Remote Exploitation of CAN BUS

I'm!

Andrei Borcea

20 anni . Grandissimo appassionato di tecnologia fin da piccolo.Amo programmare e risolvere problemi.Solitamente lavoro con Python,C,C++,Java,C#,Javascript(Node e tecnologie varie lato web),SQL,Assembly.Amo ambiente linux lato server ma uso windows quotidianamente(non mi odiate)

Sono un appassionato di sicurezza e passo il tempo a trovare bug,problemi nei codici degli altri.Ho basi minime di Assembly in versioni x86,x64,ARM,ELF.Il reverse engineering è il mio amico da sempre.

Attivo nella comunità del jailbreak Apple, PlayStation jailbreak e tool di utilità generale.

Tool: IDA,Radare2,CFF Explorer,API Monitor,HexRays,Visual Studio e la suite completa di Jetbrains

- Github:@hustlercoder
- Twitter:@hustlercoder_

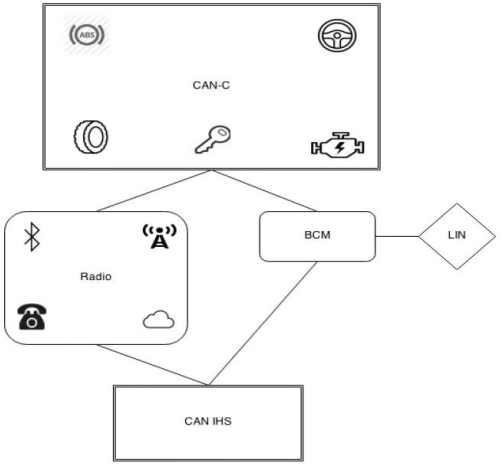
DEFINIZIONI

CAN BUS: Il **Controller Area Network**, noto anche come **CAN-bus**, è uno standard seriale per bus di campo (principalmente in ambiente automotive), di tipo multicast, introdotto negli anni ottanta dalla Robert Bosch GmbH, per collegare diverse unità di controllo elettronico (ECU).

ECU: una **unità di controllo elettronico** (*electronic control unit* o *ECU* in inglese) è un sistema di controllo software che è integrato direttamente nel sistema che controlla.Solitamente la chiamiamo centralina.

CAN-C: sottoinsieme del CAN BUS,questa è la parte a basso livello del CAN BUS che si occupa del controllo di sistemi come ABS, iniettori, ESP e altre tecnologie.

CAN-IHS: seconda categoria di CAN BUS che si occupa della parte del controllo dei dispositivi di confort: AC,WIFI,Bluetooth,Comandi al volante,ecc



Message frame

Field name	Priority					Message type ID													Service not message	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																						0	1...127						
CAN ID bytes	3					2																1	0						

Anonymous message frame

Field name	Priority					Discriminator													Lower bits of message type ID		Service not message	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values																						0	0						
CAN ID bytes	3					2																1	0						

Service frame

Field name	Priority					Service type ID								Request not response	Destination node ID							Service not message	Source node ID						
CAN ID bits	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Allowed values															1...127							1	1...127						
CAN ID bytes	3					2								1								0							

Informazioni sul sistema

- Basato su sistema operativo QNX 32 bit ARM
  - Microcontrollore Texas Instruments OMAP-DM3730 per comunicazione tra CAN IHS e CAN C
  - Sviluppato da Harman Kardon(in alcune versioni)
- File System e servizi
  - IPL(Initial Program Loader) usato per caricare il firmware della ECU
  - IFS-ETFS : IFS contiene l'immagine del sistema operativo con le relative configurazioni, ETFS un file system temporaneo usato nelle operazioni di rw dalla ECU sul bus
  - MMC(Multimedia Card) usato per memorizzare i file temporanei oppure le impostazioni del microcontrollore(vulnerabile ad attacchi di privilege escalation)

