A Hackers Vacation

January 2016 Issue

Bits Versus Electrons

A Hacker's Vacation

By Bob Frankston

I just returned from a trip to San Francisco with my wife. Part of the fun for me was renting a Tesla. The scarcity of dedicated controls and buttons made it very clear that much of the functioning of the car is defined by software. That experience serves as inspiration for this column, but I'm writing about the ideas rather than the Tesla itself. This is similar to when I write about open application program interfaces (APIs) for navigation systems. Those APIs might or might not be available in a real car, but we can still consider the implications.

After my first day of driving, I settled into my hotel for the night and noticed that it provided a clock radio with a plethora of specialized buttons and a place to mount a first-generation iPhone. This is a hard coupling, and hotels can't keep replacing the clock radios for each change of interface and device connector. One solution is for users to bring their own devices, and this is happening, just as travelers are using their cellular phones instead of the hotel's phone system to make calls.

The airlines are ahead on this trend and are removing their built-in screens and transitioning to a bring-your-own-device approach. Some airlines give out iPads, but, increasingly, passengers are expected to have their own devices to stream content from a server on the aircraft. In the future, I may not even need to bring my own device but would simply bring my mobile persona (personalization settings) to the device as I do when I open up a browser. We are evolving toward common practices that will allow such portability, such as with HTML5 as a virtual platform (see my April 2014 "Bits Versus Electrons" column). The physical form factor still matters, and I may want to take advantage of hotel-provided features like an ability to project the time onto the ceiling. Or why not have a video screen there so I can watch sheep jumping over the bed as I try to fall asleep?

I purposely said an evolving set of practices rather than standards to emphasize that such approaches are adopted rather than imposed. The approach works best when the elements are loosely coupled as with APIs.

USB

USB provides a good lesson in how pragmatic standards evolve. USB was created in the mid-1990s as a way to simplify connecting simple devices to personal computers. It succeeded because it was easy to understand—you just plugged a USB device into a computer (or a hub) and could see the relationships between the device and the computer. The use of a wire had an advantage over Bluetooth in that you didn't need elaborate protocols for pairing nor did you need to create a security protocol beyond controlling physical access to the wire.

USB had one other advantage that might not have been obvious at the time—it provided power to the peripherals so you only needed a single wire. Thus, when it became time to standardize a power connector for devices, especially cell phones, it was easier to get agreement on USB than to develop a new standard. The standard process didn't stop people from kludging around USB to use that single wire to power disk drives and finding ways to get more current for charging.

These practices have culminated in the USB-C/3.1 standard that can serve as a smart power supply up to 100 W. We can also use it as a generic network connector by using it as if it were an Ethernet wire by using a generic driver. It's still a work-in-progress, but I expect to see the USB-C connector becoming very common, even if only for a power supply.

As an aside, I expect to see USB-C as a first step into a shift to dc power. But that's another topic worthy of deeper discussion for another time (and another column). Maybe one day, I'll be able to charge a Tesla using a successor (USB-D?) both in the sense of transferring power and transferring the payment.

TESLA—IN PRACTICE AND THEORY

I want to be clear—no Teslas were sacrificed in the writing of this piece, but I did drive one. Fortunately, the owner was there to give me an initial tour of the car. For the most part, the user interface is discoverable once you understand the idioms. I adjusted the seat to my liking and saved it as the default user, although I could've added

Digital Object Identifier 10.1109/MCE.2015.2484918
Date of publication: 16 December 2015

This article is now available here on the IEEE site. You can see other columns and writings in Further Readings.

A Hacker's Vacation

Or what I discovered on my summer vacation renting a Tesla, connecting my devices and searching for my luggage.

I just returned from a trip to San Francisco with my wife. Part of the fun for me was renting a Tesla. The scarcity of dedicated controls and buttons made it very clear that much of the functioning of the car is defined by software.

That experience serves as inspiration for this column but I'm writing about the ideas rather than the Tesla itself. This is similar to when I write about open APIs for the navigation system. Those APIs might or might not be available in a real car but we can still consider the implications.

