Even in this connected age, it’s sometimes necessary to disconnect for a while. For example, you can’t roam to every square kilometer of the planet with any wireless technology currently available. So while networks are valuable, being off-network is a practical reality that must be dealt with—by both users and developers.

Several different behavioral models exist for handling off-network situations. In the case of cell phones, the network generally handles disconnections automatically. Incoming calls may be routed to a (network-side) voice mailbox instead of your unreachable handset. If the caller opts to leave a message, the network will let you know about that as soon as you are again within range.

Another off-network behavioral model operates like a digital camera, wherein the device is still useful, though perhaps in more limited ways. For example, you may only be able to photograph a few dozen images before running out of storage capacity in your digital camera. And you certainly won’t have access to the image manipulation software and color printing capabilities that make digital photos so much fun.

This month, the Internet Appliance Design section focuses on two technologies for adding new capabilities to embedded devices. The first of these is the Wireless Application Protocol. Despite its heavy association with Web-enabled cell phones, WAP has much broader potential applications. Specifically, its Wireless Markup Language—a stripped down version of HTML—could be used to develop interactive data content for any display-limited device. The scripting language WMLscript is also available to embedded developers for all sorts of applications.

Another useful connection technology is USB. One of the perceived problems with this PC-centric bus is the need to develop a number of operating system drivers (Windows 98, Windows 2000, and Linux to name just three) for each new device you might care to develop. However, that’s not always entirely true. Many embedded devices can get by with a bare bones amount of USB firmware and no new drivers at all—at least on modern Windows platforms. This opens the door to embedded devices not intended to be utilized primarily alongside PCs to include USB support anyway. For example, a piece of stereo equipment might have a USB connector and firmware tossed in, so that a user desiring to hook that product up to their computer could do so. (I can already hear the lawyers for the record labels grumbling at that specific thought.)

I hope you enjoy these Internet Appliance Design articles and the ones to come. I’m taking a few months away from writing this column. The decision to disconnect is a difficult one, but sometimes you have no other choice. Right now, I’m too busy with the additional responsibilities of being editor-in-chief. I do hope to return to this space sometime early next year.

Michael Barr is the editor in chief of Embedded Systems Programming. He holds BS and MS degrees in electrical engineering from the University of Maryland. He is the author of the book Programming Embedded Systems in C and C++ (O’Reilly & Associates). Michael can be reached via e-mail at mbarr@cmp.com.