Punti vulnerabili

Entry Point	ECU	Bus
› RKE	› RFHM	› CAN C
› TPMS	› RFHM	› CAN C
› Bluetooth	› Radio	› CAN C - CAN IHS
› FM/AM/XM	› Radio	› CAN C - CAN IHS
› Cellular	› Radio	› CAN C - CAN IHS
› Internet/Apps	› Radio	› CAN C - CAN IHS

ROOT ACCESS INFOTAINMENT

Jailbreaking del sistema infotainment

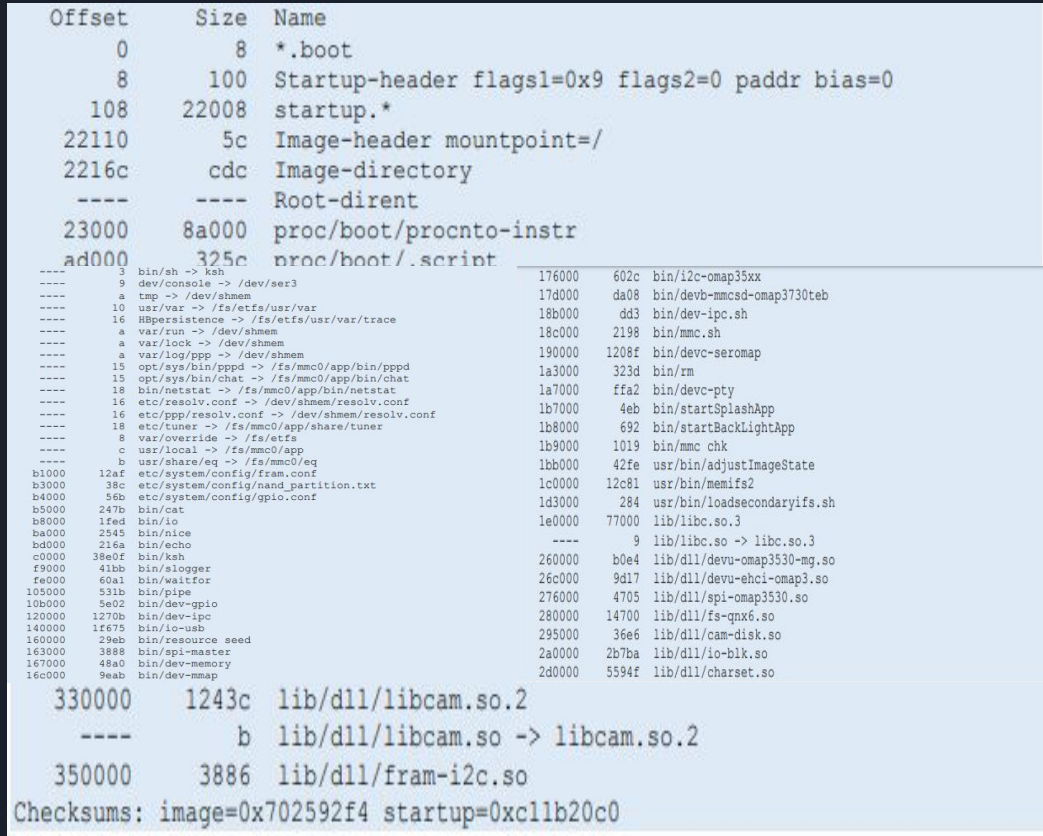
Necessario:

- emulatore QNX per virtualizzare il sistema operativo e cercare bug
- modificare il codice per rendere possibile l’utilizzo della shell
- D-Bus Python lib
- Reset della lingua e impostarla in inglese

No checksum

Il sistema operativo esegue una verifica sul checksum per confermare che il firmware sia autentico.

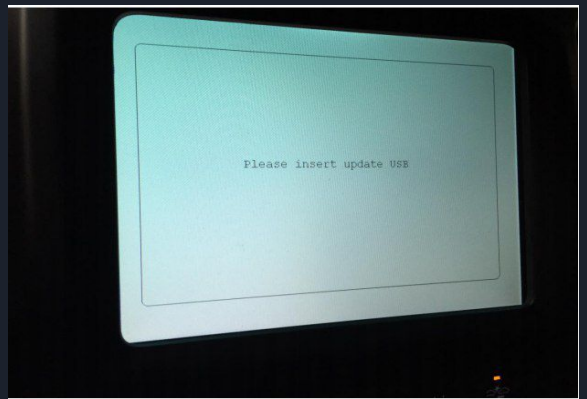
Per bypassare questo blocco dobbiamo virtualizzare il firmware con : *memifs2 -q -d /fs/usb0/usr/share/swdl.bin /* dentro l'ambiente QNX Emulator.



