Connected Vehicle

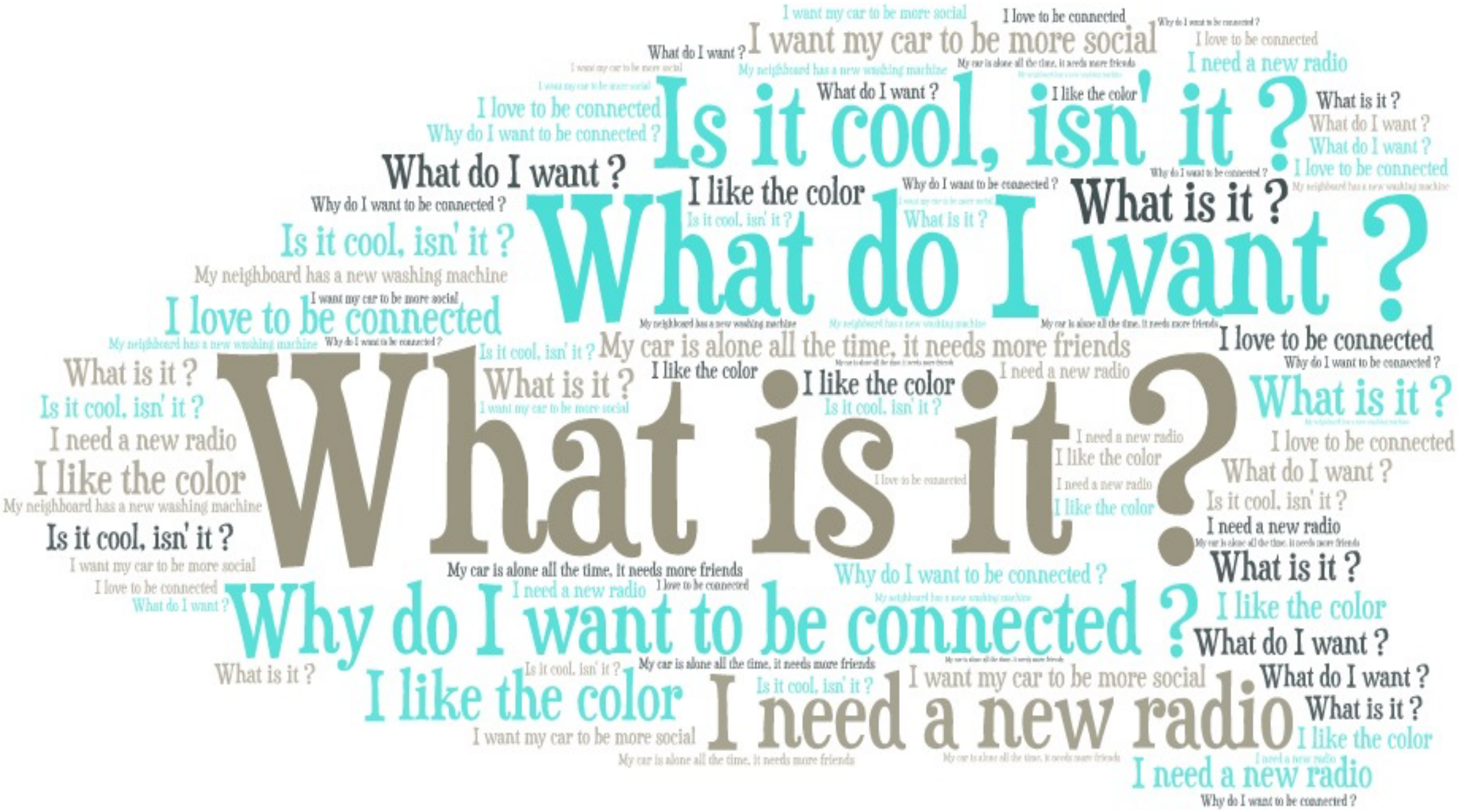
Fabrizio Manfredi Furuholmen

manfred.furuholmen@gmail.com

www.cios.io

Agenda

- Overview
- Hardware
- Software
- Build your own solution
- Demo (Risk)
- Q&A


What is it ?
What do I want ?
Is it cool, isn' it ?
Why do I want to be connected ?
I need a new radio
I like the color
I love to be connected
I want my car to be more social
My car is alone all the time, it needs more friends
My neighboard has a new washing machine

Connected Car

- A **connected car** is a car that is equipped with Internet access, and usually also …
- Today's car has the computing power of 20 personal computers, features about 100 million lines of programming code, and processes up to 25 gigabytes of data an hour.

The new Klondike

There is a estimation that while the total cost of ownership of vehicles will remain **stable** for consumers, the **dramatic increase** in vehicle connectivity will increase the value of the global market for **connectivity components** and services to **€170 billion** by 2020 from just €30 billion today.

Intel acquires Mobileye: $15.3 billion

Cisco bets on Jasper: $1.4 Billion

Samsung Electronics has completed the **acquisition** of **Harman** International Industries in a deal worth $8 billion…

Verizon gobbling up connected car tech: $2.4 billion +

Qualcomm buys NXP: $46 billion…

Automotive giant Lear will pay $320M to acquire Seattle-area connected car startup Xevo..

Volkswagen buys Volvo's connected car service for $122 million deal, which gives the German carmaker 75.1 percent of the connected driving …

What Can I do ?

- Mobility Management

  functions that allow the driver to reach a destination quickly, safely, and in a cost-efficient manner (e.g.: Current traffic information, Parking lot or garage assistance, Optimised fuel consumption)

- Commerce

  functions enabling users to purchase good or services while on-the-go (e.g., fuel, food & beverage, parking, tolls, video content)

- Vehicle management

  functions that aid the driver in reducing operating costs and improving ease of use (e.g., vehicle condition and service reminders, remote operation, transfer of usage data)

- Breakdown prevention

  connected to a breakdown service, with a back end algorithm predicting breakdowns and an outbound service intervening via phone, SMS or push notification

- Safety

  functions that warn the driver of external hazards and internal responses of the vehicle to hazards (e.g., vehicle condition and service reminders, remote operation, transfer of usage data)

- Entertainment

  functions involving the entertainment of the driver and passengers (e.g., smartphone interface, WLAN hotspot, music, video, Internet, social media, mobile office)

- Driver assistance:

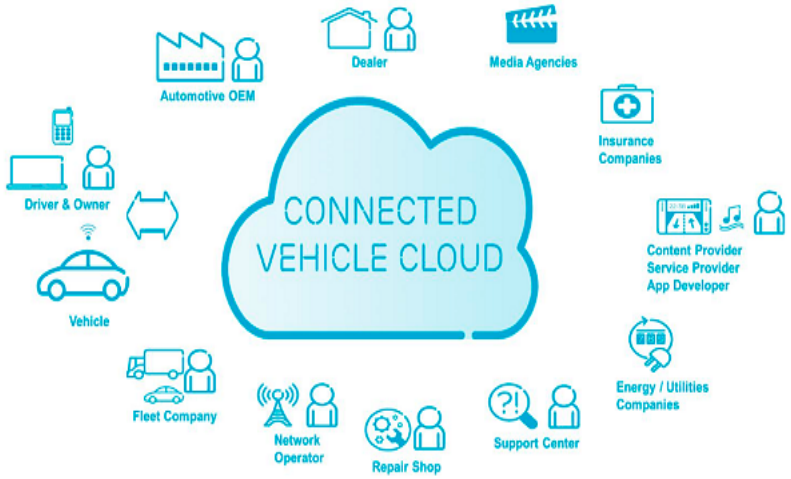
  functions involving partially or fully automatic driving (e.g., operational assistance or autopilot in heavy traffic, in parking, or on highways)

