So here it is:

While using the S8 data line locator with OpenBTS I provisioned imaginary numbers. When switching SIM cards I forgot to turn of the voice activated callback.

So long story short, some guy with the number 3333333 listend in on me for 2 minutes:



07.11.2017	16:53	01775371xxx	SMS Services	00:00:00	0,09 €
07.11.2017	16:52	3333xxx	Anruf	00:02:00	3,98 €
07.11.2017	16:52	01775371xxx	SMS Services	00:00:00	0,09 €

(images/11_01_provider_logs.jpg)

The number appears to be some sort of "service" number, hence it was an expensive call. But more importantly I did not notice this until I reviewed the logs!

So my resume on these little hardware espionage implants: **They are stealthy and dangerous as fuck!**

Appendix: Video

An accompanying video is available at https://www.youtube.com/watch?v=hVOyoIfHA4E (https://www.youtube.com/watch?v=hVOyoIfHA4E)

Bibliography

[1] NSA/CSS, "ANT product catalog (USB)," leaked documents (pdf: https://cryptome.org/2013/12/nsa-ant-usb.pdf (https://cryptome.org/2013/12/nsa-ant-usb.pdf)).

[2] A. "bunnie" Huang and S. X. Cross, "Fernvale: An open hardware and software platform, based on the (nominally) closed-source mT6260 soC," talk at the 31st Chaos Communication Congress (slides: http://www.bunniefoo.com/fernvale/fernvale-31c3.pdf (http://www.bunniefoo.com/fernvale/fernvale-31c3.pdf), video: https://www.youtube.com/watch?v=hpEqDPYtf9s (https://www.youtube.com/watch?v=hpEqDPYtf9s)).

---

1. https://twitter.com/securelyfitz/status/917862004152397826 (https://twitter.com/securelyfitz/status/917862004152397826)↩

2. https://twitter.com/CyberQueenMara/status/925925840205987840 (https://twitter.com/CyberQueenMara/status/925925840205987840)↩

3. https://twitter.com/CyberQueenMara/status/926104852605706240 (https://twitter.com/CyberQueenMara/status/926104852605706240)↩

4. https://spflashtool.com/ (https://spflashtool.com/)↩