FEUERLABS

Connected Device Development By Contract

HVAC control through Exosense

by Ulf Wiger, Co-Founder, Feuerlabs Inc

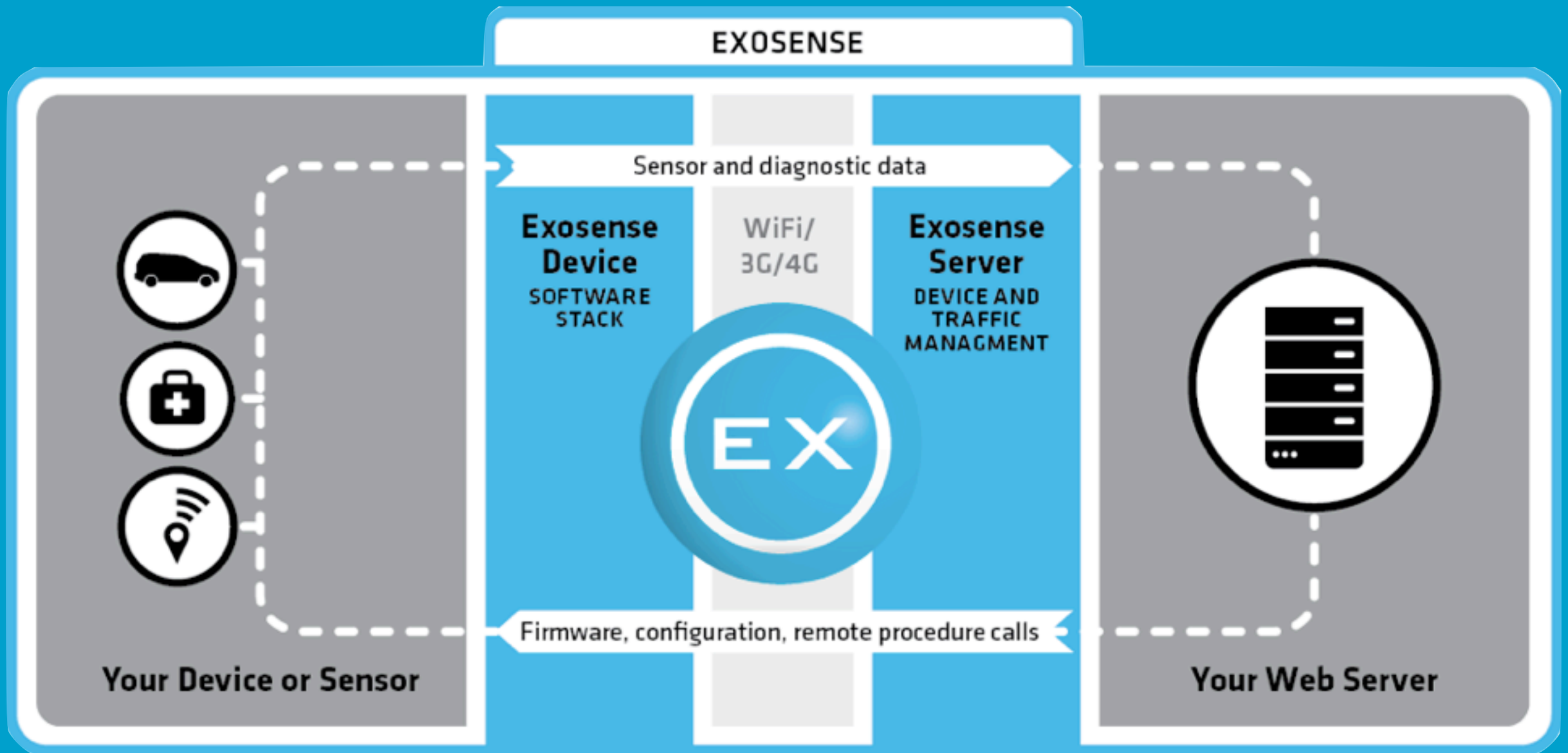
FEUERLABS

Mission

- Connected Device market set to explode
- Device management needs rethinking
  - move away from proprietary monoliths
  - should be available as a service
  - should appeal to discriminating programmers
  - must be robust and scalable