- Well-being

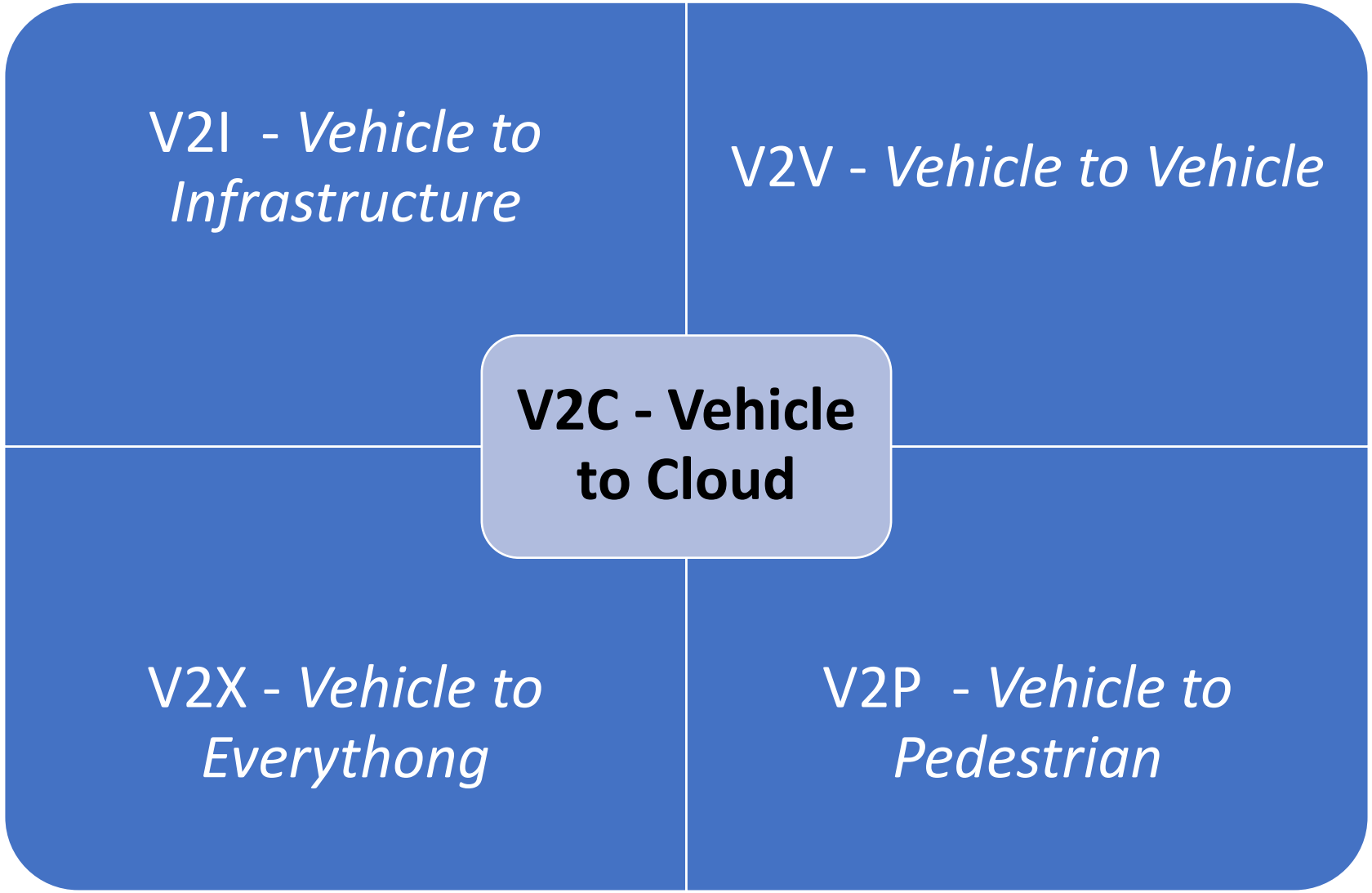
  functions involving the driver's comfort and ability and fitness to drive (e.g., fatigue detection, automatic environment adjustments to keep drivers alert, medical assistance)

- Data

  Collect data for R&D or for selling

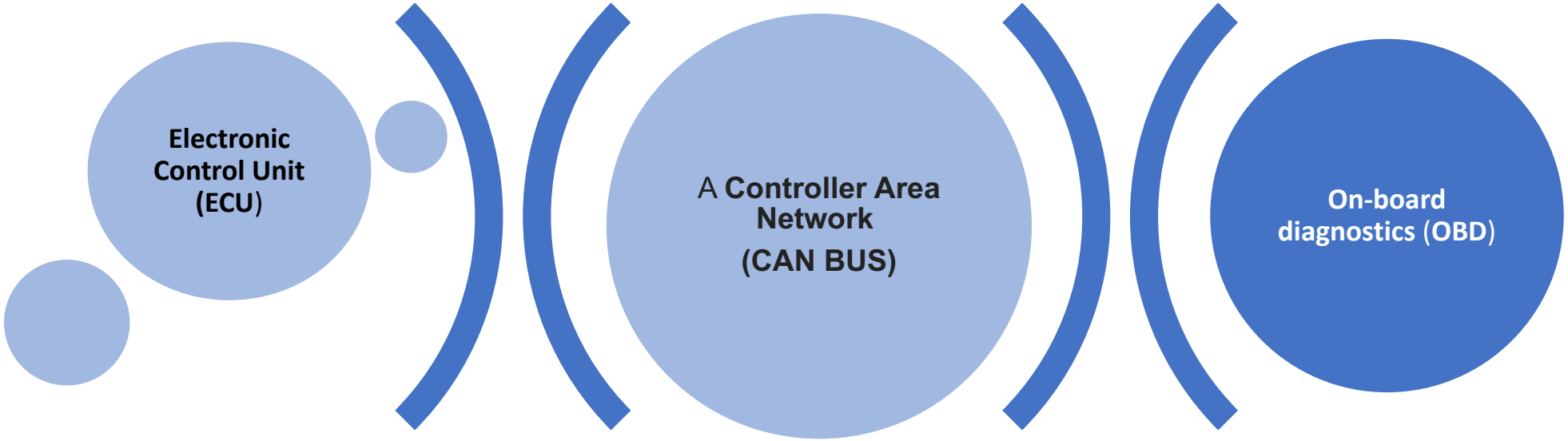


Five Categories



How can I talk with my Vehicle?

CAN BUS is your Friend



An Electronic Control Unit (ECU)

is any embedded system in automotive electronics that controls one or more of the electrical systems or subsystems in a vehicle. Some modern motor vehicles have up to 80 ECUs.

A **Controller Area Network (CAN bus)**

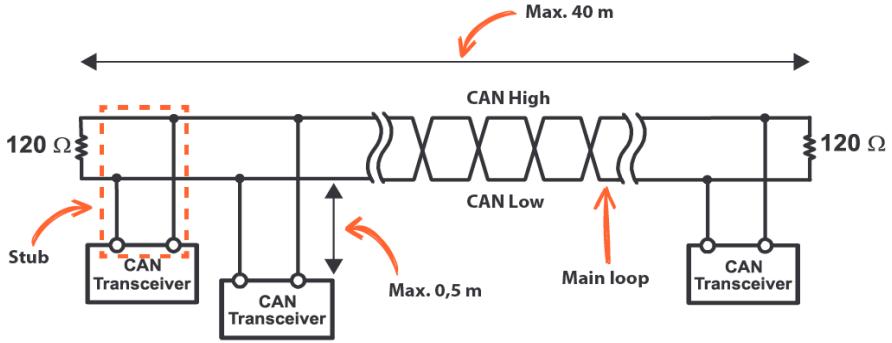
is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other in applications without a host computer.

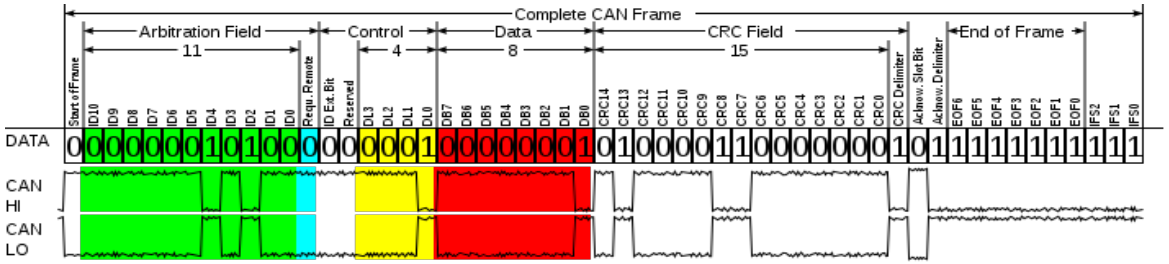
On-board diagnostics (OBD)

is an automotive term referring to a vehicle's self-diagnostic and reporting capability. OBD systems give the vehicle owner or repair technician access to the status of the various vehicle subsystems

