Connected Car - Smart Home

Requirements for a new generation of Infotainment Systems

Dr. Roland Krause
Engineering Director
Integrated Computer Solutions Inc.
Connected Cars

- Are quickly becoming reality and features are a distinguishing factor between OEMs
- How will cars communicate with homes and our phones and tablets
- Automotive Infotainment Systems are obvious candidates for communication nodes
- This poses significant challenges
  - Architecture and Design
  - Platforms, Development Tools and Processes
  - Security and Robustness
Some Typical Use Cases

- You are playing music at home and have to leave the house:
  - Lock the house, enter the car
  - The music “follows” you to the car
  - Picks up where it was left off in the house, seamlessly w/o user interaction
Some Typical Use Cases

- An item was left in the car (wallet, cell phone)
  - A smart home knows where it is
  - This works the other way around, too
- A teenager drives the car
  - The car breaks down, e.g. a tire becomes deflated
  - Smart home is informed, smart home TV facilitates video call with the car’s infotainment system
More Use Cases

● Home alarm system was disabled while you are away in the car
  ○ Message is displayed in the car
  ○ Video of the home surveillance cameras can be displayed in the car

● You make a restaurant reservation while at home
  ○ Your car knows time and location of the reservation
  ○ Has navigation information ready and loaded
  ○ Understands your calendar and plans the route to pick up potential companions
  ○ Adjusts the in-vehicle temperature prior to the trip
Intelligent Collaboration

● Very different components of hardware and software
● Purpose is clear: *enhancing everyday life*
● Connecting Car - Home is uncharted terrain because of the proprietary components “Car” and “Home”
● Medium phone as the interface is in early stages (Android Auto and CarPlay)
● Infotainment Stacks should allow for direct connections
Home and Phone Components

- Automotive Components
  - Proprietary, embedded systems
  - Closed (almost) all the way
  - Hundreds of sensors and actuators
  - Center Stack Infotainment System with access to some Telemetry data
  - Internet connected
Home and Phone Components (continued)

- Home
  - Many communication protocols X10, Zigbee, Z-Wave, Insteon, Bluetooth Classic and LE, etc..
  - Even more commercial offerings, mostly all integrated
  - No standards for aggregation, most systems interface with mobile apps directly through WiFi and BLE or through “Smart Hubs” (Amazon Echo, SmartThings, Leviton, Nexia, etc.)
  - IoT M2M protocols could help, e.g. MQTT, CoAP, LWM2M, etc.

- Phone
  - iOS and Android app development is easy, cheap, well documented and mature
Challenges

● Differences in lifecycles
  ○ Cars, consumer devices, phones, home automation
  ○ Different development times
  ○ Dictated by regulation, consumer expectations: car life cycle (>10 years)

● Automotive and mobile industries have very different objectives

● Consumer acceptance
  ○ Depends on significant lifestyle improvements
  ○ Balanced with costs, guided by marketing

● Disruptive technologies
  ○ Mobility vs. car ownership (ZipCar and Uber)
  ○ Autonomous cars will pose new opportunities
Standards and Protocols

● Nurturing this ecosystem
  ○ Means understanding the importance and supporting the development of open standards
  ○ Or the basket of remote controls will doom all efforts

● Automotive manufacturers
  ○ Working directly with cell network operator (AT&T, Sprint, Verizon, T-Mobile...)
  ○ Cell phone makers (Apple, Samsung, LG, Motorola...)

● Remote Vehicle Interaction (RVI)
  ○ Spearheaded by Jaguar Land Rover
  ○ Adopted by GENIVI as - “Open source technology to handle authentication, authorization, discovery, and data exchange between services in a sparsely connected peer-to-peer network”
A Much Simplified Architecture
Prototype Implementation

- ICS developed an In-Vehicle-Infotainment Prototype based on automotive hardware and middleware
  - Modern UI, Plugin based architecture, easily customizable and extendable
- Features a rich Media Stack
  - BYOD Music, Bluetooth Connectivity, Navigation, Rear Seat Entertainment and much more
- Already had remote control capabilities from speech recognition module
- Ideal for fast prototyping of the Connected Car scenario
Media Management

- At ICS we have designed and built many In-Vehicle-Infotainment systems (see e.g. this video)
- When asked to look into Media Management we found this to be a vexing and complex problem
- The challenge for automotive IVI implementations is that
  - People’s media -- their music, videos, audiobooks, podcasts and television -- exist in a multitude of forms and originate from many disparate sources.
  - For example, some music files may reside at home in an iTunes library, others may have been purchased from Amazon Music or Google Play.
  - Media may have then been downloaded to a computer, a USB drive or a phone, or stored on a cloud server.
  - Management of digital rights adds yet another layer of complexity to the situation -- one that can’t be ignored
Requirements

- The job of finding and making available media to the passengers of a car is that of the Media Manager
- First step: Recognizing a Device is brought into the car
- Next: Finding and Indexing Media on the Device
- Possibly: Enhancing Media Information to allow improved search, filtering, etc..
- Definitely: Playing of Media using the car’s advanced audio systems
Requirements (continued)

