2015-01-0224
Published 04/14/2015

doi:10.4271/2015-01-0224
saepcelec.saejournals.org

Addressing Challenges in Automotive Connectivity: Mobile Devices, Technologies, and the Connected Car

Patrick Shelly
Mentor Graphics Corp.

ABSTRACT

With the dramatic mismatch between handheld consumer devices and automobiles, both in terms of product lifespan and the speed at which new features (or versions) are released, vehicle OEMs are faced with a perplexing dilemma. If the connected car is to succeed there has to be a secure and accessible method to update the software in a vehicle's infotainment system - as well as a real or perceived way to graft in new software content. The challenge has become even more evident as the industry transitions from simple analog audio systems which have traditionally served up broadcast content to a new world in which configurable and interactive Internet-based content rules the day.

This paper explores the options available for updating and extending the software capability of a vehicle's infotainment system while addressing the lifecycle mismatch between automobiles and consumer mobile devices. Implications to the design and cost of factory installed equipment will be discussed, as will expectations around the appeal of these various strategies to specific target demographics.

CITATION: Shelly, P., "Addressing Challenges in Automotive Connectivity: Mobile Devices, Technologies, and the Connected Car," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(1):2015, doi:10.4271/2015-01-0224.

INTRODUCTION

Contemporary vehicle infotainment systems face an interesting challenge. The basic functionality of a vehicle infotainment system is expected to remain current for the life of the vehicle, perhaps 10 to 15 years when one considers the secondary market for such a vehicle. But conventional attitudes toward cost optimization generally prohibit oversizing or “future-proofing” system resources. The challenge of keeping systems current can be addressed through a combination of enabling periodic software updates, and by extending the system functionality by taking advantage of the capabilities available in connected devices like mobile phones and tablets.

For software updates the path is clear: we are seeing a transition from entire module replacements at the dealership to secure probe USB or Wi-Fi based updates. Ultimately, it's expected the automotive industry will follow the wireless industry's lead with a firmware over-the-air (FOTA) secure update and provisioning strategy.

However, there are many competing technical solutions available for adding new content to an in-vehicle infotainment (IVI) system. At the most basic level, the addition of new content could simply be addressed through the software update mechanism. However, this approach may not offer the level of flexibility necessary to accommodate user-directed customizations. Establishing an app store approach is another option [Figure 1], although this comes with serious security considerations and available system resources must be carefully taken into account. The use of app stores is expected to grow significantly in the coming years as automotive OEMs begin to explore apps not only on IVI systems, but on other components of the automotive platform as well.

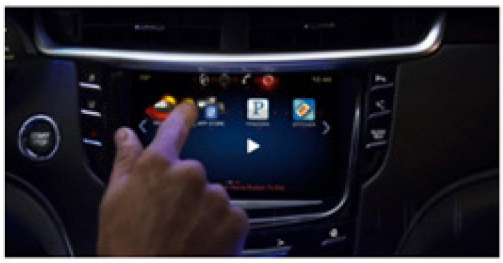

Figure 1. The popular app store approach as it appears on a vehicle's head unit display.

Still, other approaches take advantage of content executing from connected smartphones or other nomadic devices using Bluetooth or other connectivity infrastructures. The use of Bluetooth, according to ABI Research, will rise sharply in the coming years as more car owners grow comfortable using it to listen to music or make phone calls. It's just a matter of time before Bluetooth becomes the preferred method to check emails, exchange text messages, or a post the latest news on a social network account. Bluetooth is expected to enable this in a safe and seamless manner.

THE CHALLENGES

There are many challenges associated with the implementation of connectivity as it pertains to today's connected car. Foremost among these challenges is connectivity *within* the vehicle and connectivity *outside* the vehicle such as Internet via Wi-Fi, LTE, or some other means. Many of the challenges relate to providing connectivity to the myriad of nomadic devices that are brought into the vehicle. Some deal with providing on-going support for the ever-changing formats for audio/visual content, which is often deeply tied to the semiconductor device selected for the system. Both connectivity to consumer devices and support for multimedia formats presuppose the existence of some mechanism for system upgradability over the life of the vehicle. Some challenges result from enabling the use of apps in the vehicle, and providing an application framework that will support these apps in various formats (originating from a variety of sources).

A few of the more pressing challenges for IVI connectivity include:

Codecs and Connectivity

One of the first challenges encountered when designing a new IVI system is dealing with ***codecs and connectivity***. Codecs are components; either implemented in the hardware, or software-based with strong ties to the underlying system-on-chip (SoC) or peripheral silicon devices. Some codecs are open but not all. Further, some codecs require the proper licensing which will have to be figured into the design costs. Codecs are used to encode and decode multimedia content into various formats. In modern silicon architectures, a software-based codec is implemented on any of several processor cores or digital signal processors (DSPs) - as asymmetric multiprocessing (AMP) architectures are becoming increasingly popular. Codecs can also be optimized based on a certain processor or system architecture. The point here is that codecs obviously have implications for the portability of resulting software, as there will be variability in both the interfaces to the codec and the compilation tools used.