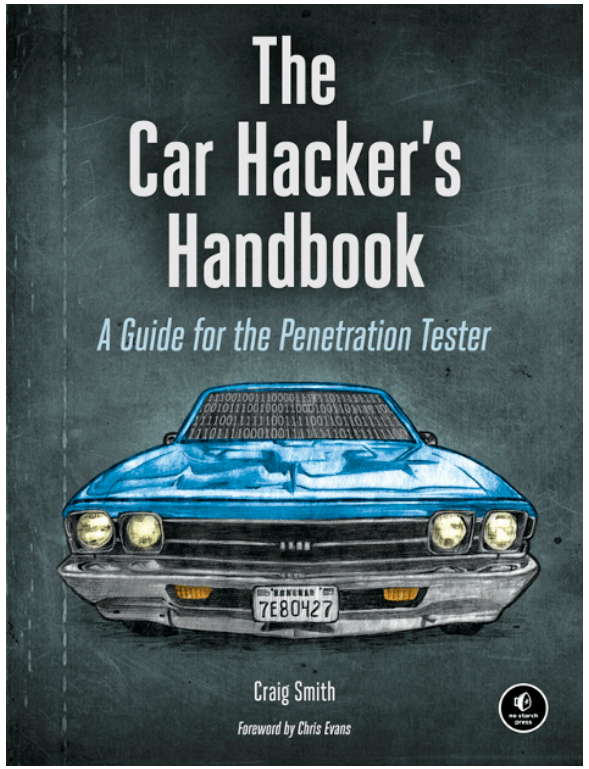
Can Bus is not so Friendly

CAN is a multi-master serial bus standard for connecting Electronic Control Units [ECUs] also known as nodes. Two or more nodes are required on the CAN network to communicate. The complexity of the node can range from a simple I/O device up to an embedded computer with a CAN interface and sophisticated software. The node may also be a gateway allowing a general purpose computer (such as a laptop) to communicate over a USB or Ethernet port to the devices on a CAN network. Everyone define own protocol.





https://en.wikipedia.org/wiki/CAN_bus



http://opengarages.org/handbook/ebook/

Cars has at least 3 CAN Bus:

- Comfort
- Body
- System

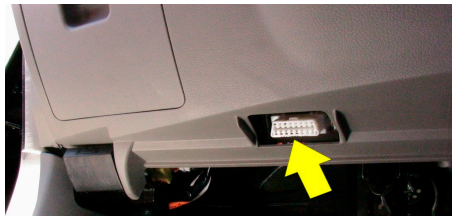
OBD is your friend

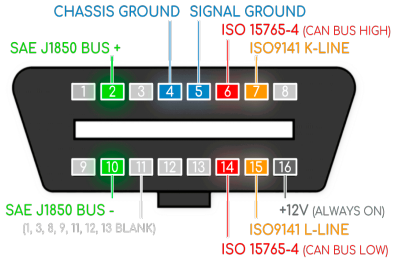
OBD-II
provides access to data from the electronic control unit (ECU) and offers a valuable source of information when troubleshooting problems inside a vehicle.

OBD-II PIDs
the various parameters that are available are addressed by "parameter identification numbers" (parameter IDs or PIDs) which are defined in J1979. For a list of basic PIDs, their definitions, and the formula to convert raw OBD-II output to meaningful diagnostic units,.
Manufacturers are not required to implement all PIDs listed in J1979 and they are allowed to include proprietary PIDs that are not listed.

DTCs
The PID request and data retrieval system gives access to real time performance data as well as flagged diagnostic trouble codes (DTCs)

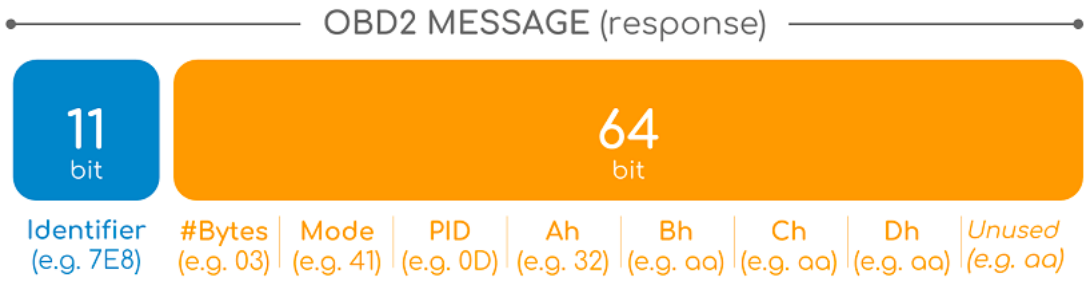
1996: OBD2 was made mandatory across USA for **cars** and **light trucks**
2001: Required in EU for **gasoline cars**
2003: Required in EU also for **diesel cars** (EOBD)
2005: OBD2 was required in US for **medium duty vehicles**
2008: US cars were required to use ISO 15765-4 (a variant of CAN) as basis for OBD2
2010: Finally, OBD2 was required in US **heavy duty vehicles**





The 1988 BMW 8-Series,
the first car to adopt the CAN bus

OBD Messages

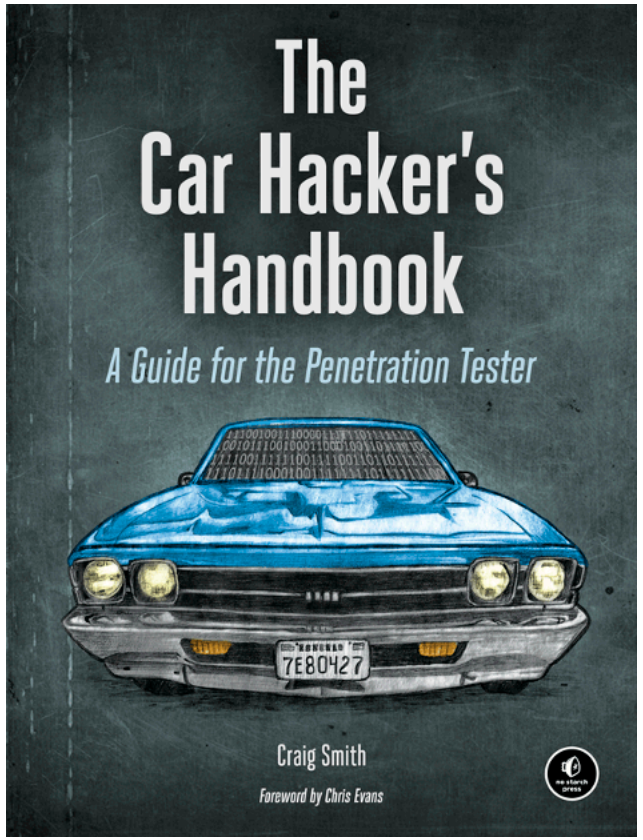


Request: 7DF 02 01 0D 55 55 55 55 55
Response: 7E8 03 41 0D 32 aa aa aa aa

Speed 50km/h

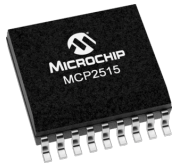
- Standard code
  - Wikipedia OBD-II PIDs
  - http://www.obdtester.com/carinfo/
- Not Standard Code
  - candump
  - cansniffer

  And Good Luck



http://opengarages.org/handbook/ebook/

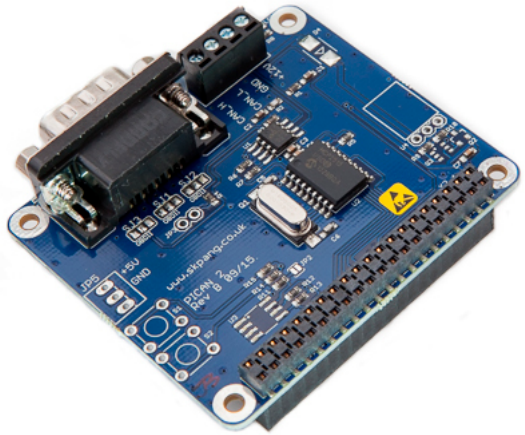
CAN BUS/OBD HW



MCP2515-
MCP2515…

MKR CAN
Arduino

Library



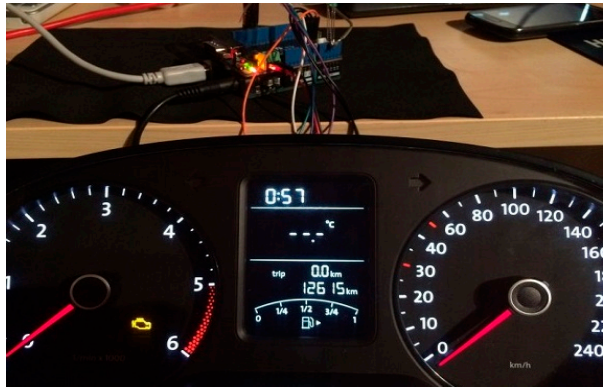
PiCan2
(Rasberry
PI)