FEUERLABS

Feuerlabs Exosense®

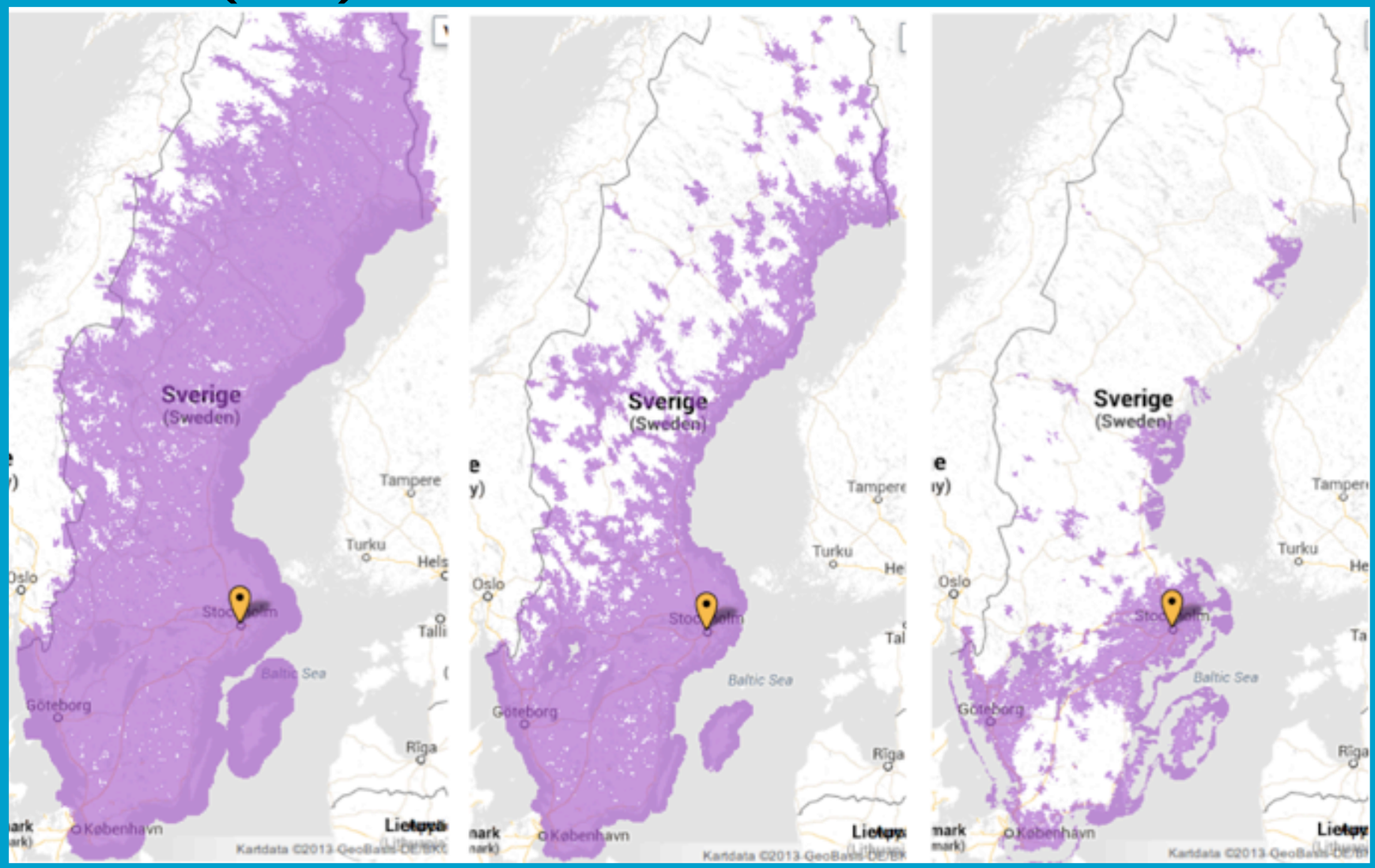


- Carrier-Grade Device Connectivity Software
- Support Rapid Prototyping, lifecycle, scalability