The consumer device connectivity aspect centers largely on the type of media used for establishing the connection. In contemporary implementations, this is commonly USB or Bluetooth. In exploring solutions related to device connectivity, the selection of the platform operating system, and optionally, a third-party connectivity solution, will be central to the effectiveness of device connectivity, as will the ability to update and extend the related connectivity software (and quite possibly the related hardware, at least with respect to the physical connection to the device).

Increased support for an interactive connection to an application executing on a connected mobile device has implications far beyond the basic USB/Bluetooth protocol. Extensions are required to support iPod Accessory Protocol, Android Accessory Protocol, and Microsoft Media Transfer Protocol. Some uses of these protocols will be discussed in the next section. Also worth mentioning are the various approaches to Digital Rights Management and the required use of an iPod authentication integrated circuit in the IVI system when connecting to some Apple products. Finally, most IVI systems provide access to unsupported or legacy devices through the availability of an auxiliary analog input. As a result, system software developers must not only map out the types of challenges before the actual design, but work with the silicon/hardware providers to ensure a tight and flawless IVI connectivity solution.

Multi-Zone Audio Connectivity and Multimedia Zones

An extension to the connectivity aspect of the IVI system is ***multi-zone audio connectivity***, which is now expanding to ***content sharing between multimedia zones***. Multi-zone audio makes it possible for the driver and passenger to define individual entertainment experiences. While the driver listens to music and/or navigation instructions over the vehicle sound system, passengers have access to alternative multimedia content, enabled primarily through support for wireless headsets. This multi-zone audio paradigm can easily be extended to support independent audio and visual displays resulting in the sharing of multimedia content (via USB/Bluetooth connections from a passenger's device) creating “multimedia zones” within the vehicle. A multimedia zone gives each passenger the ability to share and receive digital content through the vehicle's head unit. Beyond the base IVI system built by the automaker, these concepts are often available from a dealer/factory install or by aftermarket Rear Seat Entertainment (RSE) systems [Figure 2].

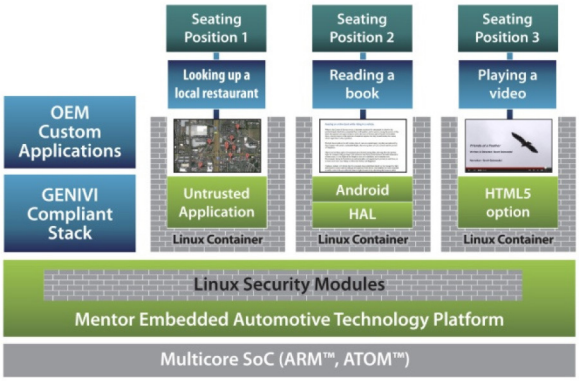

Figure 2. Multimedia zones are created by establishing both “trusted” and “untrusted” partitions that can be separated through the use of hypervisors, Linux containers, and Linux security modules.

While it is expected that vehicle architectures will evolve toward a more consolidated and comprehensive IVI, telematics, RSE, and Instrument Cluster (IC) systems will likely remain as separate and distributed architectures to facilitate vehicle platform scalability. These distributed architectures are more than likely to contain dealer-installed or aftermarket components.

Android Apps

In recent years a great deal of interest has been generated around the use of Android, or more specifically, ***Android apps*** within the automobile. While enabling support for Android apps is technically feasible and may be a great selling point for certain models, there are still serious concerns. Central among these are system security and driver distraction issues. Several approaches exist to deal with the security aspect of Android integration and some of these will be

covered later in this paper. Driver distraction is a very real threat and there are non-profit and government bodies ready to step in and address driver distraction issues.

While in-vehicle Android apps are fueled by the many variations of the Android smartphone, there is one distinct *disadvantage* among automakers that choose to use an app store environment as the sole means of a vehicle's IVI system. There is increasing evidence that vehicle differentiation created by a specific IVI implementation (especially among the mid-range and premium autos) is now a key factor in the success of that vehicle. Android apps, and other apps for that matter, have a very limited scope when it comes to making a vehicle IVI user experience stand out over another user experience. (Although it can be argued, this familiarity between smartphone and dash is exactly what some end users and automakers prefer.) Further, as mentioned before, automakers are reluctant to employ an "all-app" approach due to limited monetization possibilities.