Invalidare il checksum

Invalidare il checksum è un operazione piuttosto semplice in questa versione del firmware poiché basta cambiare via hex editor un pezzo del codice.

Bypassare la validazione del firmware si esegue una modifica nella **all offset 128(0x80) in ‘S’ (0x53)**



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
0000h:	0C	B0	47	9C	27	F6	2D	93	3B	EC	74	5A	8D	A5	E3	98	.°Gœ'ö-";ìtZ.¥ã˜
0010h:	B6	9F	BD	F7	C8	57	6B	89	C2	C7	6B	E7	BC	6B	21	EA	¶Ÿ½÷ÈWk‰ÂÇkç¼k!ê
0020h:	96	DC	0C	D8	DD	F4	B9	9B	64	CE	8F	8B	2A	D0	8A	47	–Ü.ØÝô¹›dÎ.‹*ÐŠG
0030h:	2F	44	F7	D2	3F	45	06	4D	48	96	9E	6E	7D	7F	23	17	/D÷Ò?E.MH–žn}.#.
0040h:	49	C9	FE	D1	F1	22	A0	34	20	C6	D0	5E	DF	DD	E8	14	IÉþÑñ" 4 ÆÐ^ßÝè.
0050h:	A6	A6	2B	08	B1	47	41	03	79	18	F0	8C	3D	E2	6D	BC	¦¦+.±GA.y.ðŒ=âm¼
0060h:	49	5D	A0	CE	EB	CE	F7	C7	8A	1E	A5	68	FB	8E	3A	98	I] ÎëÎ÷ÇŠ.¥hûŽ:˜
0070h:	78	FE	40	EA	10	4D	38	30	07	5A	BC	D4	E8	B9	1D	34	xþ@ê.M80.Z¼Ôè¹.4
0080h:	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	S...............
0090h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00A0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00B0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00C0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00D0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00E0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00F0h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0100h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
0110h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....

WIFI Hack

- Crittografia: hard coded credentials e password generate con la data e ora odierna
- Il chip del wifi vulnerabile a DFU downgrade
- Nessun controllo dell'input

```
# netstat -n | grep LISTEN
tcp        0      0  *.6010                 *.*                    LISTEN
tcp        0      0  *.2011                 *.*                    LISTEN
tcp        0      0  *.6020                 *.*                    LISTEN
tcp        0      0  *.2021                 *.*                    LISTEN
tcp        0      0  127.0.0.1.3128         *.*                    LISTEN
tcp        0      0  *.51500                *.*                    LISTEN
tcp        0      0  *.65200                *.*                    LISTEN
tcp        0      0  *.4400                 *.*                    LISTEN
tcp        0      0  *.6667                 *.*                    LISTEN
```

Bus Daemon Process

La porta 6667 è un IP D-Bus che permette alle applicazioni di comunicare tra loro.
In informatica sono noti come IPC e RPC che sono due tecniche standard di comunicazione e invocazione comandi tra processi.
E importante menzionare il system bus daemon e session bus che sono i più importanti.Tra questi c'è un processo che è fondamentale NavTrailService.