- Controlling the flow of Media to e.g. different Speaker Zones, Headphones, Videos to headrest screens etc..
  - When multiple occupants drive in the car each individual should be able to enjoy their own audio and video selections.
  - Hence a media manager should be able to direct media to specific passengers.
The Idea - Coding for the Unknown

Today’s media consumers behavior changes rapidly:

- Remember the “Walkman” - Enjoyed it for Decades
- CDs - Lasted maybe 10 years
- MP3s on CDs, USB Pendrives, Less than 5 years
- Cloud based music sharing, Amazon tbd.
- Streaming:
  - Pandora, by all means not saying it’s dead but:
  - Spotify, is the current Darling (< 2 years)
- What is next?
  - The cycles become shorter and shorter
  - Consumers change phones 2-3 years on average
- We must keep in mind that what we create might be partially outdated by the time it is released - Ouch!
Possible Solutions

● Use a Mobile Phone or Tablet OS in the Car
● Use a Web Technology Based Stack in the Car
● Design an Open Standards based, Car specific OS and create a system of plugins and components with open interfaces that can be easily updated
● Architecture of the ICS Media Manager follows this idea
Architecture
Plugin Architecture

Media Manager Core Functionality

- **Load Plugins:**
  - Device Manager
  - Media Devices
  - Media Players
  - Services (Audio Manager, Media Enrichment)
  - Controllers (UI, RC, RVI)

- **Organize Flow of Media Info Data from Device to Player and Device to Controller**
Media Devices
Device Manager Plugin Interface

/** DeviceManagerInterface is a Plugin Interface for DeviceManagers
   * that detect MediaDevices which contain Media that can be indexed
   * by a suitable MediaDevice.
   **/

class DeviceManagerInterface : public QObject
{
    Q_OBJECT
public:
    explicit DeviceManagerInterface(QObject * parent=0) : QObject(parent) {}
    virtual ~DeviceManagerInterface() {}  

signals:
    void deviceCreated(const QString mediaDeviceType, const QUrl mediaDevicePath) const;
    void deviceRemoved(const QString mediaDeviceType, const QUrl mediaDevicePath) const;
};

#define DeviceManagerInterface_iid "com.ics.media-manager.DeviceManagerInterface"
Q_DECLARE_INTERFACE(DeviceManagerInterface, DeviceManagerInterface_iid)
Data Flow

MediaDevice
USB Pendrive

MediaSource

Video Files

Audio Files

MEDIA MANAGER

MediaSource

Video Files

Audio Files

MediaSession

Audio Files

Video Files

MediaSession

Audio Files

Video Files
Core Components

- **MediaSource**
  - Provide interfaces to devices.
  - Devices are physical media such as Phones, iPads, USB thumb drives, Microsoft Media Players, DLNA, Bluetooth, cloud or any source that can be indexed.

- **MediaSource Playlists**
  - Each source presents to the media manager one or more source playlists.
  - The media manager takes these lists and add them to corresponding MediaSessions.
  - For example, video playlists are offered to the session that interfaces to a video player, whereas Bluetooth playlists are offered to a Bluetooth Player which in turn controls a Bluetooth device through the AVRCP protocol.
MediaSession and MediaSource

- **MediaSession**
  - Each MediaSession holds a playlist of tracks specific to a media type e.g. mp3 files, video files or Bluetooth streams.
  - MediaSession interfaces a single instance of a media player for the specific media type.
  - Contains a JSON Object consisting of multiple JSON Arrays, One per MediaType present on the device.

- **MediaPlaylist is a JSON Array**
  - Each JSON Array contains indexing data
  - Indexing data are JSON Objects,
    - one for each media item
    - containing attributes of a single media item
    - e.g., file names, artists, cover art
    - and many other things of interest to the end user
DataStructure: MediaPlaylist

```json
{
  "AudioFileMediaType": [
    {
      "Album": "Southernality",
      "Artist": "A Thousand Horses",
      "CompleteName": "/mm_test/audio/a.mp3",
      "Title": "(This Ain’t No) Drunk Dial",
    },
    {
      "Album": "Billboard Top 60 Country Songs",
      "Artist": "Big & Rich",
      "CompleteName": "/mm_test/audio/b.mp3",
      "Title": "Run Away with You",
    }
  ],
  "VideoFileMediaType": [
    {
      "CompleteName": "/mm_test/video/mad_max.mp4",
      "FileName": "mad_max",
      "Format": "MPEG-4",
      "InternetMediaType": "video/mp4",
    },
    {
      "CompleteName": "/mm_test/video/sup-vs-bat.mp4",
      "FileName": "sup-vs-bat",
      "Format": "MPEG-4",
      "InternetMediaType": "video/mp4",
    }
  ]
}
```
Core Components

- **MediaPlayer**
  - MediaPlayers control the output of media
  - Implement functionality of media reproduction e.g., play, pause, stop, play index, play next, play previous etc.
  - Can also be controlled to direct output to specific channels through an audio manager component.