After my first day of driving I settled into my hotel for the night and noticed that the hotel provides a clock radio with a plethora of specialized buttons and a place to mount a first generation iPhone. This is a hard coupling and hotels can't keep replacing the clock radios for each change interface and device connectors.

One solution is for users to bring their own devices and this is happening just as travelers are using their cellular phones instead of the hotel's phone system to make calls.

The airlines are ahead on this trend and are removing their built-in screens and moving to a BYOD (Bring Your own Device) approach. Some airlines give out iPads but increasingly passengers are expected to have their own devices to stream content from a server on the aircraft.

In the future I may not even need to bring my own device but would simply bring my mobile persona (personalization settings) to the device as I do when I open up a browser. We are evolving towards common practices that will allow such portability as with HTML5 as a virtual platform (as per my April 2014 column). The physical form factor still matters and I may want to take advantage of hotel-provided features like an ability to project the time on the ceiling. (Hmm – why not a video screen there so I can watch sheep jumping over the bed as I try to fall asleep?).

I purposely said an evolving set of practices rather than standards to emphasize that such approaches are adopted rather than imposed. The approach works best when the elements are loosely coupled as with APIs.

USB

USB provides a good lesson in how pragmatic standards evolve. USB was created in the mid 1990's as a way to simplify connecting simple devices to personal computers. It succeeded because it was easy to understand – you just plugged a USB device into a computer (or a hub) and could see the relationships between the device and the computer. This use of a wire had an advantage over Bluetooth in that you didn't need elaborate protocols for pairing nor did you need to create a security protocol beyond controlling physical access to the wire.

USB had one other advantage that might not have been obvious at the time – it provided power to the peripherals so you only needed a single wire.

Thus when it became time to standardize on a power connector for devices, especially cell phones, it was easier to get agreement on USB than to develop a new standard. The standard process didn't stop people from kludging around USB to use that single wire to power disk drives and finding ways to get more current for charging.

These practices have culminated in the USB-C/3.1 standard that can serve as a smart power supply up to 100W. We can also use it as a generic network connector by using it as if it were an Ethernet wire by using a generic driver.

It's still a work-in-progress but I expect to see the USB-C connector becoming very common even if only for a power supply.

As an aside I expect to see USB-C as a first step into a shift to DC power. But that's another topic worthy of deeper discussion, for another time, another column. Maybe one day I'll be able to charge a Tesla using a successor (USB-D?) both in the sense of transfer power and transferring the payment.

Tesla – in practice and theory

I want to be clear – no Teslas were sacrificed in the writing of this piece but I did drive one. Fortunately the owner was there to give me an initial tour of the car. For the most part, the user interface is discoverable once you understand the idioms.

I adjusted the seat to my liking and saved it as the default user though I could've added my own personality. I expect that in the future that I could access my personal profile from the cloud and apply it to the car I am in. That might even happen automatically when I use my personal token.

Many cars associate seat settings with a particular key fob. In the future I expect that one's personality token will work across cars. This is already happening to some extent as smartphones become amulets that represent me but we need to think a bit more in the abstract. Instead of associating my identity with a particular piece of hardware I would login to the device I am using just like I can walk up to a kiosk and login to a site using a browser.

I already get some of this effect for phone calls by using Google Voice. I only give out my GV number and it will reach me on my current device be it a phone in my pocket or the Samsung Gear on my wrist. I wrote more about these relationships (bindings) in my October 2015 column.

Imagine renting a car and it will become yours in the sense of having your settings, be it the seat position or your entertainment choices. You shouldn't even have to go through the process of pairing your phone with the car -- it should "just happen". It should also just unhappen – how many people remember to unpair the phone when returning the car?

What is most striking to me about the Tesla driving experience, from a software perspective, is the center surface. I say surface not just screen because you can interact with it by touching. In the future we can expect other modes of interaction such as gestures. That may take a while as we develop common idioms just as are now familiar with standard menu items like copy and paste.