Tool per scripting: D-Bus library Python

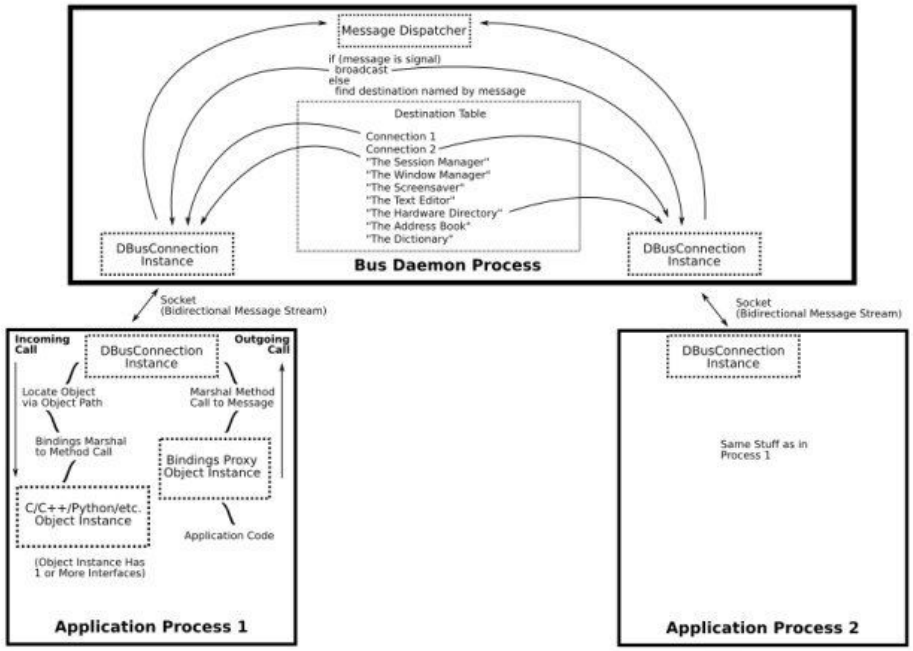


Foto: http://dbus.freedesktop.org/doc/diagram.png

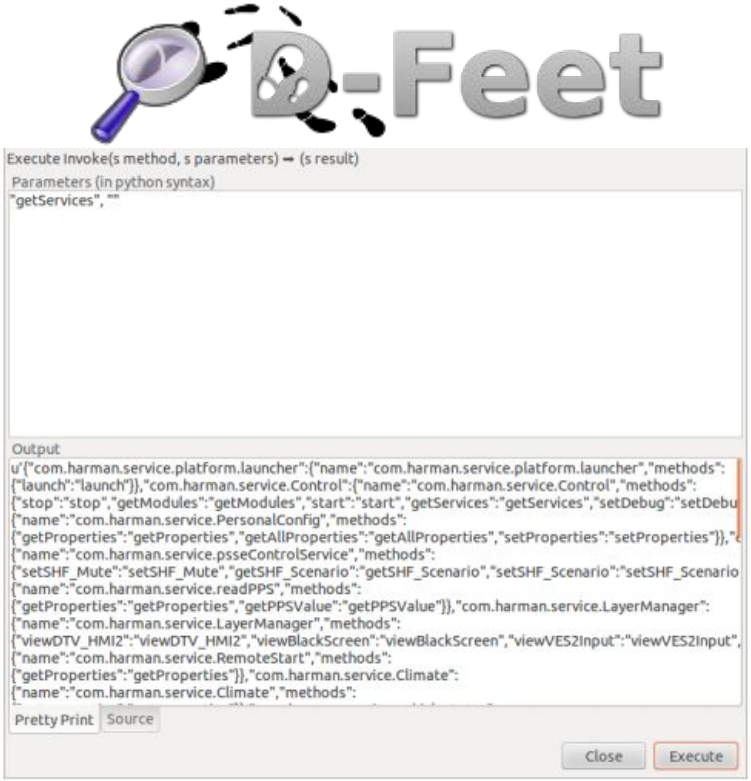
```
function methods.rmTrack(params, context)
  return {
    result = os.execute("rm \"" .. trail_path_saved .. params.filename .. "\"")
  }
end
```

```
$ telnet 192.168.5.1 6667

Trying 192.168.5.1...

Connected to 192.168.5.1.

Escape character is '^]'.

a

ERROR "Unknown command"
```

Dfeet

Basta un semplice comando su Dfeet per avere tutti i servizi attivi sul CAN BUS e tutti i dettagli su come accedere alle loro informazioni.
Le altre applicazioni vengono eseguite sempre sulla porta 6667 e attraverso metodo TCP, però per semplicità mi sono litato al caso base.Il resto è test e studio

https://github.com/GNOME/d-feet



Prossimo passo?