/dev/can

1. ssh

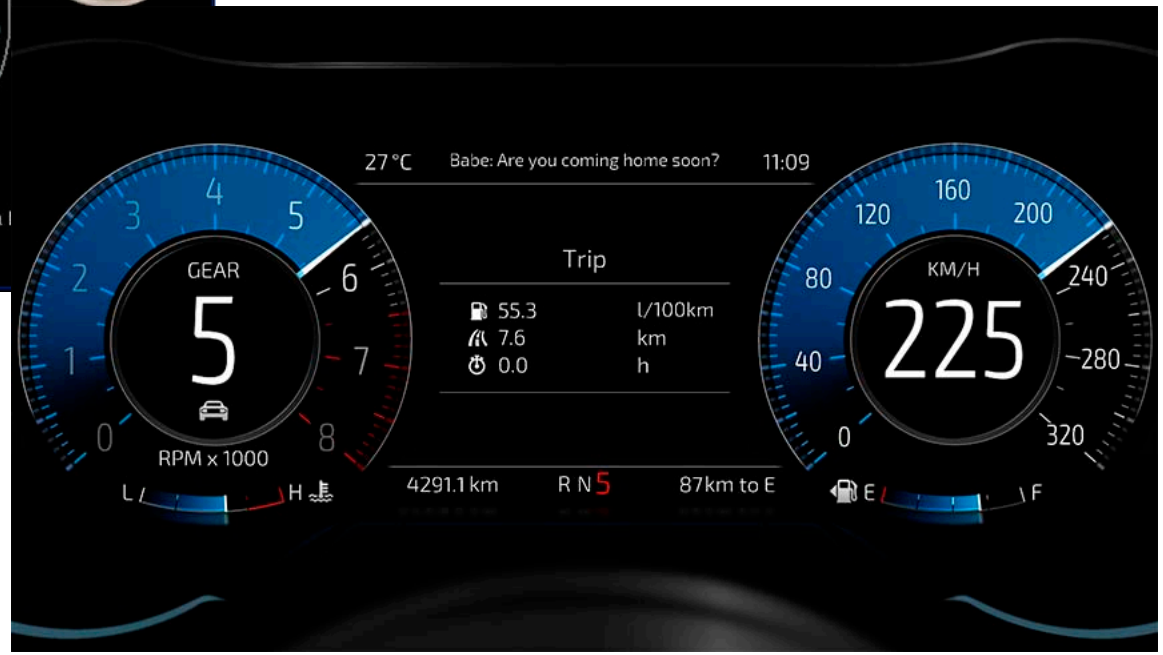
Count	Time	dt	ID	DLC	Data	Func code	Node ID
3	432.409822	0.606576	0x000	2	01 02	NMT	
131	464.048829	0.250010	0x080	0		SYNC	
1	431.803900	0.000000	0x081	8	00 10 01 49 45 00 00 00	EMCY	0x01
1	431.562248	0.000000	0x101	1	00		
1	431.547911	0.000000	0x102	1	01		
1	431.548065	0.000000	0x103	3	00 00 00		
307	464.136379	0.105426	0x104	8	AE D7 F1 01 33 00 00 00		
309	464.134964	0.104725	0x105	8	BE 6D 6D 42 00 CC 91 41		
306	464.136937	0.104687	0x110	8	4F 38 50 3C DA EA D6 BC		
309	464.140567	0.105522	0x119	8	00 00 00 00 5C 47 D8 41		
1	431.552476	0.000000	0x11E	2	7B 00		
306	464.138374	0.105435	0x11F	8	10 87 82 3A BA F8 DA BA		
30	463.187999	1.057955	0x120	8	00 00 00 00 3D 2D 76 3A		
30	463.188937	1.057968	0x121	8	D0 ED 22 3C D7 28 D2 BC		
29	464.240023	1.058092	0x140	8	C1 A0 E7 A0 D9 A0 60 00		
29	463.182889	1.057991	0x141	6	04 0A 0G 0C 79 0F		
29	463.183970	1.057959	0x142	8	17 04 01 20 01 07 60 00		
29	463.184933	1.057988	0x143	8	F8 7D 05 00 79 26 9E 04		
29	463.185941	1.057801	0x144	8	01 C0 0F 2C 00 00 00 0F		
30	463.186877	1.057934	0x145	5	73 0F 2B 00 00		
1	431.548757	0.000000	0x160	8	70 11 01 00 10 27 00 00		
1	431.548937	0.000000	0x161	8	A8 61 00 00 96 00 E8 03		
1	431.549193	0.000000	0x162	8	00 00 00 00 00 00 00 00		
1	431.549537	0.000000	0x163	8	00 00 00 00 E8 03 E8 03		
1	431.549882	0.000000	0x164	8	2C 01 00 00 50 00 00 00		
1	431.550087	0.000000	0x165	8	C8 00 00 00 96 00 E8 03		
1	431.550321	0.000000	0x166	8	80 4F 12 00 C0 27 09 00		
1	431.550565	0.000000	0x167	8	50 C3 00 00 F4 01 64 00		
129	464.052292	0.250228	0x181	8	00 00 00 00 00 00 00 00	TPDO1	0x01
131	464.049037	0.250008	0x201	8	00 00 00 00 00 00 00 00	RPDO1	0x01
1	432.812001	0.000000	0x202	8	00 00 00 00 00 00 00 00	RPDO1	0x02
20	435.689667	2.999875	0x215	8	00 00 00 00 00 FF 00 00	RPDO1	0x15
12	461.802071	2.499475	0x281	7	1C 00 1A 00 24 00 01	TPDO2	0x01
130	464.049242	0.250007	0x301	6	00 00 00 00 00 00	RPDO2	0x01
1	432.812207	0.000000	0x302	8	00 00 00 00 00 00 00 00	RPDO2	0x02
1	431.641212	0.000000	0x315	8	00 00 00 00 00 00 00 00	RPDO2	0x15
12	461.802305	2.499478	0x381	8	50 04 49 45 74 18 00 00	TPDO3	0x01
1	432.812374	0.000000	0x402	4	00 00 00 00	RPDO3	0x02
3	435.689448	2.999782	0x415	8	3F 02 00 00 00 00 00 00	RPDO3	0x15
3	436.811541	2.499587	0x581	8	60 11 6C 01 00 00 00 00	SDO_TX	0x01
101	432.759100	0.003415	0x582	8	60 20 73 0A 00 00 00 00	SDO_TX	0x02
1	431.678278	0.000000	0x595	8	60 17 10 00 00 00 00 00	SDO_TX	0x15
3	436.802125	2.498437	0x601	8	2B 11 6C 01 00 18 00 00	SDO_RX	0x01
101	432.753440	0.003603	0x602	8	2B 20 73 0A 00 00 00 00	SDO_RX	0x02
1	431.642133	0.000000	0x615	8	2B 17 10 00 64 00 00 00	SDO_RX	0x15
326	464.191658	0.099988	0x701	1	05	HEARTBEAT	0x01
317	464.212059	0.100623	0x702	1	05	HEARTBEAT	0x02
327	464.209639	0.100834	0x715	1	05	HEARTBEAT	0x15
676	464.198860	0.049964	0x77F	1	05	HEARTBEAT	0x7F
327	464.155192	0.099955	0x0000017B	4	00 00 00 00		
340	464.154684	0.099881	0x0000097B	8	00 00 00 00 00 00 00 00		
340	464.154912	0.099941	0x00000E7B	8	0A F8 00 00 08 97 DA AF		

OBD
with Bluetooth



Android/iOS Dashboard

https://linux.softpedia.com/get/Utilities/JDash-34315.shtml



http://www.realdash.net/test/gallery.php

Now send the Data!

GPRS/WCDMA/LTE – GPS/GNSS

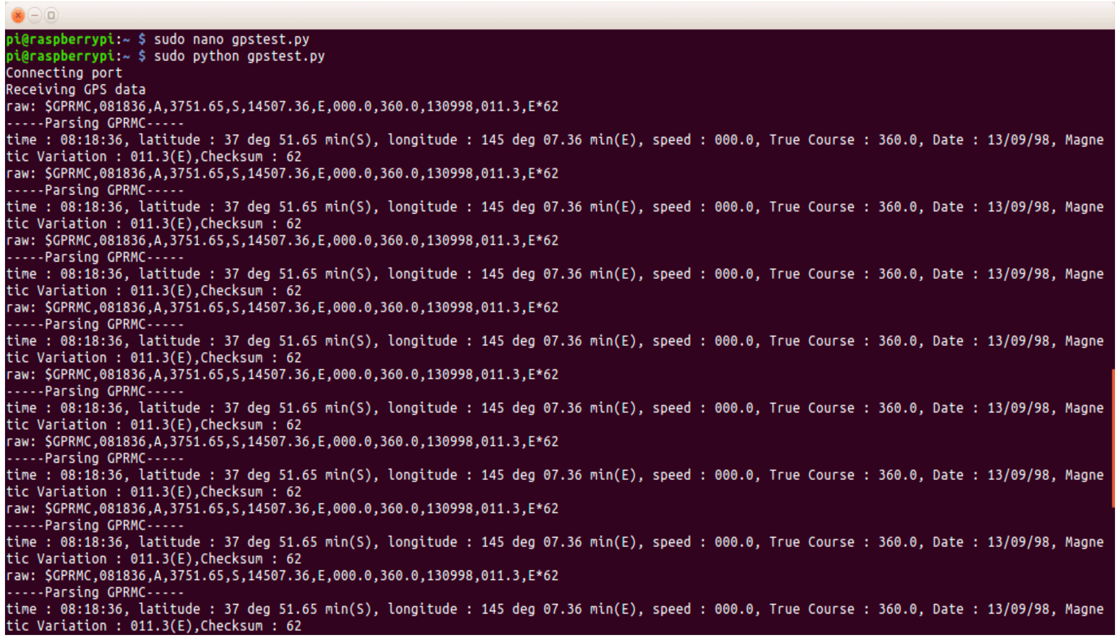
Internal unit

- A lot of ”shield” with mobile connection and GPS, but at the end there are 2 vendors:
    - SIM XXX
    - Quectel xxx