FEUERLABS

Key Challenges: Coverage

2G (SMS) **3G** **4G**

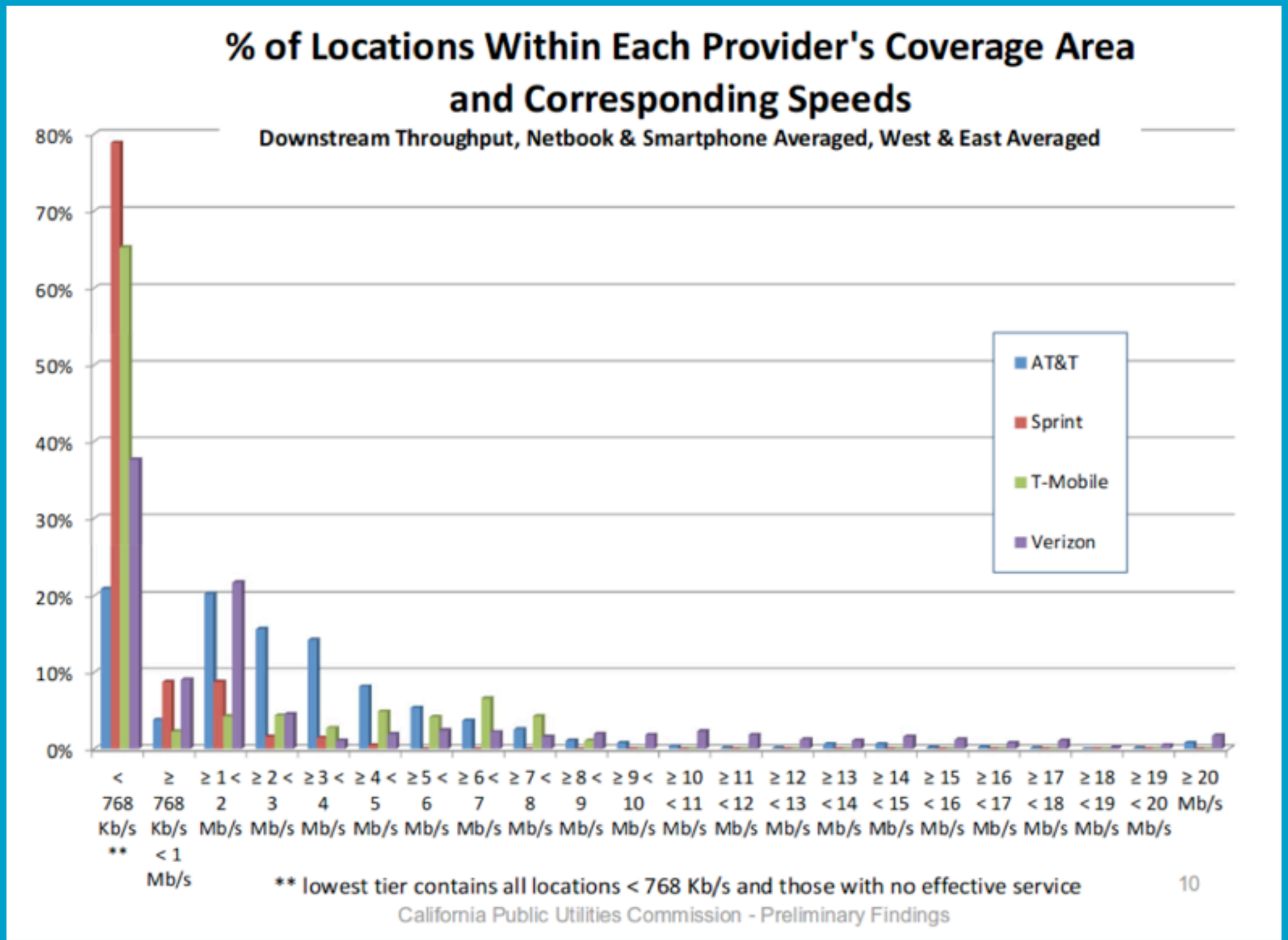


Telia coverage for Sweden, Aug 2013

- Higher speed–Less coverage

FEUERLABS

Real Life – Barely 3G



FEUERLABS

Exosense Approach

- Request store-and-forward with persistent prioritized queueing
- Push notifications (e.g. SMS) when device is offline
- Communication layer agnostic
- Scheduling of operations

FEUERLABS

Challenge: Lead Time

- Device Management frameworks are usually either extremely bloated, or extremely limited
- Exosense is both

FEUERLABS

Challenge: Lead Time

- Device Management frameworks are usually either extremely bloated, or extremely limited
- Claim: Exosense is neither

FEUERLABS

Challenge: Lead Time

- Goal: to support working end-to-end prototypes within a few working days
  - First 10 (non-commercial) device sessions free
- Plan to provide hosted device mgmt service
  - Continue supporting through the life cycle