HVAC(*Heating, Ventilation and Air Conditioning*)

Cosa possiamo controllare?

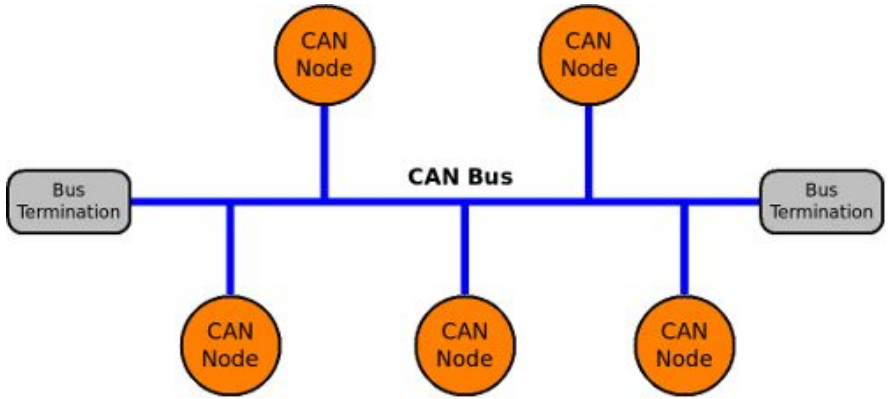
- Aria condizionata
- Radio Volume
- Radio FM
- Bottoni Radio e Riscaldamento

Ritorno al passato

Il CAN BUS permette la trasmissione di 4 tipologie di frame.Ogni tipo di frame è unico e con una struttura ben precisa.

- **Data frame**: frame contenente i dati che il nodo trasmette.
- **Remote frame**: frame che richiede la trasmissione di un determinato identificatore.
- **Error frame**: frame trasmesso da un qualsiasi nodo che ha rilevato un errore.
- **Overload frame**: frame che introduce un ritardo fra data frame e/o remote frame.

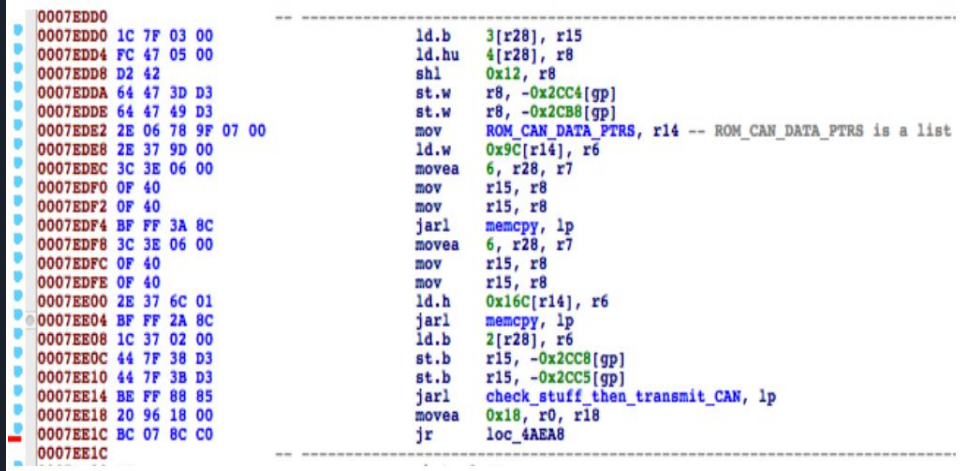
Sul CAN BUS si possono trasmettere fino a 2^{29} = 536 870 912.