- Different mobile module, From few kb/s to Mb/s :
    - GSM
    - GPRS
    - WCDMA
    - NB-IoT
    - LTE

LTE+GPS

Nb-IoT+ GPS



```
pi@raspberrypi:~ $ sudo nano gpstest.py
pi@raspberrypi:~ $ sudo python gpstest.py
Connecting port
Receiving GPS data
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
raw: $GPRMC,081836,A,3751.65,S,14507.36,E,000.0,360.0,130998,011.3,E*62
-----Parsing GPRMC-----
time : 08:18:36, latitude : 37 deg 51.65 min(S), longitude : 145 deg 07.36 min(E), speed : 000.0, True Course : 360.0, Date : 13/09/98, Magne
tic Variation : 011.3(E),Checksum : 62
```

External unit:

- Bluetooth
- USB

Read the position from GNSS

AT Command

- All the module expose at command through serial interface, with proprietary protocol

```
void IRAM_ATTR taskGPS(void* arg)
{
#if USE_SOFT_SERIAL
  portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
  for (;;) {
    while (readRxPin());
    uint32_t start = getCycleCount();
    uint8_t c = 0;
    taskYIELD();
    taskENTER_CRITICAL(&mux);
    for (uint32_t i = 1; i <= 7; i++) {
      while (getCycleCount() - start < i * F_CPU / GPS_BAUDRATE + F_CPU / GPS_BAUDRATE / 3);
      c = (c | readRxPin()) >> 1;
    }
    taskEXIT_CRITICAL(&mux);
    if (c && gps.encode(c)) {
      updates++;
    }
    while (getCycleCount() - start < (uint32_t)9 * F_CPU / GPS_BAUDRATE + F_CPU / GPS_BAUDRATE / 2) taskYIELD();
  }
```

GPSD daemon

- multiplexing; it allows multiple applications to get sensor data without having to contend for a single serial device.
- coping with the hideous gallimaufry of badly-designed protocols these devices use
- on operating systems with a hotplug facility (like Linux udev), GPSD will handle all the device management as USB devices are plugged in and unplugged.
- Several languages supported

```
session = gps(**opts) session.stream(WATCH_ENABLE|WATCH_NEWSTYLE)
for report in session:
                    print report
```

Life is easy with Linux

Sometime you have a driver and view as QMI Modem

```
[ 5.130501] qmi_wwan: loading out-of-tree module taints kernel.
[ 5.134831] qmi_wwan 1-1.4:1.5: cdc-wdm0: USB WDM device
[ 5.137514] qmi_wwan 1-1.4:1.5 wwan0: register 'qmi_wwan' at usb-3f980000.usb-1.4, WWAN/QMI device, ba:7c:bc:a7:0d:d5
[ 5.137698] usbcore: registered new interface driver qmi_wwan
[ 5.146165] usbcore: registered new interface driver option
[ 5.146267] usbserial: USB Serial support registered for GSM modem (1-port)
[ 5.147263] option 1-1.4:1.0: GSM modem (1-port) converter detected
[ 5.150019] sd 0:0:0:0: Attached scsi generic sg0 type 0
[ 5.168171] usb 1-1.4: GSM modem (1-port) converter now attached to ttyUSB1
[ 5.168798] option 1-1.4:1.1: GSM modem (1-port) converter detected
[ 5.170376] usb 1-1.4: GSM modem (1-port) converter now attached to ttyUSB2
[ 5.171092] option 1-1.4:1.2: GSM modem (1-port) converter detected
[ 5.175605] usb 1-1.4: GSM modem (1-port) converter now attached to ttyUSB3
[ 5.187888] option 1-1.4:1.3: GSM modem (1-port) converter detected
[ 5.188542] usb 1-1.4: GSM modem (1-port) converter now attached to ttyUSB4
[ 5.189047] option 1-1.4:1.4: GSM modem (1-port) converter detected
[ 5.189595] usb 1-1.4: GSM modem (1-port) converter now attached to ttyUSB5
root@raspberrypi:~# qmicli --nas-get-signal-strength -d /dev/cdc-wdm0
[/dev/cdc-wdm0] Successfully got signal strength
Current: Network 'lte': '-88 dBm’
RSSI: Network 'lte': '-88 dBm’
ECIO: Network 'lte': '-2.5 dBm’
IO: '-106 dBm' SINR: (8) '9.0 dB’
RSRQ: Network 'lte': '-10 dB’
SNR: Network 'lte': '7.6 dB’
RSRP: Network 'lte': '-109 dBm'
```

GPSD

```
#! /usr/bin/python

from gps import *
import time

gpsd = gps(mode=WATCH_ENABLE|WATCH_NEWSTYLE)
print 'latitude\tlongitude\ttime utc\t\t\taltitude\tepv\tept\tspeed\tclimb’
 # '\t' = TAB to try and output the data in columns.

try:
   while True:
      report = gpsd.next() #
      if report['class'] == 'TPV':

         print  getattr(report,'lat',0.0),"\t",
         print  getattr(report,'lon',0.0),"\t",
         print getattr(report,'time',''),"\t",
         print  getattr(report,'alt','nan'),"\t\t",
         print  getattr(report,'epv','nan'),"\t",
         print  getattr(report,'ept','nan'),"\t",
         print  getattr(report,'speed','nan'),"\t",
         print getattr(report,'climb','nan'),"\t"

      time.sleep(1)

except (KeyboardInterrupt, SystemExit): #when you press ctrl+c
   print "Done.\nExiting.”
```

Sometime isn’t

Boards implement transport, available via
AT Command, but with a vendor specific syntax

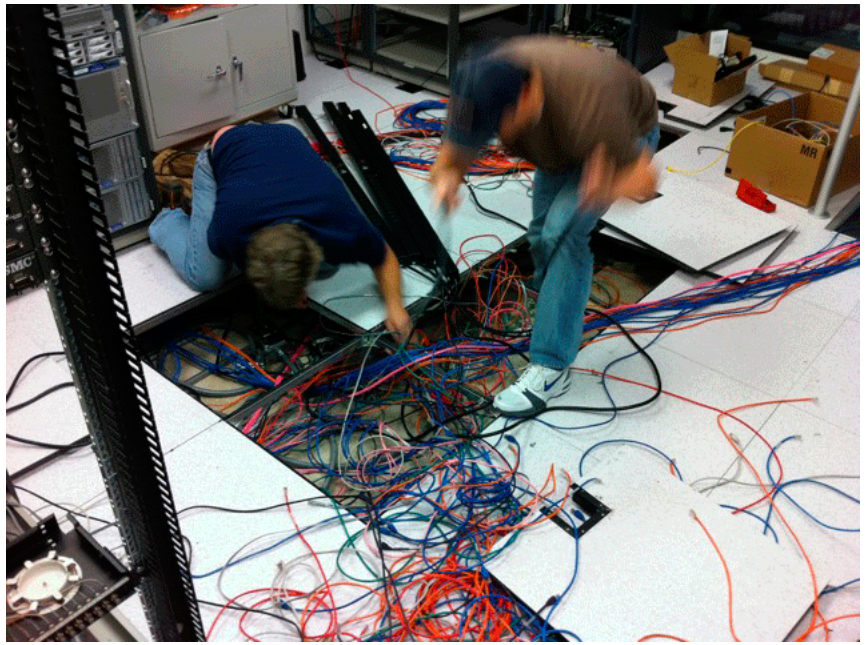
```
do {
        success = sendCommand("AT+CREG?\r", 5000, roaming ? "+CREG: 0,5" : "+CREG: 0,1");
      } while (!success && millis() - t < 30000);
      if (!success) break;

      do {
        success = sendCommand("AT+CGREG?\r",1000, roaming ? "+CGREG: 0,5" : "+CGREG: 0,1");
      } while (!success && millis() - t < 30000);
      if (!success) break;

      sendCommand("AT+CGSOCKCONT=1,\"\",\”wind.internet\"\r");
      //sendCommand("AT+CSOCKAUTH=1,1,\"APN_PASSWORD\",\"APN_USERNAME\"\r");
      sendCommand("AT+CSOCKSETPN=1\r");
      sendCommand("AT+CIPMODE=0\r");
      sendCommand("AT+NETOPEN\r");
    } while(0);
```