The cruise control was the most striking feature to me. Traffic on I-80 was stop and go and would've required constant attention in most cars. But having a cruise control that, in essence, followed the car in front of me and turned me into a passenger. I could relax though I still needed to pay attention to the road and steer the car. I also still had be careful when I changed lanes. But that is supposed to change soon with a software upgrade to handle lane changing.

The adaptive cruise control was set to leave a 4 second gap between me and the car ahead. That worked well in California but in Boston I'd expect cars to use that gap to slip in ahead of me. It's a reminder that driving is a social activity. An automated car will be gamed by other drivers who know that it will always back off in competition for space in the lane. This is one reason I welcome assistance in my driving but am reluctant to embrace the idea of totally automating driving.

(I can fantasize about my car having its own drone flying ahead to determine the optimal lane and competing with other drones … OK, back to reality).

While such capabilities are not at all unique to the Tesla there is far more of a sense that everything is considered a software upgrade though one might have to install new sensors for some features. Having screens instead of knobs, buttons and dials means that fewer of the capabilities are fixed function as in traditional cars. Instead of having to install a new radio I can just update the software or use my own device.

Even on cars with interactive screens most still have separate physical controls. It is going to take a while for the designers to take advantage of the new freedom and, perhaps, for buyers to accept the changes. Perhaps the Tesla example will encourage the transition to soft interfaces.

These capabilities touch upon the idea of a driverless car without having to have a car that is smarter than I am. I don't require automation. I want modest assistance although our expectations may increase as the technology becomes more capable.

It's nice to have a voice recognition system that worked fairly well. But I want more -- the car should be able to synchronize with my itinerary by having an API and a standard format for events. For that matter my alarm clock (of course, an app not a physical device) would also be able to know when and, just as important, why, to wake me up.

There's an app that tells me what charging stations are available. A great and simple improvement would be to transfer that information to the navigation/display application in the car. This can be done with relatively simple APIs that don't affect the safety and integrity of the car and driving.

Luggage – Lost and Found

I try to avoid checking luggage but this trip was an exception. We checked our shared bag when we arrived at the airport and could then relax. Arriving in Boston I waited for my luggage and it didn't show up. Finally, I spoke to the baggage people at United who told me that the luggage had been scanned in – an hour before my plane had arrived. Weird, until I realized it had taken the previous flight.

This is an opportunity for customer service and traveler satisfaction: United could have (or really, *should have*) notified me. I signed up to be told when my flight takes off and arrives. But there doesn't seem to be a way to get told or even query where my bags are. There is some irony in the fact that I wasted an hour trying to find my bags because the bags had arrived an hour before I did!

Here too, notification is a capability that needn't affect system integrity. It can and should be integrated as a basic part of the system accessible to users. If we think of it as an application then each such feature needs to be carefully engineered into the whole system. Adding a notification API should be simpler and that API could be used for the airline's own baggage management.

Such an API is a good practice for IT departments because it reduces the integrity perimeter by allowing the company to provide apps that are not tightly coupled to the main systems.

There is an alternative – my baggage can tell me where it is and, in fact, it soon will. I have two devices on order via Indiegogo. One is a simple tracker that uses the cellular phone system for determining approximate location and for sending me notifications. Unlike a Bluetooth beacon this device can work just about everywhere. In the future we will have open Wi-Fi connectivity available but in the meantime we can use the cellular phone system.

I also have a suitcase on order that will have such capability built in as well as a battery for charging devices. It was originally a Bluetooth device but it will also now have GPS and cellular connectivity. It will even weigh itself.

There are many ways to obtain and provide this tracking information thanks to the increasing availability of generic devices and open interfaces.

Loosely Connected

In 1989 I wrote *Rush Hour 1997* (http://rmf.vc/RushHour1997) in which I attempted to look ahead to a connected world. Today we are far along that path.

The consumer electronics world needs to adjust and expand with this new landscape. I shouldn't have to buy a new phone or, for that matter, a new car, just to get a new software capability.

It is going to take a while to come to terms with a world being redefined by software. This transition continues to be a challenge for the IEEE as problems are increasingly solved by empowered users and not just engineers.