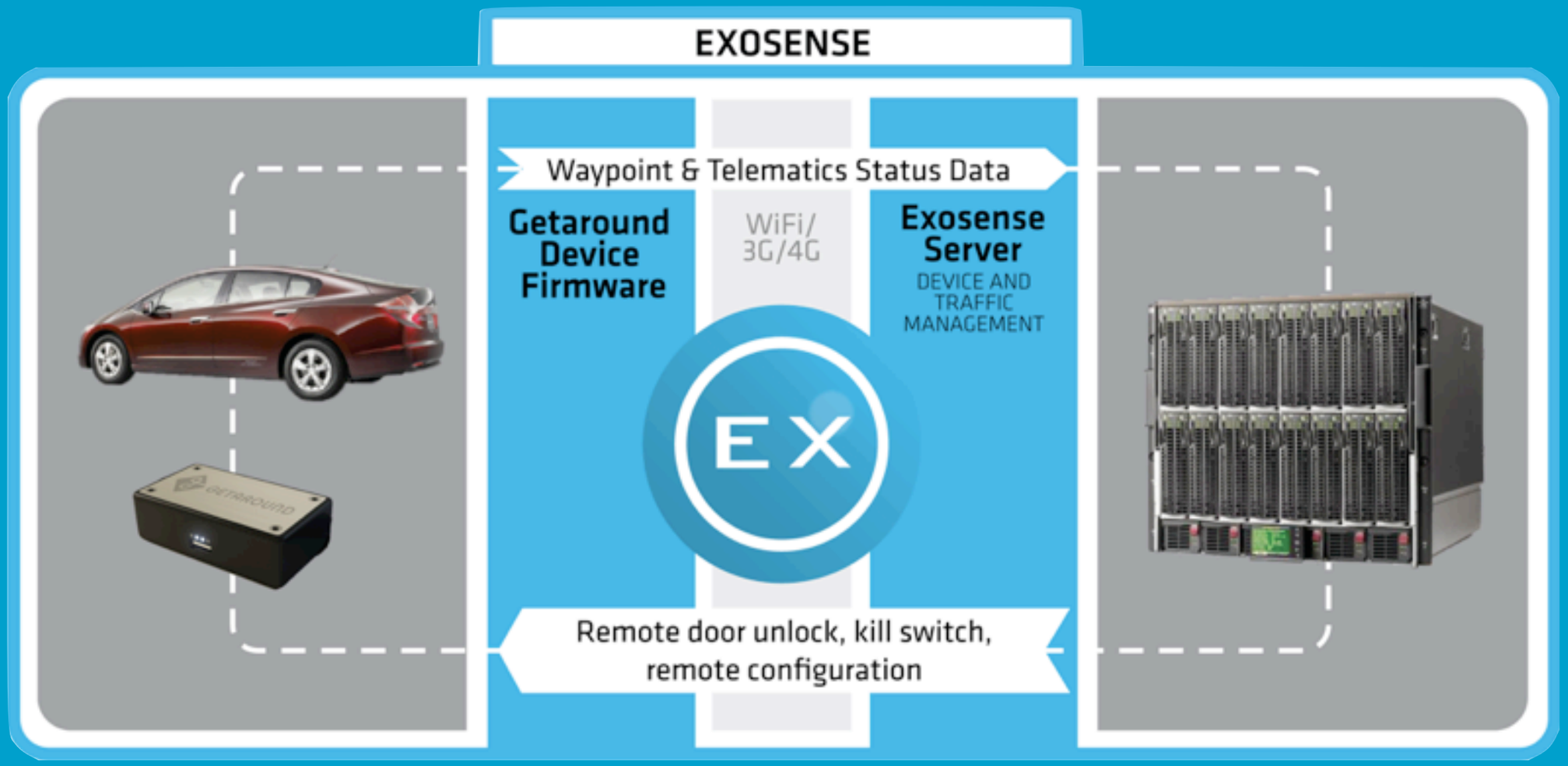
FEUERLABS

Challenge: Lifecycle

- Handle simultaneous different device versions
- **Programming by contract**:
  - RPC specifications tied to device type + pkg
  - Validation of each message + response
  - Device can trust that requests are valid
  - YANG (RFC 6020) used as spec language