Handle the protocol

```
// generate HTTP header
    char *p = buffer;
    p += sprintf(p, "%s %s HTTP/1.1\r\nUser-Agent: ONE\r\nHost: %s\r\nConnection: %s\r\n",
      method == HTTP_GET ? "GET" : "POST", path, HTTP_SERVER_URL, keepAlive ? "keep-alive" : "close");
    if (method == HTTP_POST) {
      p += sprintf(p, "Content-length: %u\r\n", payloadSize);
    }
    p += sprintf(p, "\r\n");
    sys.xbWrite(buffer);
```



Other sensors

- **Temperature**
- **Digital Input/output**

Reading data

```
import machine
i2c = machine.I2C(scl=machine.Pin(5), sda=machine.Pin(4))
byte_data = bytearray(2)
 i2c.readfrom_mem_into(24, 5, byte_data)
 value = byte_data[0] << 8 | byte_data[1]
 temp = (value & 0xFFF) / 16.0
 if value & 0x1000:
     temp -= 256.0
     print(temp)
```

- **Accelerometer**
- **Gyroscope**

Reading data

```
from machine import I2C, Pin
import mpu6050
i2c = I2C(scl=Pin(5), sda=Pin(4))
accel = mpu6050.accel(i2c)
>>> accel.get_values()
{'GyZ': -46, 'GyY': -135, 'GyX': -1942, 'Tmp': 26.7888, 'AcZ': 24144, 'AcY': 68, 'AcX': -1004}
```

Calculate di G acceleration

```
while True :
   #Read the accelerometer,gyroscope and magnetometer values
   ACCx = readACCx()
   ACCy = readACCy()
   ACCz = readACCz()

   print("##### X = %f G  #####" % ((ACCx * 0.244)/1000)),
   print(" Y =   %fG  #####" % ((ACCy * 0.244)/1000)),
   print(" Z =  %fG  #####" % ((ACCz * 0.244)/1000))
```

- **Many others**

Data format

More efficient

HEX (completely custom)

- Fixed format
- Bin format

```
Received data:
080400000113fc208dff000f14f650209cca80006f00d604000400040301011503160300001460000015d0000000113fc17610b000f14ffe0209cc580006e00c0050001000403010115031601000146000001 5e0000000113fc284945000f150f00209cd20000950108040000000403010115001603000146000001 5d0000000113fc267c5b000f150a50209cccc000930068040000000403010115001603000146000001 5b0004
```

OBD Format (kind of standard)

- <PID 1>:<value 1>,<PID 2>:<value 2>, ..
- Custom PID for sensor and GNSS

```
0:68338,10D:79,30:1010,105:199,10C:4375,104:56,111:62,20:0;-1;95,10:6454200,A:-32.727482,B:150.150301,C:159,D:0,F:5,24:1250
```

JSON (some standards, openXC)

- Flexible
- Possible to store as is in NoSQL

```
{ "command": "diagnostic_request", "action": "add", "diagnostic_request": {
                    "bus": 1, "message_id": 1234, "mode": 1, "pid": 5, "payload": "0x1234",
"multiple_responses": false, "frequency": 1, "name": "my_pid"
                    }
}
```

Less efficient

Hardware

Raspberry PI

- High number of Drivers
- Simply to build programs
- Enough cpu power for local operations
- High power consumptions
- Linux

ESP32

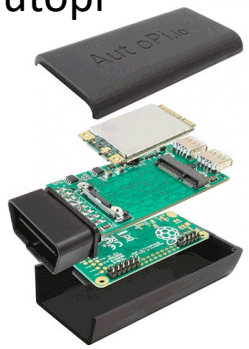
- High number of library
- Compact
- Not Enough cpu power for local operations
- low power consumptions and hibernation (sleep mode 5ma)
- Small factor
- FreeRtos

Spare parts
70-120$

Buy 90-275$

Freematics

autopi



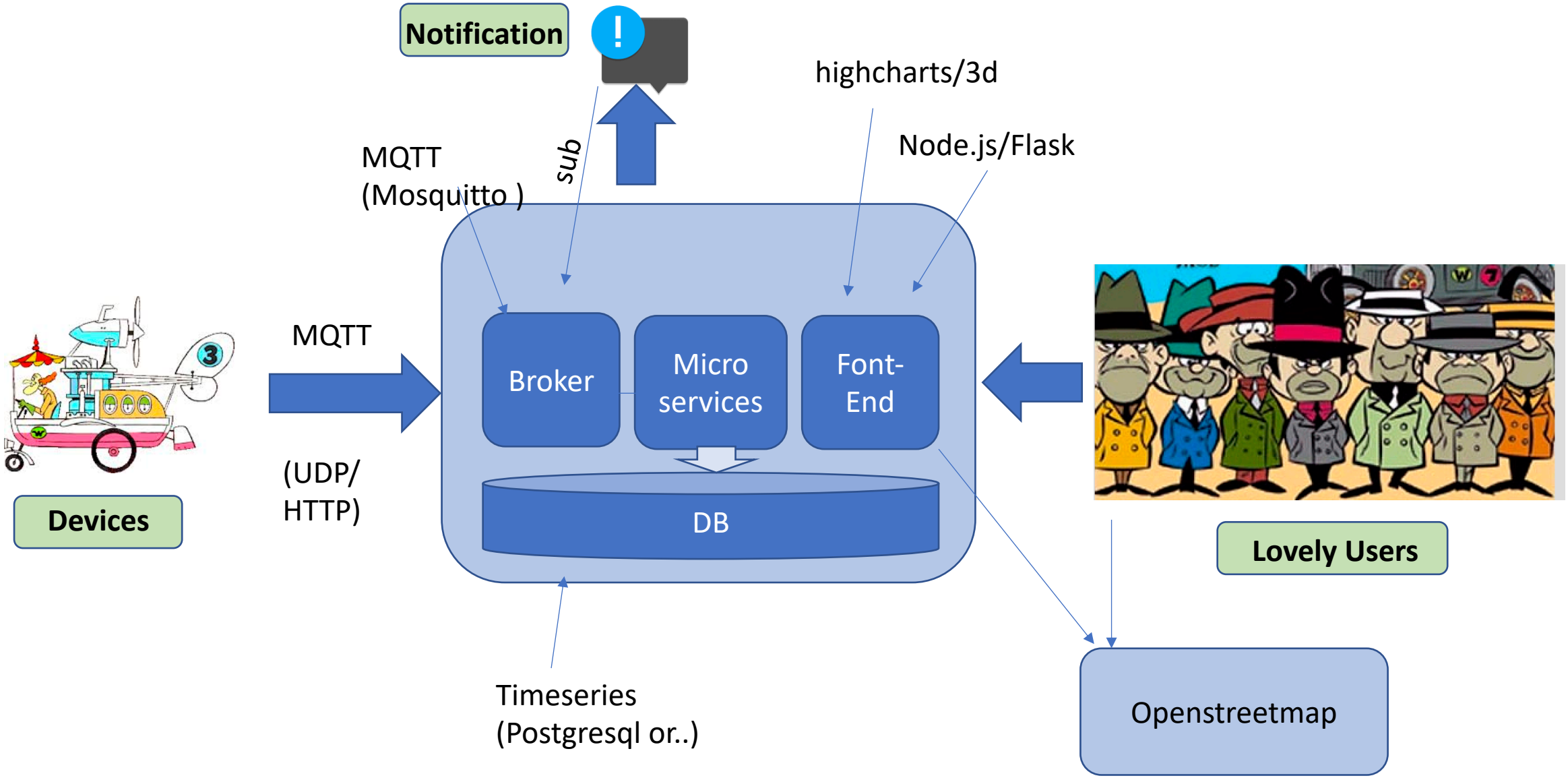
Use cases

Monitoring:
- Notification when the vehicle has been moving
- Going out of a specific area
- Going to fast
- Track all your trip
- Check the Status

Prediction:
- Integration with traffic
- Analyze fuel consumption/well driving
- Route optimization
- Maintenance prediction