Nevertheless, the market for apps in automotive is expected to grow exponentially. ABI Research expects the number of apps downloaded in cars will jump from 12 million downloads today to 4.3 billion downloads by the end of 2018.

Driver distraction is a prevailing theme when designing any new IVI system. Most of the existing connectivity mechanisms have direct implications of distracting drivers due to the inability to control precisely what appears on the IVI system display. It appears automakers need to do more to mitigate driver distraction. Last year, the National Highway Traffic Safety Administration (NHTSA) reported motor vehicle crashes spiked - to more than 3,300 deaths on U.S. roadways - which can be attributed to driver distraction issues. This uptick has prompted regulators to draft a set of proposed guidelines for automakers and their suppliers. But there is some hesitancy to adopt guidelines because they would take years for widespread adoption and others fear innovation would be stifled. One way to decrease driver distraction is to automate more secondary tasks. Others believe mandatory driver training and better collaboration between technology stakeholders will lesson future potential for driver distraction.

The Internet

Another challenge when adding connectivity to the vehicle is the very nature of the connection. The ***Internet connection*** can be provided using an on-board wireless chipset, or through a USB-connected or Bluetooth-paired cellular device. While systems marketed with an emphasis on safety typically include on-board wireless connectivity, this nonetheless, exposes vehicle OEMs to the necessity of addressing communication technology changes over time - and across geographies. Many of these changes cannot be addressed through a simple software update or FOTA. This is further complicated in the U.S. market as there is a major transition from the established CDMA to the more advanced LTE technology. And for the end user of the vehicle, leveraging an on-board wireless solution as a basic telecommunications device tends to be prohibitively expensive.

In either case, the availability of an Internet connection raises the possibility to provide Wi-Fi hotspot access for passengers, or even using an engineered Wi-Fi network (i.e., a statically configured network with fixed IP addresses) to supply basic connectivity between the modules in the vehicle. An unrestricted Wi-Fi hotspot can certainly be provided through a nomadic device such as a cellular phone or wireless modem (much like what BMW has started to do). An on-board wireless chipset along with in-vehicle Wi-Fi networking opens the door to additional services to passengers, and offers vehicle OEMs the possibility to simplify wire harness design and perhaps better accommodate the adoption of new connectivity technologies.

Streaming Multimedia Content

Internet connectivity brings the possibility of ***streaming multimedia content***. Traditional multimedia players, including those packages available with many legacy RTOS products, are well-suited for playing multimedia from a file. Often, the architecture of these systems requires redesign to better support streaming media. Use of a more sophisticated operating system, especially one with instantiations supporting a desktop environment (Linux, for example) would definitely be beneficial.

Upgradeability

A central issue related to many of the challenges noted in this section is ***upgradeability***. Most new IVI systems are requiring at least some level of support for software updates, even if using a simple USB-connected mechanism. Extensions to this idea include FOTA software update strategies when an Internet connection is available. This introduces further concerns around provisioning, including dealing with the compatibility of component versions and having knowledge of the current software component version configuration in a given system. The later concern is particularly important when dispersing condensed update packages (or "binary diffs") in the interest of speeding update transmittal and thereby reducing the duration of the overall software update. Any update strategy will also have considerable implications for the memory requirements of the system as it is often necessary to maintain redundant memory regions to facilitate the update and protect against update failure.

The update of individual apps in the system will be much less complex as the interdependencies between apps and dependencies on the underlying system services are somewhat easier to address. One option is to simply rely on an app store for handling updates, where a more complete FOTA update strategy is required to update the system software (which may contain some foundation apps). And further simplification of the software update mechanism can be achieved by leveraging apps that actually run on a connected mobile device. Several of the existing connectivity solutions that will be covered in the next section exploit the execution of apps directly on the mobile device, thereby dramatically simplifying software updates (and connectivity hardware updates) for the overall system. Reliance on mobile apps in this way also helps isolate the vehicle OEM to some extent from update concerns.

Security

A topic that is increasingly discussed for new connectivity architectures is ***security***. The openness provided through Internet and device connectivity marks a new epoch for automotive electronics. Much of the discussion seems to revolve around updating the foundational software and the secure deployment and updating of apps. However, security of the Internet connection itself must also be considered (a security breach could result in increased warranty expenditures). Published reports have appeared of malcontents "wirelessly" breaking into vehicles by means of a home-made black box contraption [1]. This type of security breach is invariably the result of hacking into a Wi-Fi or Bluetooth network. Building tighter encryption protocols and safety codes into the host software will certainly lesson the possibility of future hacks.

Another concern reported by the media is the possibility of hackers breaking into a vehicle and commandeering its vital controls [2]. These reports state that a hacker could take control of a vehicle's electronics system and force an unexpected crash. This is highly unlikely primarily because the safety-critical functions of an automobile are partitioned and operate in closed systems. Further, much of the safety critical operations are ultimately based in mechanical engineering, which is immune to software digital hackers.