FEUERLABS

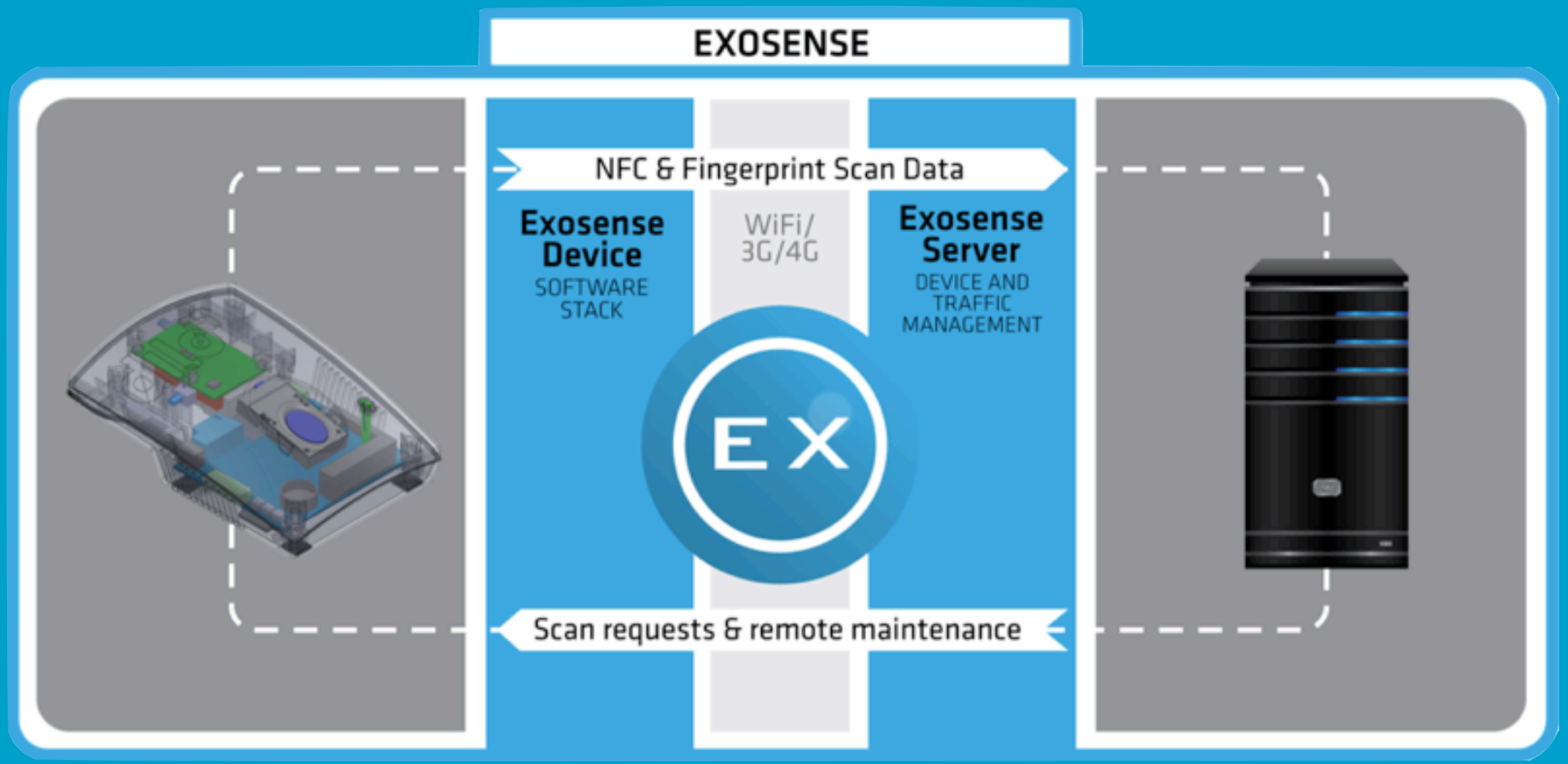
Use Case: Peer-to-Peer Car Sharing



- Security features prevent unauthorized access
- Integrated GPS
- Fast deployment

