Bluetooth interoperability:  
IT’S ALL IN THE DETAILS

Bluetooth faces interoperability issues on several fronts, all of which play a critical role in how quickly Bluetooth will penetrate the consumer market.

We certainly haven’t yet reached “The Year of Bluetooth,” but this fledging technology is quickly growing up. With projected shipments of 100 million chip sets this year, and double that next year, Bluetooth is beginning to bring revenue to the multitude of companies that have been hoping to bring it to reality (Figure 1). Mobile phones will clearly drive this market to start, but computing devices, such as PCs and PDAs, are expected to provide Bluetooth’s foothold in the world of embedded systems.

Given the general availability of Bluetooth radios, the market has shifted to the next level of maturity: interoperability. Three critical areas for Bluetooth are interoperability between Bluetooth devices from different vendors; interoperability or coexistence with other 2.4-GHz technologies, such as WiFi (Wireless Fidelity) and microwaves; and interoperability with users, or ease of use.

ARE YOU AN RF EXPERT?

A Bluetooth subsystem contains several layers that affect performance and pose interoperability issues. Some of the key layers are physical or radio, baseband, stack, profile, OS driver, and application. Each layer depends on the others, and each interface between layers holds the potential for ambiguity or error. For example, even though a chip set may be qualified from the radio through the profiles, an application may be able to break the stack if you, say, pass the wrong parameter with an out-of-range value. If the stack fails to range-test incoming parameters—perhaps in an effort to save processing cycles—the potential for a crash arises. Another example is sending a proprietary preamble with control data; the Bluetooth spec does not define what you can send before the preamble, and some stacks choke on this optional use of the protocol. Another area of vulnerability concerns security; different vendors may handle various optional issues, such as whether you can turn encryption on and off turning runtime, and control, such as how you manage the flow of data through the subsystem.

Bluetooth interoperability:  
IT’S ALL IN THE DETAILS

The Bluetooth networking starter kit from IOGear provides two Bluetooth-to-USB adapters/dongles, enabling consumers not only to connect non-Bluetooth desktops with Bluetooth devices, but also to network multiple desktops.

We certainly haven’t yet reached “The Year of Bluetooth,” but this fledging technology is quickly growing up. With projected shipments of 100 million chip sets this year, and double that next year, Bluetooth is beginning to bring revenue to the multitude of companies that have been hoping to bring it to reality (Figure 1). Mobile phones will clearly drive this market to start, but computing devices, such as PCs and PDAs, are expected to provide Bluetooth’s foothold in the world of embedded systems.

Given the general availability of Bluetooth radios, the market has shifted to the next level of maturity: interoperability. Three critical areas for Bluetooth are interoperability between Bluetooth devices from different vendors; interoperability or coexistence with other 2.4-GHz technologies, such as WiFi (Wireless Fidelity) and microwaves; and interoperability with users, or ease of use.

ARE YOU AN RF EXPERT?

A Bluetooth subsystem contains several layers that affect performance and pose interoperability issues. Some of the key layers are physical or radio, baseband, stack, profile, OS driver, and application. Each layer depends on the others, and each interface between layers holds the potential for ambiguity or error. For example, even though a chip set may be qualified from the radio through the profiles, an application may be able to break the stack if you, say, pass the wrong parameter with an out-of-range value. If the stack fails to range-test incoming parameters—perhaps in an effort to save processing cycles—the potential for a crash arises. Another example is sending a proprietary preamble with control data; the Bluetooth spec does not define what you can send before the preamble, and some stacks choke on this optional use of the protocol. Another area of vulnerability concerns security; different vendors may handle various optional issues, such as whether you can turn encryption on and off turning runtime, and control, such as how you manage the flow of data through the subsystem.
Unfortunately, an application could crash a stack through no fault of the stack. For example, without an MMU (memory-management device), an errant pointer could corrupt the stack. Therefore, you may have to dig into someone else’s code to determine which layer of the subsystem is at fault.

In reality, you’re probably going to have to dig into code anyway, if not to find the source of a problem, then at least to identify potential areas to test whether the code will break. You’re also going to need some understanding of radio signals if you want to determine the threshold of operation for your device. In other words, if you care about the difference between merely building a working device and building a qualifiable and interoperable robust device, you need a rough understanding of what’s happening at each layer. The shape and even the color of the plastic that houses your device may affect radio sensitivity, and, if you know nothing about radios, you won’t know what to look for when there’s trouble. You may buy certified components, but they may not stay certified if you don’t know how the rest of your design can affect the radio.

You aren’t completely alone on RF issues, though. Most chip-set vendors employ application engineers who can walk you from basic design to manufacturing test. If you do need to solve an RF problem, you need some kind of sniffer at either the physical-signal or the protocol layer to capture the signal. You can then send the signal to an RF expert for analysis. The ability to capture signals in the field is important if you plan to ship to multiple countries whose RF environments can vary greatly.

**PLAYING WELL TOGETHER**

Before you release your product, you need to test it with other Bluetooth devices. Although you can’t practically test your device against every device on the market, you can buy (or trade with other vendors you want to interoperate with) a few that your device should work with. However, testing only a single headset isn’t enough.

At the official “unplugfests” that the Bluetooth SIG (Special Interest Group) sponsors, vendors usually play nice with each other. The environment is clean and controlled. But consumers aren’t nice, and they live in noise. You need to go beyond nice to figure out where your device or the design of the devices you want to interoperate with start to wobble. You can obtain this information by changing operating conditions to determine the threshold of interoperability.