Security of personal data is also a top concern. If a smartphone is connected to a vehicle's IVI system there is potential for a security breach where sensitive personal data might be stolen. There are a number of software- and hardware-supported technologies that will become commonplace over time to avoid many of the potential threats that exist in this area today. For now, proven measures such as secure boot, OS security features such as Security-Enhanced Linux (SELinux) or other Linux Security module-based Mandatory Access Control (MAC) implementations, and hardware security extensions such as the ARM® TrustZone® technology are readily available - and should be seriously considered. Employing a software hypervisor, which we'll discuss later in this paper, also provides added layers of security.

GPL Licensing

While not necessarily a concern, an awareness of GNU General Public License version 3 (GPLv3) licensing for open source software components is important. Use of open source software such as the Linux operating system may expose some software components as open source, commonly under the GPL version 2 (GPLv2) license. With GPLv3, it is also required to provide instructions and possibly hardware interface components to enable reprogramming of a deployed module. All automotive oriented-software projects, and supporting open source communities (GENIVI, for example) seem to be well aware of this basic issue and are avoiding the use of GPLv3-licensed components for production deployments.

OEM Branding

An important consideration for the vehicle manufacturer is ***OEM branding versus a consistent user interface***. As previously discussed, several connectivity technologies exist today that simply replicate a cellular phone screen on the infotainment system display, or otherwise generate the IVI display image from the phone. While this may provide the end user with a consistent user interface between the phone and the vehicle, which is perhaps desirable in some regard, it does limit the OEM's ability to brand or otherwise influence the user experience. While this approach is seen in lower-end vehicles, deployment of this strategy in mid-range or high-end luxury vehicles has been largely limited to the proprietary solutions from Apple and Google. Later, we will see how HTML5 and other Web-based technologies can offer an elegant solution.

In addition to some of the challenges highlighted in this section, there are also "internal" concerns at play. For instance, a Tier 1 supplier of IVI systems might have a large code base available from legacy systems. New programs may have requirements that are fundamentally incompatible with the existing code base, but the desire to exploit code reuse on the new system remains.

The remainder of this paper will look at some existing approaches, many of which are point solutions or proprietary in nature. We will also explore some general approaches to address the challenges previously noted.

EXISTING CONNECTIVITY SOLUTIONS

This section examines existing technologies related to bringing new or updated applications into the IVI system as well as extending system capabilities. From migrating to a modern application framework to a wide range of solutions (bringing additional applications and features from a consumer device into the IVI system), we will see there are a wide variety of solutions available [Figure 3]. All are viable at some level, most have concerns, and many suffer due to their proprietary nature.

Figure 3. A variety of in-dash connected car technologies exist today, each with its own set of strengths and weaknesses.

First, we should note the ***primordial approach*** used as a technology baseline where the complete content of the system is provided by one or more fixed-content files. The content is programmed to nonvolatile memory at production time, generally it is not extensible, and supports only minimal reconfiguration by the end user. If changes are required or new functionality is to be added, the binary files must be regenerated and the device reprogrammed. In recent years, extensions have been made to the electronic control unit (ECU) bootloader to allow the reprogramming of a module over a vehicle network such as CAN, somewhat simplifying the reprogramming process for fielded vehicles.

While this primordial approach has been used since the earliest microcontroller-based automotive electronic module designs, it continues to be the primary approach employed for most ECUs in the vehicle. Systems based on the AUTOSAR standard also apply this approach. Perhaps a bit surprising, this approach also continues to be the primary method employed in a development of instrument cluster and telematics modules.

A far better approach would be using the ***native application framework approach*** which enables extensibility of the system feature set, where a framework for storage and launch of applications is provided by the system.

Native Application Framework Approach

When implementing this approach one could utilize an existing technology such as the Qt platform, or one of the many available commercial product offerings. When using a native application framework, a secure packaging and transfer mechanism is provided to deploy and install new application content into the system. A secure application launcher should also be present to both validate the application, and execute the application with permissions granted allowing access to appropriate system-level resources.

A sampling of existing native application frameworks include:

MirrorLink

MirrorLink, formerly known as Terminal Mode, has evolved from a rather simple virtual network computing (VNC) screen replication connection into a full protocol that includes mechanisms intended to address driver distraction at the application level, as well as a certification process for applications. In addition to VNC, several other common technologies are employed including IP, USB, Wi-Fi, Bluetooth, RTP, and universal plug-and-play (UPnP). Originally developed and maintained by Nokia and Consumer Electronics for Automotive (CE4A), MirrorLink is now under the oversight of the Car Connectivity Consortium (CCC). CCC members include more than 80 percent of the world's automakers, and more than 70 percent of global smartphone manufacturers. The MirrorLink technology is compatible with the Linux-based automotive platform solution from Mentor Graphics.