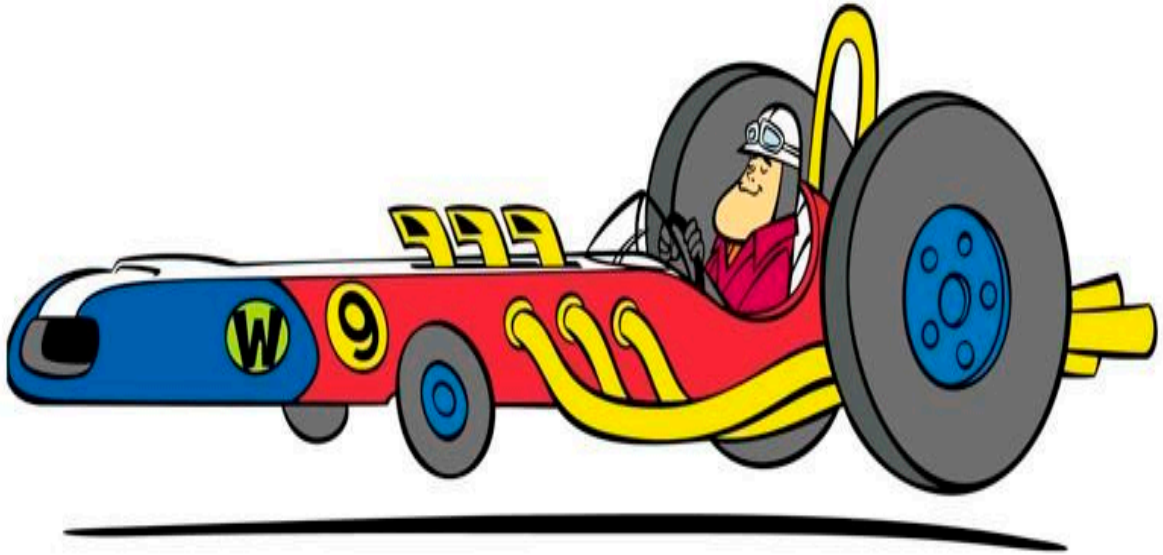
Social:
- Share with friends
- Chat with you vehicle

Build a Backend



Make Or Buy, what is the best for you ?

Vs



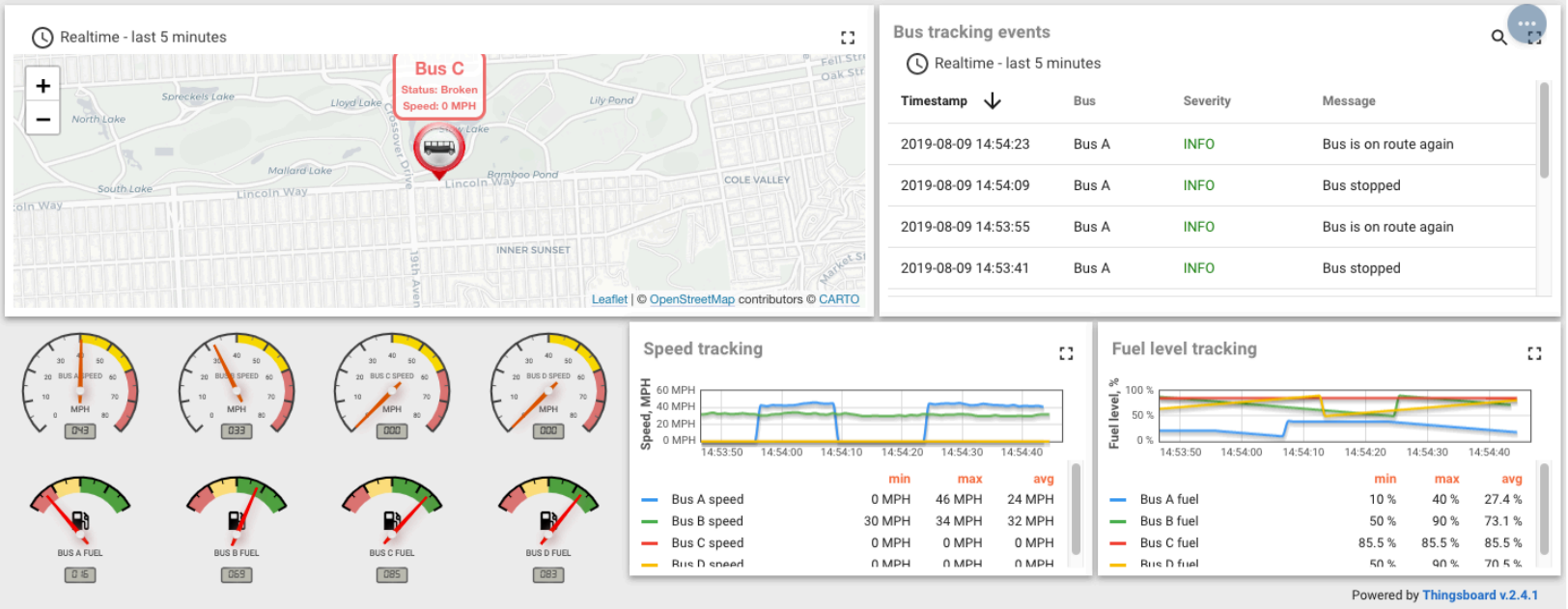
Solutions 1/5

ThingsBoard

is an open-source IoT platform for

- data collection
- Processing
- Visualization
- device management.

https://thingsboard.io

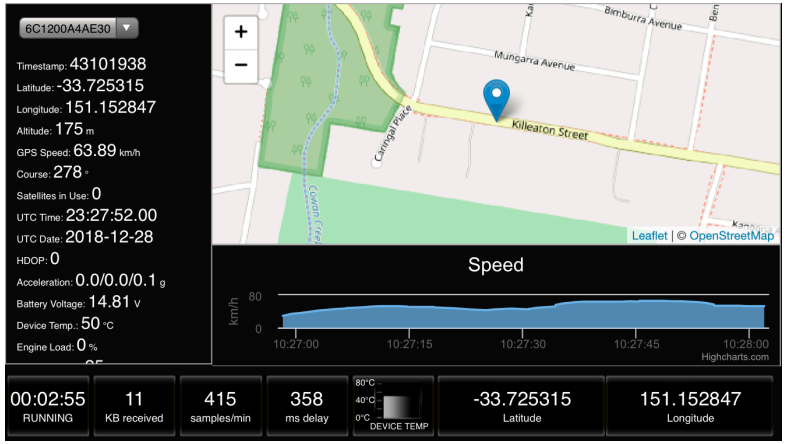


https://thingsboard.io/fleet-tracking/

Solutions 2/5

Freematics HUB

- Reference client implementation in form of Arduino sketches
- Access to real-time and history data by simple REST APIs over HTTP/HTTPS
- No need for application to maintain connection with telematics devices
- Dedicated telemetry transport protocol for high throughput and low latency vehicle data transmission with minimum data overhead
- Remote command queue support, sending commands to device or remotely querying data
- Simply