FEUERLABS

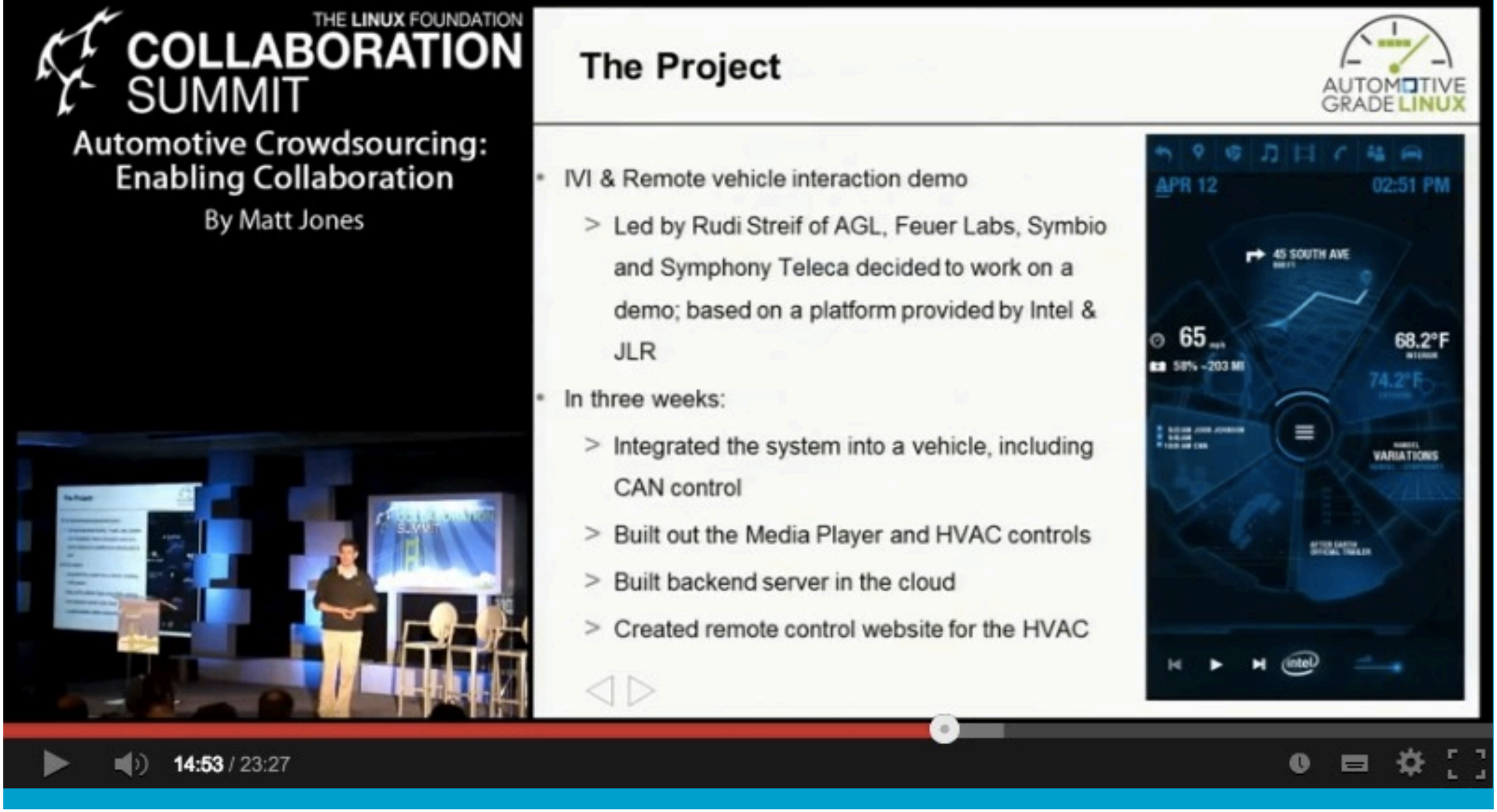
Single-card Health Services Access



- Connectivity to a Government services backend
- Designed for 200K+ users within 4 months of project deployment
- Has been extended to include smartphone-based NFC terminals, kiosks

AGL/JLR/Feuerlabs/Symbio prototype

http://www.youtube.com/watch?v=b_jfa7SFRxI



FEUERLABS

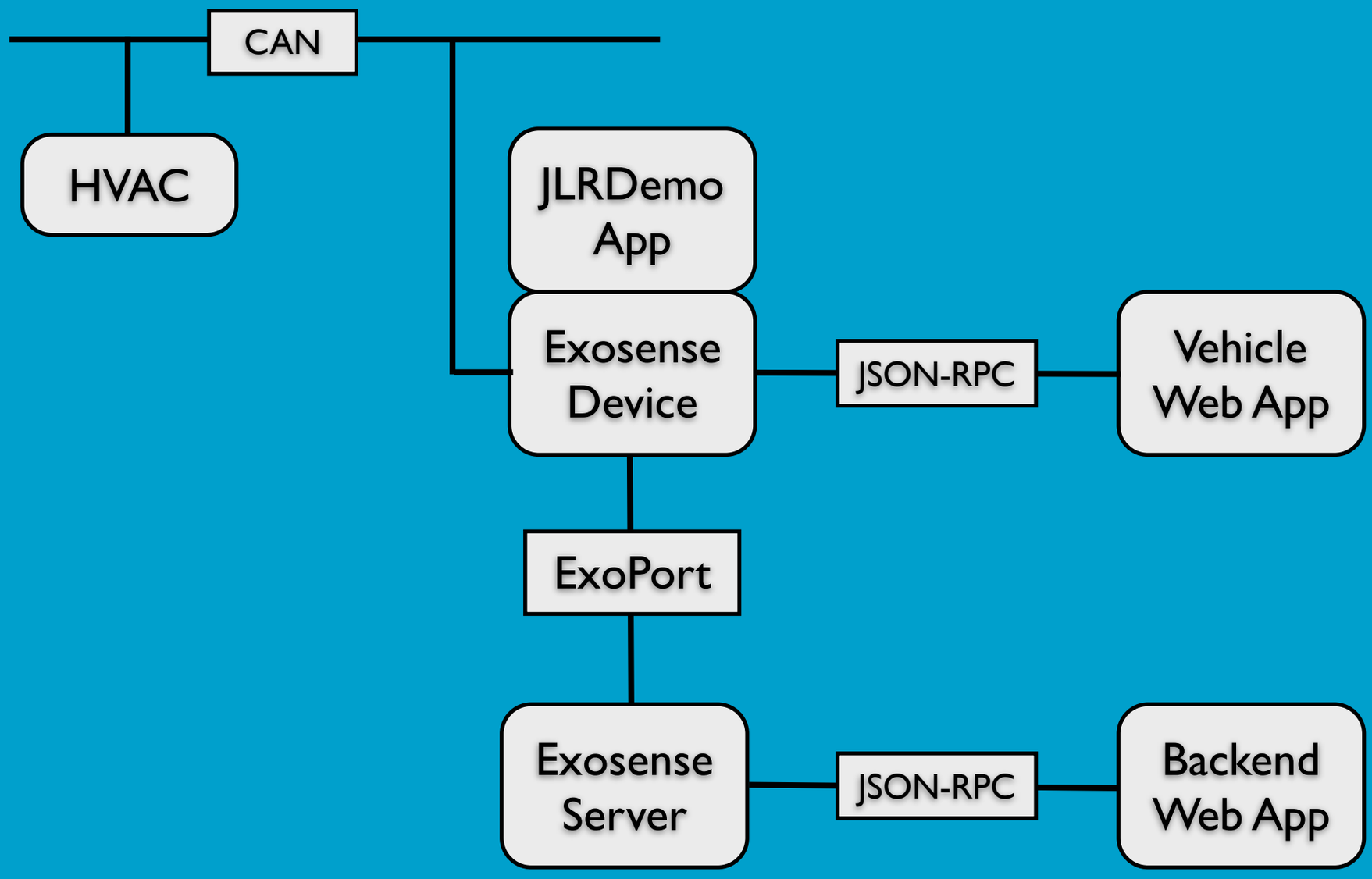
Prototype Walkthrough

- HVAC (climate system) control prototype
- Running on AGL Intel Reference Hardware
  - Intel Atom E640 up to 1.0GHz
  - On-board DDR2 1GB (up to 2GB)
  - CAN/UART/HDMI/Eth/...
  - USD 2,000

http://automotive.linuxfoundation.org/node/47

FEUERLABS

System schematics



FEUERLABS

High-level Steps

- Create an Exosense account
- Store YANG spec(s)
- Create a device type, link to YANG spec
- Create device(s) of given device type
- Hack, hack, hack... test
- Demo