MirrorLink was first deployed in the Toyota Touch system, the European version of Toyota Entune [3]. A more recent example can be found in certain subcompact models from U.S. auto manufacturer General Motors; select Chevrolet models carry a diminutive version of the “MyLink” infotainment system which relies in part upon MirrorLink technology [4].

With MirrorLink, the display of the cellular phone is replicated on the infotainment system screen verbatim. Input from either the touch screen or buttons associated with the IVI system is fed back to the phone application for processing. The key point in this approach, and a few of the other solutions we will review, is that the application itself actually runs on the mobile device. No software is required on the IVI system beyond the MirrorLink infrastructure and required device connectivity [Figure 4].

Figure 4. The MirrorLink approach enables popular smartphone apps to appear on a vehicle's in-dash display or head unit, providing a great deal of familiarity for the driver.

The MirrorLink approach could essentially eliminate the need to support an application framework for user-directed extensions to system functionality. One major downside to this approach is that the OEM relinquishes the ability to re-brand displayed content. However, it does provide a mechanism for a Tier 1 supplier or other hardware vendor to support a quick demonstration of high-level features on newly developed hardware. The beauty of MirrorLink is that it can conceivably leverage any available transport medium, such as Wi-Fi, USB, or Bluetooth. In principle, this approach also largely eliminates much of the customer support burden from the OEM, and places this burden squarely with the handset provider. Provided there are no changes to the protocol or the connectivity mechanism, software updates can be isolated to the connected consumer device. Although the claim is that MirrorLink technology is operating system agnostic, at the time of this writing, Apple is not a member of the CCC and iOS-based devices are not supported [5].

CarPlay

A different, but closely related technology developed by Apple is **CarPlay**, in which the IVI system display is completely rendered by the connected iPod or iPhone. The connection from the phone to the IVI system supports graphical output to the IVI system and user input back to the connected phone. This approach offers a consistent user interface between the vehicle and the consumer device, quite possibly resulting in improved ease of use. However, the phone determines the content and format of the output produced on the IVI system display, which could have implications around driver distraction. Under this approach, the OEM loses ability to brand the content, potentially inadvertently resulting in an “iCar.” CarPlay supports only a subset of the iPhone capabilities including multimedia and telephony and therefore, is less general than MirrorLink which applies to other applications as well. CarPlay supersedes the “iOS in the Car”, “Siri Eyes Free” and “iPod Out” initiatives from Apple. Introduced with iOS 7.1. This feature is only available on Apple products, and is rolling out in a large number of OEMs.

Android Auto

Google's **Android Auto** initiative is similar to CarPlay, in that it is ultimately a projection technology. That is, the connected phone renders the image displayed on the in-vehicle display, and applications reside and run on the phone. This technology is also expected to be widely deployed in the near future.

Functionality similar to that embodied by the projection technologies from Apple (CarPlay) and Google (Android Auto) could become available from other device manufacturers; however, the specific interfaces will likely continue to vary between phone or device manufacturers.

HTML5

There has been a lot of interest in the use of Web technologies for automotive systems - IVI systems in particular. These technologies have a lot of offer. In addition to having a large base of developers, technologies such as **HTML5** enable rebranding and restructuring of content that emanates from the mobile device. Expectations for current HTML5 implementations are that HTML content would reside either locally on the IVI system, or on a connected mobile device, and would be seen on the IVI system display. In the case of content residing on a mobile device, an HTTP server is typically required to transmit the content. The HTML5 mechanism allows for a seamless transition as the cloud computing paradigm is adopted, reusing the same infrastructure as applications and content migrate from resident in the IVI system, to resident in the phone, and ultimately residing in the cloud [Figure 5].

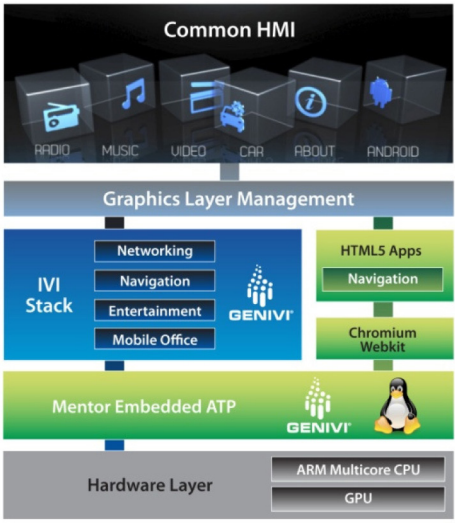

Figure 5. HTML5 is supported by WebKit-based Chromium. Performance has been optimized through the use of the GPU in the underlying SoC.

