W3C Web and Automotive Workshop

Tizen IVI Vehicle Data

Mikko Ylinen
Intel Open Source Technology Center
Nov 14th 2012

Agenda

- Position Paper Highlights
- The API
- Demo Setup and Architecture

Position Paper Highlights (1/2)

- In order to be able to provide automotive rich UI and applications, an access to IVI system provided data is needed. Ideally, this is done through standardized automotive Web APIs.
- To date, there's no *standardized* Web API for IVI to access automotive data. Furthermore, different car manufacturers (or even different car models) can provide very different data items.
- Different parties, e.g., GENIVI or Webinos, have started to specify their own specifications for automotive Web APIs.

Position Paper Highlights (2/2)

- Some design principles for a Vehicle Web API
  - The API is lightweight and provides getting and setting data items
  - The API implements an extremely minimal set of data items that are the base set required for compliance
  - The API has mechanisms for event based updates to requested data items
  - The API implements a method for a Web application to query for supported data items, so graceful functionality degradation is possible

The API

`get(eventlist, successCB, errorCB)`

- Retrieve the specified data values, or all data values if blank

`set(eventlist, valuelist, successCB, errorCB)`

- Set the specified data values

`getSupportedEventTypes(type, writeable, successCB, errorCB)`

- Retrieve the list of events available on the platform

`subscribe(eventlist, successCB, errorCB)`

- Subscribe to update events for the specified data items

`unsubscribe(eventlist, successCB, errorCB)`

- Unsubscribe from update events for the specified data items

Example code - events

```
function engineSpeedEvent(data) {
    var newRPMs = data.value;
}
function onLoad() {
    vehicle.subscribe(["speedometer", "engine_speed"]);
    document.addEventListener("speedometer", function(data) {
          var newSpeed = data.value;
    });
    document.addEventListener("engine_speed", engineSpeedEvent);
}
```


Example code - set

```
function setDriverHeight() {
    vehicle.set("driver_seat_position_headrest", 10,
          function() {
                console.log("Driver headrest set successfully!");
          },
          function(error) {
                console.log("Error setting Driver headrest value: " + error.message);
          }
    );
}
```


API FIXME

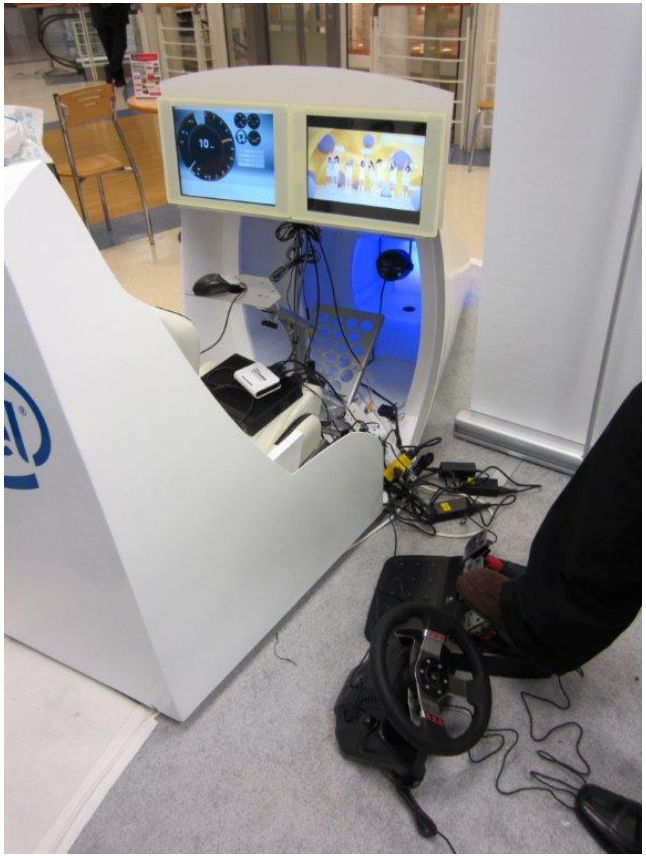
- Create a WebIDL definition from the API
- Improve API’s successCB and errorCB handling
- Listen feedback, re-iterate

Demo Description

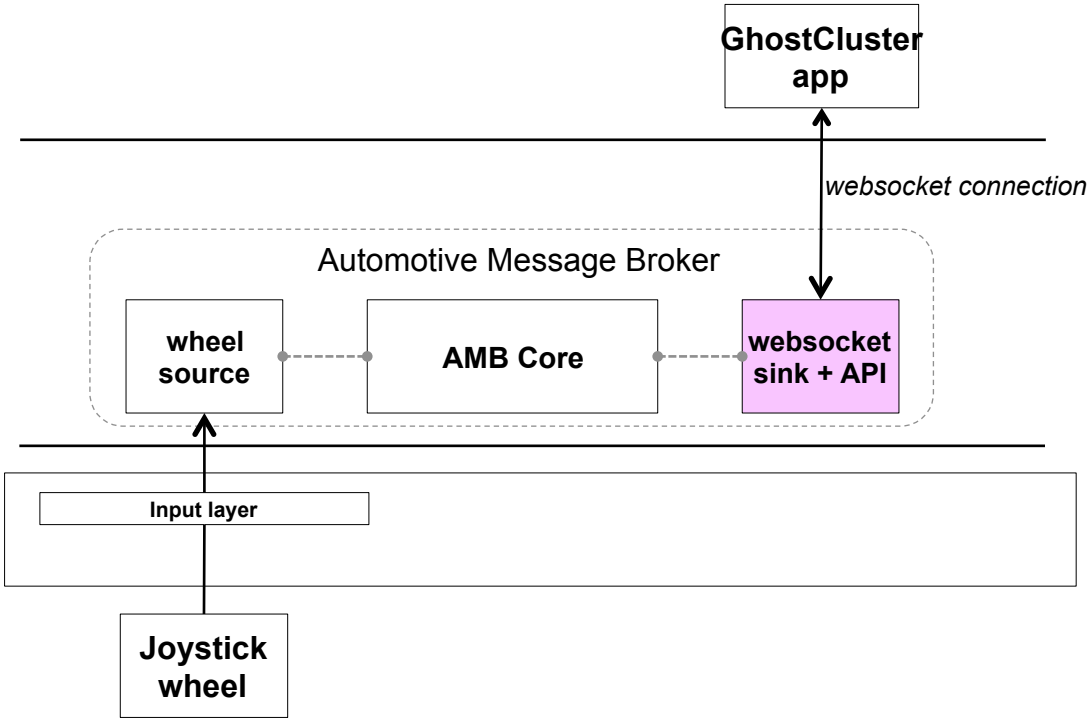
Abstract: *This demo shows how the Automotive Message Broker can be used to translate the action of a steering wheel, pedals, gear shift or any other vehicle information into data that can be displayed in a web application. Tizen includes the Automotive Message Broker which abstracts CAN messages and other low level vehicle information and exposes them to developers as a high level API.*

