A Qt-based GENIVI Stack

Johan Thelin, Pelagicore
• Founded 2009
• Offices in Gothenburg and München

Open Source Infotainment
Enabling Great Design
A Changing Business

- Cost ratio hardware / software
- In the old days, a T1 sells a box with software
- Software contains much OEM specifics

- Who should own the software?
- Who should make the software?
User Expectations

CC-BY Travis Goodspeed – http://www.flickr.com/photos/travisgoodspeed/
The User

- Roaming user profiles
  - Your next car
  - Family cars
  - Rental cars
  - Car pooling
- Who owns the user?
  - Google?
  - OEM?
Selling More Stuff

- Selling vehicle functions
- Selling apps
- Selling data (maps)
- etc
Deployment

- Many screens
  - Instrument cluster
  - Heads-up Display
  - Central head unit
  - Rear-seat entertainment nodes

- Combinations
• Downloadable dynamic contents
  - A new way to make money
  - Grow platform features over time
• Scary
  - How to validate the whole system
  - Legal requirements – and indemnification
  - Who develops?
GENIVI is a non-profit industry alliance committed to driving the broad adoption of an In-Vehicle Infotainment (IVI) open-source development platform.

The alliance aims to align requirements, deliver reference implementations, offer certification programs, and foster a vibrant open-source IVI community.

Our work will result in shortened development cycles, faster time-to market, and reduced costs for companies developing IVI equipment and software.
Expert Groups

- Automotive
- CE Connectivity
- HMI Application Framework
- Location Based Services
- Media and Graphics
- Networking
- System Infrastructure
Open Source

• Focuses on specifying a Linux based system
• Reduce fragmentation and reduce cost
• Utilize existing functionality
  – Avoid reimplementing everything for every project
• Utilize common needs with other verticals
  – Media playback, bluetooth, base os, etc
System Compliance

- An evolving compliance specification
- What components to use for what purpose
  - Placeholders – there is a need
  - Abstract – use these interfaces
  - Specific – use this component
- Priorities: mandatory or optional
- Goal: to be able to move components between platforms
...and for Apps

• Works with GENIVI
  – Work in progress!
  – Specify application dependencies and APIs
  – Make it possible to build a common eco system for applications
Adopting Components

- Selects and adopts components from the community
  - connman
  - bluez
  - systemd
  - Linux kernel
  - etc

- Cooperates with the upstream project to adapt to the use case
- Compliance usually focus on interfaces – Abstract Components
Developing Components

• Automotive middleware is not the obvious playing ground of open source hacking
  – Audio Manager
  – Diagnostic Log and Trace
  – Layer Management
  – etc

• Not only for automotive
  – d-bus optimizations – AF_BUS
  – tracker-ivi

http://projects.genivi.org/
Automotive loves communication buses and distributed systems

- CAN, LIN, MOST, FlexRay, Ethernet, d-bus, etc
- Freely move software components between ECUs
IPC Abstractions

- Franca IDL
  - Describe the component interfaces
- CommonAPI C++
  - Generator and support for talking to Franca IDL interfaces (API)
  - Reference run-time based on D-Bus (ABI)
- Possible to change IPC mechanism by replacing the run-time shared object
- We do a Qt wrapper generator based on Franca IDL / CommonAPI C++
class ... : public QObject {
    Q_OBJECT

    Q_PROPERTY(quint16 currentTrack
        READ currentTrack
        WRITE setCurrentTrack
        NOTIFY currentTrackChanged)

    public:
        Q_INVOKABLE play(quint16 trackId);
        Q_INVOKABLE nextTrack();
        Q_INVOKABLE previousTrack();

    signals:
        void endOfPlaylist();
};
The GENIVI Stack

- Focusing at the platform
  - No apps
  - Middleware focus
  - Some OS adaptations
  - No BSP
Components

- Examples from GENIVI

- Node Start-up Manager
- Node State Manager
- Diagnostic Log and Trace
- Layer Manager
- Audio Manager
- User Profile Manager
- Persistency
- AF_BUS
- Kernel config e.g. cgroups
• GENIVI has two base lines Yocto and Baserock
• We work with Yocto
  – Based on OpenEmbedded
  – Recipes
  – Builds rootfs image, sysroot, cross compiler, etc

https://www.yoctoproject.org/
### Layers

- Yocto works with layers
  - Recipes (.bb)
  - Patches (.bbappend)
  - Are prioritized for patch order

- You build an image recipe with top level items, and the rest gets pulled in as dependencies

| Project Configuration | Feature C | Feature B | Feature A | Base Distro (e.g. poky) | BSP |
meta-ivi

- Layer for Yocto with IVI components
- Based on GENIVI compliance
- Makes it easy to get started

http://git.yoctoproject.org/cgit/cgit.cgi/meta-ivi
Where does Qt fit?
- Everywhere!
Qt?

- Where does Qt fit?
  - Everywhere!
- More specific?
  - Applications
  - Compositor
  - Services
Qt for Applications

- Qt and QtQuick rocks for building graphical applications!

- We can generate service proxies from Franca IDL

- Simply wrap in models / proxys for ease of use from declarative
Qt as Compositor

- Build a Wayland compositor using QtWayland

- But, layer manager?
  - Needs support for the layer-manager extension
  - Available as weston-ivi, but needs to be reimplemented through Qt
It is dead easy to write services using Qt

Using the Qt D-Bus bindings
  - Expose QObject instances
  - We're working on doing the same from Franca IDL
The Pelagicore Stack

- We build on a GENIVI / Yocto base

- Adding
  - Services, e.g. Application Manager, tracker-ivi, etc
  - Configurations, e.g. audio routing rules, etc
  - Application run-time environments
  - Applications

- Mostly using Qt!
Application Manager

- Built using Qt
- The Wayland compositor
- Provides information for
  - audio focus
  - access arbitration of shared resources
  - etc
- Launching applications in various run-time environments
- Installing and updating applications
Run-times

• Native code
  – Can be run in a container

• QtQuick with access to the platform services
  – Provide a common set of QML plugins for platform access
  – Possible to pre-load the run-time to reduce start-up times

• HTML5 apps
  – Using Qt WebKit WebEngine
  – Vehicle data APIs are specified by GENIVI
  – Platform access and toolkit bindings are needed
Applications

- Core set of applications
  - Home screen
  - App store
  - Settings
  - System wide search
  - Browser
  - Music player
  - Video player
  - Games
  - Tuner
  - Integrated streaming services, e.g. Spotify, Pandora, etc
  - Navigation
  - 3D vehicle status view
  - etc
Automotive

- Conservative niche
  - Legal requirements
  - Standards compliance
  - Development processes

- The value change and ownership is changing
  - User expectations
  - Cost of software

- Qt fits here
  - Both in apps and system software
Qt and GENIVI

This is what is happening right now!
Thank you!

johan.thelin@pelagicore.com

www.pelagicore.com