The major drawback with a Web technologies approach is one of scalability; many evaluations have shown that the HTML approach does not scale well to low-end processors. And a tremendous effort is required to optimize an HTML rendering engine for a specific processor architecture and graphics processing unit (GPU). These shortcomings are expected to diminish over time. And of course, even when a more complete reliance on cloud technology prevails, there must be a solid fallback strategy to cover in the event of network outage or loss of connectivity due to local obstructions.

Ford AppLink

Ford AppLink offers a rather elegant solution to the driver distraction issue. Like many of the other mechanisms discussed here, AppLink allows the IVI system functionality to be extended by accessing new applications and services provided by a connected mobile device. However, in the case of AppLink, when an application is invoked on the mobile device, the mobile device display goes inactive, and all application control is performed strictly through voice commands and steering wheel control buttons.

Ford has announced that it will make the source code for its AppLink API available through a project hosted by the GENIVI Alliance, (referred to as Smart Device Link) so there is the possibility we will see this system widely deployed [9]. AppLink ships in the current Ford SYNC, but it remains to be seen whether this technology will be embraced by other vehicle OEMs.

The Cellular Connectivity Approach

One other dimension to the IVI system is whether the **cellular connectivity** is provided by an omnipresent on-board module, or through a nomadic consumer device by way of tethering. Either approach can be enabled by the OEM. However, the approach taken will have commercial implications including determining the beneficiary of the ongoing revenue stream for connectivity services, and to determine the responsible party with respect to software updates and addition of new feature content. The selection of onboard versus tethered connectivity will also have implications to the overall availability of connectivity in the event of an emergency.

There are a wide variety of strategies available to extend the functionality of an IVI system after production. Some of these are implemented directly on the IVI system and require an accessible mechanism to update the software in that module. Most approaches attempt to extend the IVI system functionality by tapping into applications and services that reside on a connected consumer device, thereby reducing the need to update software in the module. And one approach, HTML5, offers the possibility of seamlessly supporting cloud-based applications and content.

STRATEGIES TO ADDRESS CONNECTIVITY CHALLENGES

In this section we will look at some strategies to solve the concerns highlighted throughout this paper. Until we begin to see some standardization (if even *de facto* standards) around application frameworks, software updates, and device connectivity, we will more than likely require a multipronged approach. In the meantime, there will continue to be a strong desire to confine frequently updated components and installation of extended applications to the mobile device, while implementing on-board cellular network connectivity for safety purposes. Any security features available in the underlying hardware and software platforms should be

aggressively leveraged. And of course, embracing modern Web-based technologies around the applications and application framework where possible will be important.

The use of ***open source software (OSS)*** can help address many of the connectivity issues, as most new connectivity technologies are prototyped and initially released in the context of the Linux operating system. The Mentor® Embedded Automotive Technology Platform (ATP) consists of a full commercial-grade Linux distribution with IVI-specific middleware including components from the GENIVI Alliance, a state-of-the-art build environment, sophisticated development tools, and a professional services organization with strong background in automotive applications [Figure 6].

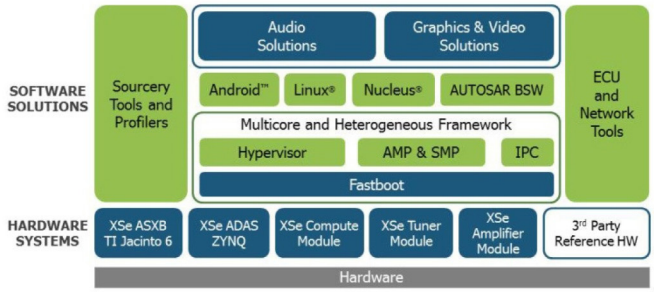

Figure 6. The Mentor Graphics automotive portfolio offers both software and hardware automotive system solutions.

Given the early availability of new technologies under Linux, and the broad base of users and developers, one can easily see why open source software is becoming the software platform of choice for IVI systems. The additional time needed to port new technologies to niche proprietary operating systems can be eliminated from development schedules, at the same time overall development and production costs are dramatically reduced. And advanced security features, the development of which has been driven in part by the United States National Security Agency (NSA), have been available in Linux for quite some time and continue to evolve.

To eliminate many of the connectivity challenges discussed in this paper, an IVI system should ***incorporate an application framework*** of some type. Ideally, the framework will support the installation and launch of applications of differing origins. That is, the applications may have been developed using different methodologies and programming paradigms such as HTML, native OpenGL, Qt, or one of the popular automotive human machine interface (HMI) development systems. There should be a way to limit the underlying system resources that are available to a given application, similar to what is commonly implemented in smartphones today. For example, access to GPS data would commonly be granted to a navigation or weather application, perhaps under the user's approval. But sensitive information from the vehicle communications networks (such as CAN) should only be made available to applications with specific approval and for well-understood use cases.