- **MediaManager**
  - MediaManager maintains a set of session objects and a set of source objects.
  - Interfaces with an audio manager for audio channels
  - Interfaces with a device manager for device notifications.
  - Provides a controller interface which allows for direct user interface (UI) implementation using the toolkit of choice as well as remote control through RVI or web interfaces.
Media Manager Core Functionality

- MediaSessions store sets of playlists (in JSON Arrays)
  - Identified with the MediaSource they came from
  - It is trivial to update playlists upon removal of a device at the cost of rebuilding the playlist and transferring it to the MediaPlayer again.
  - Use of “implicitly shared” container classes is fundamental to a robust and efficient implementation.
Media Manager Core Functionality (continued)

- When a device is connected, e.g. a USB pen drive is plugged in:
  - Media Manager receives a notification from the Device Manager plugin.
  - With the help of a suitable MediaDevice indexing results in a MediaSource object - delivered to and received by the Media Manager.
  - Media Manager stores and accesses MediaSession objects corresponding to MediaTypes contained in the MediaSource.
  - Appends the playlists coming from the MediaSource object to the MediaSession.
  - During this step, filtering and sorting can be applied.
The problem of indexing media is two-fold:

- Files must be found, identified and the results stored.
- Media contained in files and streams must be classified.
- We are looking to answer questions like:
  - How long is this “mp3” file? Who sang this song? Who directed this orchestra?
- All of this information should be available to the user as fast as the medium permits while preserving an always responsive, modern user experience.
The problem of indexing media is two-fold:
- Files must be found, identified and the results stored.
- Media contained in files and streams must be classified.
- We are looking to answer questions like:
  - How long is this “mp3” file? Who sang this song? Who directed this orchestra?
- All of this information should be available to the user as fast as the medium permits while preserving an always responsive, modern user experience.
Indexer - MediaInfo

- Notable Open Source indexing solutions:
  - Gnome projects Tracker and KDE’s Nepomuk are powerful and complex search and indexing solutions appropriate for desktop solutions.
  - Light Media Scanner (LMS), and FFmpeg project’s ffprobe are more suited for constrained environments and use cases.
- Another widely used option is MediaInfo
  - Highly customizable,
  - can easily be integrated in C++ based applications,
  - has support for hundreds of media types and is a fast and robust solution with a long standing track record.
MediaPlayers

- MediaPlayers do not provide UI control elements!
  - They do however have visible elements
  - E.g. video surfaces
  - Control is through a plugin interface
Controllers

- Media Manager employs the concept of active MediaSessions
  - Control the actual playback of media.
  - It calls the active sessions player with the standard actions of playing
  - E.g. play, pause, next, previous, play by index etc..
  - The control of the Media Manager itself is through a MediaManagerControllerInterface that is implemented by a variety of “stateless” plugins.
  - E.g., a simple UI plugin allows for a graphical user interface to be implemented while a “remote controller” plugin allows mobile devices to control the Media Manager and thus its playing functionality.
QtQuick - UI Controller
Integration GENIVI Development Platform
JSON Rpc Controller

- TCP based JSON-RPC
- Utilizes QJsonRpc
- Implements MediaManagerControllerInterface
- Same Interface as UI based Controllers
- Stateless Controller Architecture guarantees that all Controllers are in the same state
- Signal and Slot implementation allows to add controllers at will without changing the MediaManager code
Currently in Development

- Plugins and Indexers for Phones and Tablet devices - these require mobile apps
- Plugins for DLNA devices
- Plugins for Bluetooth device playback
  - BlueZ, AVRCP and A2DP protocols
What to do for Home Connectivity

- Customized Media Player Plugin
  - RVI client implementation
- Tap into a Car CAN Bus or simply retrofit a car with additional sensors and an IoT Hub
  - Seat sensors notify of occupancy
  - Car notifies home controller
  - Home Controller locks doors turns off lights
  - Home Controller brings video to Smart-TV
What is Next?

- Automotive industry is at the center of a paradigm shift
  - Demand for connected lifestyle is evident
- Must keep up with the pace of developing mobile and home automation ecosystems
  - Automotive industry could separate shareable components from differentiating value propositions
  - Joint development of common, shareable components based on open standards
  - Provides the platform to hold pace with development cycles of mobile industry
- Autonomous mobility will accelerate demand for connected lifestyle
Conclusion

- As the vision of autonomous driving changes the role of the automobile itself:
  - Our Vision is to create software that allows the Automobile to be an integration point for Media
  - Similar to the “Connected Home”
  - Central point where Media “comes together”
- Architecture of our components should not withstand the developments of the future but adopt to it.
- Visit us - Talk to us - Work with us!
- How can we help you?
Integrated Computer Solutions
www.ics.com/ivi

We provide:

- Consultancy and services for the automotive industry:
  - Architecture and implementation
  - Middleware components
  - User Interface components

- User Experience Development
  - Workflow engineering
  - Graphics and design services