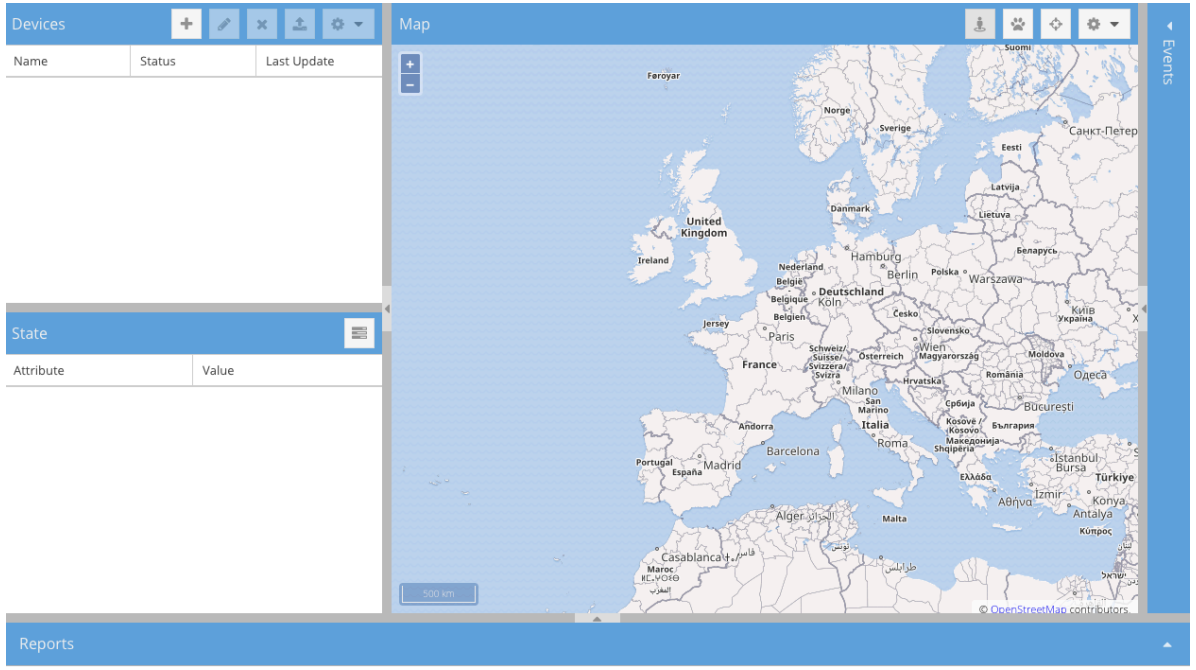


https://freematics.com/pages/hub/

Solutions 3/5

Traccar

Fleet management and tracking system, large number of protocol supported

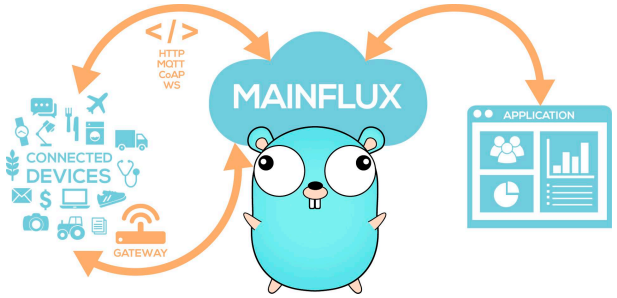


https://www.traccar.org/

Solutions 4/5

Mainflux

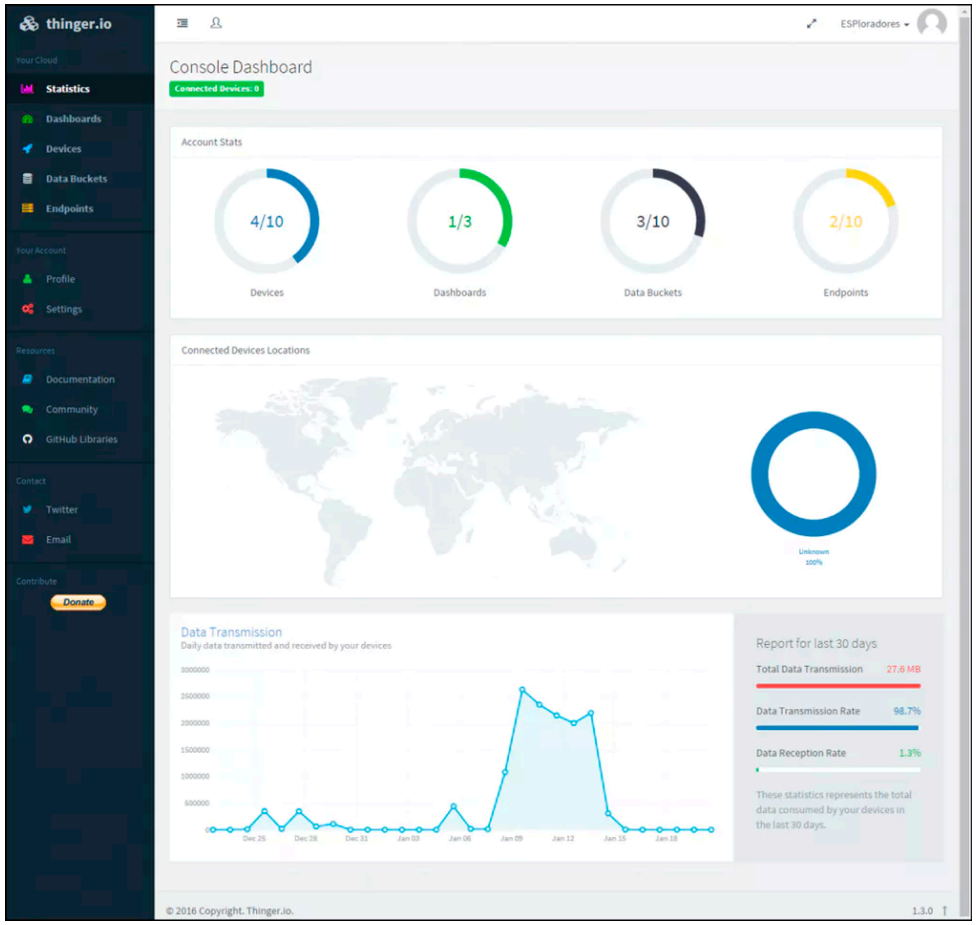
- Open source IoT Platform
- Edge computing



Solutions 5/5

Thinger.io

- Hardware agnostic
- Development environments
- Several graphical tools



https://thinger.io/

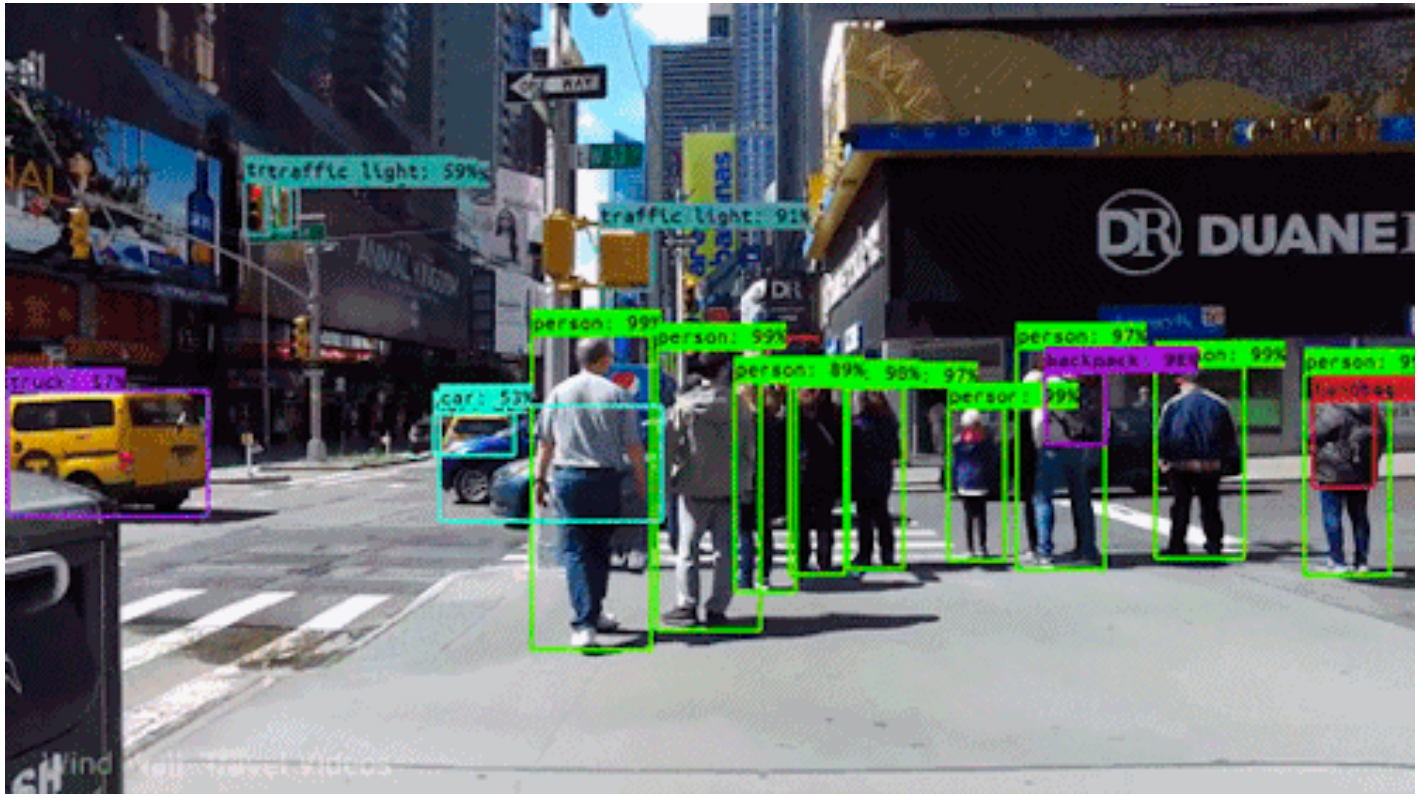
Try some demo?

Demo…

What else ?

Traffic sign recognition

Pedestrian recognition



Nice But ..

Challenges

- GPS localization startup time
  - Outdated catalog
  - Difficult to use the mobile net to help GPS
- Mobile connections
  - Not so stable
  - The coverage is not always good
- Power consumptions
  - Stay online use a lot of energy
  - Heating
- Bike no OBD only CAN Bus

Thank you !

manfred.furuholmen@gmail.com