With an application framework, the system will be able to tap into an app store concept. It is unlikely that unrestricted access to mobile apps from a repository such as Google Play or Amazon Appstore for Android will be desired by a vehicle OEM. However, having 30-50 pre-qualified apps, made available through an app outlet hosted by the OEM, would definitely be desirable. When deploying apps from an app store, a secure transfer mechanism should be used. The installer will need to unpackage the app into the file system and hook the application into the menuing scheme, and the application framework should contain provisions to confirm the validity of the application prior to launch.

Incorporating a ***Type-1 hypervisor*** [Figure 7] into an IVI system design can address several concerns, including; software reuse through integration of software components from a legacy RTOS environment; security through isolation of different operating system domains (perhaps including the integration of the Android operating system and/or Android applications); and enabling the integration of other automotive-oriented operating systems and software components such as AUTOSAR. A Type-1 hypervisor can also have a positive impact when addressing fast boot requirements. Using a hypervisor can also aid in software license segregation and offer a simplified approach to software updates.

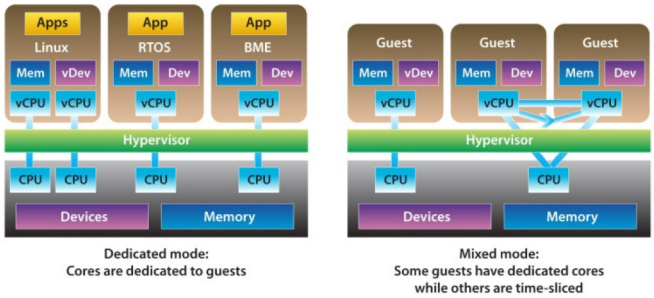

Figure 7. Type-1 hypervisor configuration options: Dedicated Mode (left) - the cores are dedicated to each guest. Mixed Mode (right) - some guests have dedicated cores while others are time-sliced.

One other popular concept related to incorporating an Android environment into a secure IVI system is ***Android via hypervisor or Linux Containers (LXC)***. Linux containers have been supported since version 2.6.29, and are well supported in current distributions. Use of Linux containers may make sense for some use cases and not others. For instance, if a Linux-based IVI system has support for multiple displays, one of those displays could be employed to implement a RSE system. In this case, it is possible to use the base Linux distribution and related middleware for the IVI system implementation, and devote the RSE screen to an Android distribution running in a Linux container. However, where greater separation is required, a hypervisor is likely more appropriate.

There have been instances where the Dalvik Virtual Machine (DVM) and related libraries for Android have been ported to other, non- Android operating systems including Linux. This “Android Player” approach represents a good architectural option if the primary objective is to simply execute Android applications. However, this can become a maintenance burden as new versions of Android come out, or if some applications do not work properly in this “emulated” environment.

Of course, Android could be included through the use of asymmetric multiprocessing (AMP), allowing Android to run on a subset of the cores on the SoC device, and something like Linux running on the remaining cores. The difficulty with this solution is that mutually exclusive peripheral allocation must be determined, and there may be security concerns similar to running a native Android distribution. A hypervisor can help by acting as a resource "monitor" in this AMP configuration.

Along this line, an extreme case of isolation can be achieved by allowing applications to run on a remote connected device. This has been seen in a few of the point solutions to connectivity available today, and discussed earlier in this paper.

Throughout this paper we have repeatedly seen the requirement to support a ***software update strategy***. Most contemporary IVI system designs now require some mechanism to perform software updates to the module, and this is in part due to the dramatically shortened design cycles we see today. One technology that can assist on the software update front is FOTA, which is primarily targeted at updates over the wireless communications infrastructure. But at a minimum, provisions for updating over USB should be available.

When deploying an update strategy, it is useful to consider the notion of *conditional* versus *silent* (or forced) updates. It may be desirable to allow the user to selectively update some software components. For example, if the user is satisfied with the performance of the voice recognition engine, and has heard of concerns with a newer version, they may choose to defer the update of the speech recognition database. And related, there could be provisions to "back out" update that has already been installed. This will potentially complicate the management of version compatibilities, and there are formal approaches to address this such as the OMA DM device management protocol, specified by the Open Mobile Alliance.

Finally, one should not ignore the pervasive qualities of the Internet. Smart developers are already planning for migration to ***cloud-based applications and services.*** This planning should accommodate and promote the use of HTML5 and other related Web-based technologies wherever possible. In the end, the real benefit of cloud-based applications and content is that system updates are simple and immediate. Application updates pushed to a cloud server will be reflected the next time a fielded IVI system reloads or launches the app.