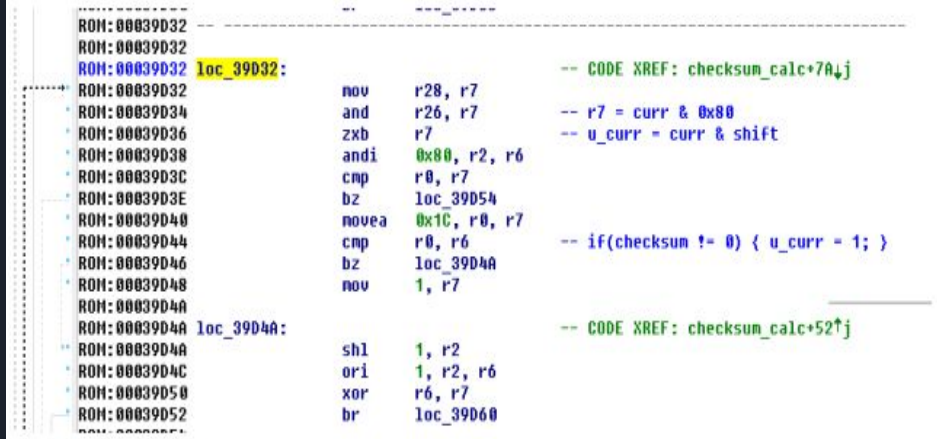


Field name	Length (bits)	Purpose
Start-of-frame	1	Denotes the start of frame transmission
Identifier (green)	11	A (unique) identifier which also represents the message priority
Remote transmission request (RTR) (blue)	1	Must be dominant (0) for data frames and recessive (1) for remote request frames (see Remote Frame, below)
Identifier extension bit (IDE)	1	Must be dominant (0) for base frame format with 11-bit identifiers
Reserved bit (r0)	1	Reserved bit. Must be dominant (0), but accepted as either dominant or recessive.
Data length code (DLC) (yellow)	4	Number of bytes of data (0–8 bytes)[a]
Data field (red)	0–64 (0-8 bytes)	Data to be transmitted (length in bytes dictated by DLC field)
CRC	15	Cyclic redundancy check
CRC delimiter	1	Must be recessive (1)
ACK slot	1	Transmitter sends recessive (1) and any receiver can assert a dominant (0)
ACK delimiter	1	Must be recessive (1)
End-of-frame (EOF)	7	Must be recessive (1)

Inviare messaggi sul CAN BUS

Per riuscire a inviare messaggi sul CAN dobbiamo sapere su quale canale comunica i dati……ne ha di più però quello più frequente è il 7.Il dispatcher fa da ponte tra kernel-mode e user-mode e quindi la funzione che chiamerà sarà diversa...in questo caso corrisponde alla location 0x7EE1C dove jr chiama un address space 4AEA8.



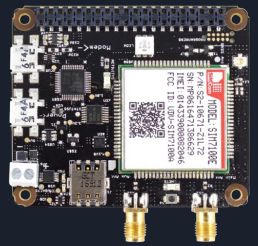


Final Hack

Cosa possiamo controllare?

- Motore
- Freni
- Sterzo
- Chiusura
- Indicatori direzione

Hacking 3G/4G



```
# ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33192
        inet 127.0.0.1 netmask 0xff000000
pflog0: flags=100<PROMISC> mtu 33192
uap0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 30:14:4a:ee:a6:f8
        media: <unknown type> autoselect
        inet 192.168.5.1 netmask 0xffffff00 broadcast 192.168.5.255
ppp0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1472
        inet 21.28.103.144 -> 68.28.89.85 netmask 0xff000000
```

192.168.5.1 = l'indirizzo gateway per i dispositivi che si connettono al hotspot wifi
68.28.89.85= Indirizzo che serve nel caso la macchina si voglia connettere con qualcuno.La porta 6667 non è aperta quindi è poco utile.
21.28.103.144 = l'indirizzo effetivo con cui i pacchetti escono dalla macchina tramite il NAT.funziona però soltanto con i provider di rete abilitati.Questi sono fissi perché vengono assegnati dal provider della rete(in Italia difficile trovare macchine con 3G/4G),nonostante l'accensione e lo spegnimento della macchina.

Abbiamo finito

1. Controllo delle luci (tutte)
2. RPMS
3. Kill engine
4. No brakes
5. Steering

Domande?