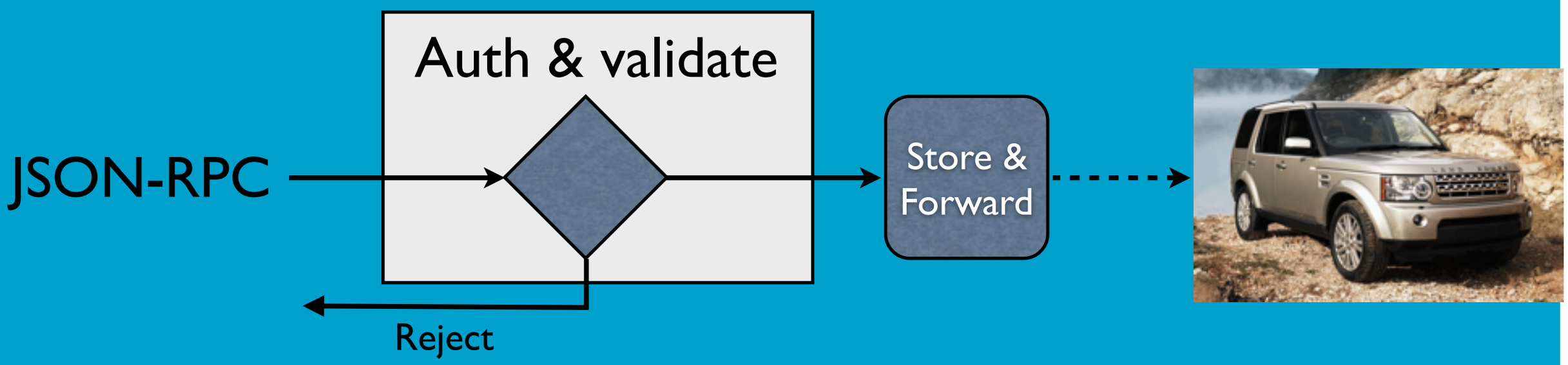
Specification

```
module jlrdemo {

    namespace "http://feuerlabs.com/jlrdemo";
    prefix demo;

    import exosense {
        prefix exo;
    }

    description
        "JSON-RPC spec for the Jaguar Land Rover demo remote HVAC control." +
        "The protocol will be used between the Exosense Server and the web " +
        "server as well as between the Exosense Device JLR Demo Application " +
        "(JLR App) and the Webkit running the IVI UI in the vehicle.";
...
```



Specification (cont...)

```
//
// Set the HVAC fan speed
//
rpc set-fan-speed-request {
    description "...";

    input {
        description
            "Input consists of standard request elements " +
            "and a fan speed between 0 and 9.";

        uses exo:std-request;

        leaf fan-speed {
            description "Fan speed to set. 0 = Off. 9 = Max.";
            type uint32;
        }
    }

    // Output only contains status.
    output {
        description
        "Output sent in response to a set-fan-speed-request";
        uses exo:std-callback;
    }
}
```

Curl script

```
#!/bin/sh
. $HOME/.exodmrc

if [ $# != 2 ]
then
    echo "Usage: $0 device-id speed(0-10)"
    exit 255
fi
curl -u $USER_AUTH -k -X POST $URL -d @- <<EOF
{
    "jsonrpc": "2.0",
    "method": "jlrdemo:set-fan-speed-request",
    "id": "1",
    "params":
    {
      "device-id": "$1",
      "fan-speed": $2

    }
  }
EOF
```

FEUERLABS

Device-side Logic

jlrdemo_rpc.erl

```
%% JSON-RPC entry point
%% CAlled by local exo http server
handle_rpc(<<"jlrdemo">>, Method, Args, _Meta) ->
    case Method of
        <<"set-fan-speed-request">> ->
            Res = 'set-fan-speed-request_'(Args),
            exoport:rpc(
              exodm_rpc, rpc,
              [<<"jlrdemo">>, <<"set-fan-speed-request">>, Args]),
            Res;
         ...
     end.

%% Called as a result of RPC from Exosense Server
'set-fan-speed-request'(Args) ->
    'set-fan-speed-request_'(Args),
    send_http_request(<<"jlrdemo">>, <<"set-fan-speed-request">>, Args),
    ok(?COMPLETE).

'set-fan-speed-request_'(Args) ->
    case lists:keyfind('fan-speed', 1, Args) of
        {_, Value} ->
            jlrdemo_can:set_fan_speed(Value),
            ok(?COMPLETE);
        false ->
            ok(?VALUE_ERROR)
    end.
```