Hardware: *Logitech steering wheel (inc. pedals, gear stick), NDiS-166, 2 monitors*

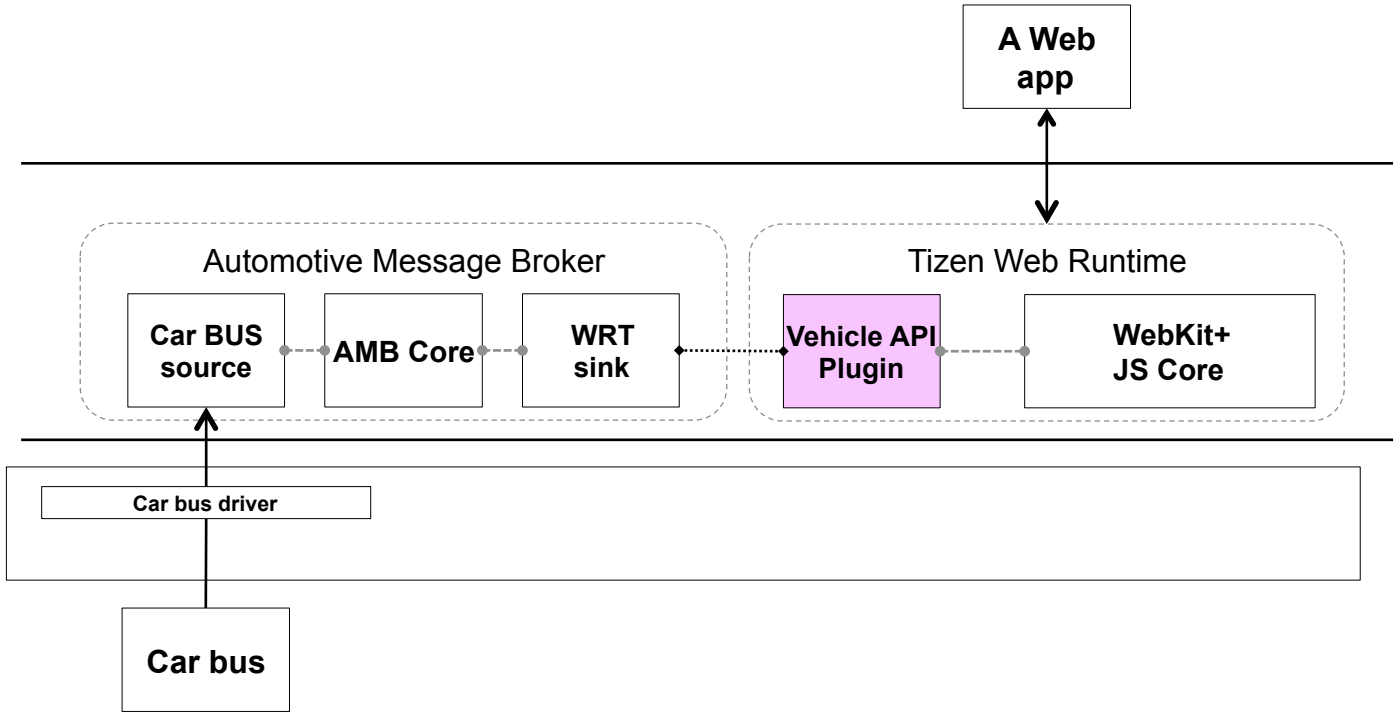
Demo Pics



Tizen IVI Web API Prototype



Tizen IVI Web API Final Architecture



Web Applications

Middleware

Kernel

Performance View

- Environment: from HW to AMB core
- Wheel plug-in: 2000+ properties per second, < 5% CPU
- OBD-II plug-in: 80-100 properties per second

the Code

- API definition

https://github.com/otcshare/automotive-message-broker/blob/master/plugins/websocketsink/test/api.js

- Demo Web App

https://github.com/otcshare/GhostCluster


TURE INTEL LINUX WIRELESS
OP
CS YOCTO
CONNMAN XEN
SYNCEVOLUTION
GUPNP
KVM POKY
OFONO
LINUX KERNEL
SIMPLE FIRMWARE INTERFACE (SFI) ENTERPRISE SECURITY INFRASTRUCTURE
INTEL OPEN SOURCE
TECHNOLOGY CENTER