CONCLUSION

Without question, there are solutions readily available to address the challenges in automotive and device connectivity; whether it's the need to update existing automotive software, or bring safer more seamless interactions between driver and machine. Nevertheless, problems exist today that will continue into the near future. In fact, difficulty in using a smartphone within the automobile, along with entertainment and navigation functions, drew the loudest complaints from consumers in the latest J.D. Power & Associates survey on new-car quality [10]. As discussed in this paper, much of the disconnect stems from the inability of automotive IVI systems to keep up with the latest smartphone releases.

How can the technology in cars, which often takes years to design and develop, possibly keep pace with the latest smartphone update or release? This paper has explored the two primary options: automotive OEMs investing to build their own IVI system (the "built-in" approach), or OEMs simply mimicking the functionality of the latest smartphone on their dashboard displays (the "brought-in" approach). And while it's true that many automakers are approaching IVI technology in a piecemeal fashion, consortiums like the GENIVI Alliance are making noticeable impacts. The CCC has also created a type of protocol for tethering smartphones inside the automobile, which many of the world's top OEMs are now following.

Figure 8. A digital instrument cluster display as part of the vehicle's IVI system.

It is safe to say most auto manufacturers believe developing their own native application framework is the best approach. Much of the same functionality and ease of use operations that have led people to bond with their smartphones can also be replicated in the automobile. Further, for safety and overall user experience, building a complete IVI system framework is the preferred way to go.

It's recommended that a hybrid approach between these two approaches be adapted at this time for the short term. That is, some level of support should be developed for the connectivity solutions available today, until we begin to see emergence of a common approach to IVI. Incorporating many of the solutions discussed will have the added benefits of reduced time to market and overall cost reduction. Benefits will also most certainly be realized during the product development cycle of your IVI system.

In the end, a cloud-based approach to application provisioning and delivery will prevail. The physical connection to the network may vary (handheld mobile device, on-board cellular module, etc.) but a relatively simple module will suffice as the factory-installed IVI system - which can be swapped out as IVI systems become available.

REFERENCES

1. Today News, "Police admit they're stumped by mystery car thefts," http://www.today.com/news/police-admit-theyre-stumped-mystery-car-thefts-6C10169993, accessed Oct. 2014
2. Forbes, "Hackers reveal nasty new car attacks - with me behind the wheel," http://www.forbes.com/sites/andygreenberg/2013/07/24/hackers-reveal-nasty-new-car-attacks-with-me-behind-the-wheel-video/, accessed Oct. 2014
3. Autoblog, "Toyota first to market with MirrorLink functionality in iQ," http://www.autoblog.com/2011/10/26/toyota-first-to-market-with-mirrorlink-functionality-in-iq/, accessed July 2014
4. Wired, "Exclusive: Hands-on with the 2013 Chevy Spark's MyLink system," http://www.wired.com/2012/06/my-link-spark/, accessed July 2014

5. Crutchfield, "Introducing MirrorLink: An overview of the next wave of smartphone interactivity," http://www.crutchfield.com/S-oPRP3t8BvMQ/learn/mirrorlink-and-your-car-stereo.html?showAll=N, accessed Oct. 2014
TechCrunch, "iOS 4's hidden 'iPod Out' feature brings iPhone support to your car without the messy third party UI," wheel," http://techcrunch.com/2010/07/13/ios-4s-hidden-ipod-out-feature-brings-iphone-support-to-your-car-without-the-messy-third-party-ui/, accessed Oct. 2014
6. Wikipedia, Apple iPhone 5, http://en.wikipedia.org/wiki/IPhone_5, accessed July 2014
7. Wired, "Apple's in-car Siri integration hits a roadblock," http://www.wired.com/2013/04/siri-eyes-free-roadblock/, accessed Oct. 2014
8. AutomotiveWorld, "Ford Extends Commitment to app developers by contributing AppLink to open source GENIVI Alliance," http://www.automotiveworld.com/news-releases/ford-extends-commitment-to-app-developers-bycontributing-applink-to-open-source-genivi-alliance, accessed Oct. 2014
9. Power J.D., "Despite continuing challenges with in-vehicle technology, automakers post a strong improvement in initial quality," http://www.jdpower.com/press-releases/2012-us-initial-quality-study, accessed July 2014

CONTACT INFORMATION

You can contact the author at
pat_shelly@mentor.com

ACKNOWLEDGMENTS

I would like to acknowledge the awesome assistance I received from Scott Salzwedel in creating this paper. Although he has abandoned his native Detroit for the green fields of Portland, Oregon, his interest and knowledge of the American automotive industry remains intact.