FEUERLABS

CAN Bus Interaction

jlrdemo_can.erl

```
-module(jlrdemo_can).
-behavior(gen_server).

...

start_link() ->
    gen_server:start_link({local, ?MODULE}, ?MODULE, [], []).

init(_) ->
    can_router:start(),
    {ok, #st{}}.

handle_call({start_can, Interface, Driver}, _From, S) ->
    can_sock:start(Interface, Driver, []),
    can_router:attach(),
    can:send(?INITIAL_CAN_FRAME),
    erlang:send_after(1000, self(), refresh_can_frame),
    {reply, ok, S#st{iface = Interface, packet = ?INITIAL_CAN_FRAME}};

handle_call({set_fan_speed, Speed}, _From,
            #st{fan_blower_speed = {Offs, Length}, packet = P} = St) ->
    << Before:Offs/binary, _:Length, After/binary >> = P
    NewP = << Before/binary, Speed:Length, After/binary >>,
    can:send(NewP),
    {reply, ok, St#st{packet = NewP};
...
```

CAN Bus Interaction

jlrdemo_can.erl

```
-module(jlrdemo_can).
-behavior(gen_server).

...

start_link() ->
    gen_server:start_link({local, ?MODULE}, ?MODULE, [], []).

init(_) ->
    can_router:start(),
    {ok, #st{}}.

handle_call({start_can, Interface, Driver}, _From, S) ->
    can_sock:start(Interface, Driver, []),
    can_router:attach(),
    can:send(?INITIAL_CAN_FRAME),
    erlang:send_after(1000, self(), refresh_can_frame),
    {reply, ok, S#st{iface = Interface, packet = ?INITIAL_CAN_FRAME}};

handle_call({set_fan_speed, Speed}, _From,
            #st{fan_blower_speed = {Offs, Length}, packet = P} = St) ->
    << Before:Offs/binary, _:Length, After/binary >> = P
    NewP = << Before/binary, Speed:Length, After/binary >>,
    can:send(NewP),
    {reply, ok, St#st{packet = NewP};
...
```

That's it!

Exosense Device – Released Features

Component	Description
gsms	SMS send/receive (3GPP 07.05)
netlink	Network availability monitor
pppd_mgr	PPP controller for link management
gsm_mux	SMS / Data multiplexing (3GPP 07.10)
exoport	Extensible RPC protocol (with SSL support)
gpio	GPIO pin manager
inpevt	Input device management (touch screen, keyboard, etc)
nmea_0183	GPS manager
kvdb	ACID tree-structured DBMS

FEUERLABS

Exosense Device – In the works

Component	Description
epx	Framebuffer graphics and GUI components
exoprio	Rule driven delivery of exoport messages over multiple data links. Supports data budgets, prioritizing, scheduling, link management, etc.
cfg_mgr	Unified configuration manager for Exosense and other device-local sub-systems.
pkg_mgr	Enhanced FOTA with dependency graphs, binary diffs, rollback, and in-service upgrades.
exofile	Chunked file transfer over exoport.

FEUERLABS

Example: Sending SMS

```
1> gsms:send([{addr, “+15551231234"}], "Hello World.").
{ok,#Ref<0.0.0.73>}
```

Receiving SMS

```
2> gsms:subscribe([{anumber, "+5551231234"}]).
{ok,#Ref<0.0.0.57>}
3> flush().
Shell got {gsms,#Ref<0.0.0.57>,
           {gsms_deliver_pdu,
            {gsms_addr,international,"+12063130024"},
            false,false,false,0,0,true,
            {gsms_addr,international,"+15551231234"},0,
            {gsms_dcs,message,uncompressed,
             default,alert,store,inactive,other},
            {{{2013,8,15},{11,38,41}},27.0},[],3,“Hello!"}}
ok
```

FEUERLABS

Remote Door Unlock

```
1    -module(tmdemo).
2    -include_lib("can/include/can.hrl").
3    -export([start/0, setup/0]).
4
5    start() ->
6        application:start(uart),
7        gsms:start(),
8        pppd_mgr:start(),
9        can_router:start(),
10       spawn(?MODULE, setup, []).
11
12   setup() ->
13       {_Result, Ref} = gsms:subscribe([]),
14       loop(Ref).
15
16   loop(Ref) ->
17       receive
18           { gsms, Ref, { _, _, _, _, _, _, _, _, _, _, _, _, _, _, "unlock" }} ->
19               CanFrame = #can_frame {
20                 id = 16#AB1,
21                 data = << 16#FF >>,
22                 len = 1, intf = 0, ts = 0
23                },
24               can:send(CanFrame)
25       end,
26       loop(Ref).
```

FEUERLABS

Questions

Contact:
Ulf Wiger
Email: ulf.wiger@feuerlabs.com
Phone: +46 76 196 6190