For example, alter various parameters, such as sniff intervals and connection times, to see whether you can crash either radio. Change operating conditions by putting devices into deep sleep or park and see how well they recover. Test the range at which devices begin to stop connecting reliably. Check whether your device disconnects gracefully when it moves far away or locks up. Try walking away during a long file transfer and see whether your device can reconnect when you step back into range.

You need to anticipate what users will do to your device. Remember that, in general, most engineers treat their devices as they “should.” Real users, however, push several keys at once, repeatedly push the same key when nothing seems to be happening, and whack the device on a table if there’s still no response.

Recognize that you will at some point want to qualify your device with the BQB (Bluetooth Qualification Body). The Prequal tool from Bluetooth vendor IAR lets you test your device from a software perspective and run the various BQB tests yourself to expose any potential problems. You’ll even be able to use the test results that this tool generates as part of your submission for BQB qualification.

Prequal targets engineers with little familiarity with Bluetooth. Note, however, Prequal flags only the location an error may have occurred or the API returned the wrong value. It doesn’t tell you how to fix the problem. You still have to interpret the results, which means that you need some understanding of the spec (see sidebar “Choose your tests wisely” on the Web version of this article).

Many 1.0 devices have been grandfathered, as many 1.1 devices will be when 1.2 is available, meaning that manufacturers are not required to requalify these devices to the new spec. For your device to interoperate with grandfathered devices, you may need to implement a work-around.

Many of the chip-set and stack vendors are on top of the grandfathering issue, discovering points of ambiguity in the spec and implementing the necessary work-arounds to keep their devices alive. If you discover that a key device with which you want to interoperate does something quirky or violates the spec, you also need a work-around: vendors are wary of adjusting software for successful and shipping products. If you don’t write your own stack or profiles, you need a solid relationship with a vendor that can and does implement these work-arounds for you. Note that stack or driver changes may require requalification (see sidebar “Device design” on the Web version of this article).

**COEXISTENCE**

Most vendors agree that WiFi and Bluetooth do not directly compete with each other on a market level. However, given that they share the 2.4-GHz spectrum, they can contend with each other, as well as with other 2.4-GHz technologies, including Zigbee and microwave ovens.

“Coexistence” refers to the idea that WiFi and Bluetooth devices recognize that other technologies may be in use in the same proximity. Given that WiFi has a foothold in the US market, it’s not unreasonable for the burden of coexistence to fall on newer competitor Bluetooth, even though WiFi suffers more from contention. WiFi tends to send longer packets, which you need to resend when there is contention. Additionally, WiFi nodes randomly back off after contention and

**AT A GLANCE**

- You need not be an RF or Bluetooth expert to design a working Bluetooth device. However, you need expertise if you want to build a qualifiable and interoperable robust device.
- The Bluetooth spec allows you to be clever but doesn’t prevent you from being careless.
- Bluetooth devices must employ coexistence strategies if they don’t want to overwhelm or be overwhelmed by other wireless devices using alternative technologies in the same spectrum.
- The biggest challenge for Bluetooth is educating consumers about device complexity and profile interoperability.
use fewer frequencies, further reducing their effective bandwidth. However, in countries in which WiFi has little penetration, making Bluetooth bear the entire burden is a less reasonable proposition.

Coexistence is a key issue for Bluetooth’s success. It depends on how you manage radios or how good a neighbor they are to other devices. The spectrum is a shared resource. Certainly, devices can transmit with distortion and more power to make sure they get heard. However, such signals may blast out other signals, reducing the performance of the PAN (personal-area network) as a whole. Good-neighbor devices dynamically adjust power as low as possible, both to prolong battery life and to promote coexistence with other devices.

Bluetooth vendors have begun to cooperate to create a coexistence standard; the new version of the spec, Version 1.2, which is due late this year or early next year, will attempt to eliminate the rush to create proprietary coexistence mechanisms, leading to further market confusion as consumers try to pair devices using different schemes and focus the market on a few standard techniques.

Coexistence mechanisms break down into noncollaborative and collaborative types. Noncollaborative mechanisms are used when WiFi and Bluetooth are independent networks with no means of communicating with each other. (In other words, one knows the other is present only when their signals cross.) A simple method for reducing contention is channel skipping, or deferred transmission. A Bluetooth device using deferred transmission avoids transmitting over parts of the spectrum in which it suspects WiFi is also transmitting. Such a scheme is straightforward to implement, and a single master can manage channel skipping for a piconet of slaves. However, this technique sacrifices Bluetooth bandwidth for the sake of WiFi. If you use the Bluetooth link for latency-bound data, such as voice or audio, such a scheme may provide unsatisfactory results.

An alternative scheme is AFH (adaptive frequency hopping), in which the Bluetooth device maps frequencies that encounter interference to new, clear frequencies, thus preserving bandwidth. The challenge this scheme poses is that you have to change the hopping algorithm to mask out the “bad” frequencies. You also have to sense WiFi’s location so that you can skip around it and avoid being fooled into thinking another form of interference is a WiFi network. Sensing uses power and bandwidth that you could have used to send data. Various vendors have implemented AFH schemes, but the schemes differ in how often they profile the spectrum and exchange information on alternative frequencies (see “Upgrading to Bluetooth AFH” on pg 20).

Pressure to solve the collocation problem—WiFi and Bluetooth in the same device—comes primarily from laptop vendors that offer WiFi and want to add Bluetooth without changing their current offerings. Therefore, Bluetooth must play nicely within their systems. If the WiFi and the Bluetooth subsystems are in different parts of a laptop, for example, AFH may be sufficient. However, when devices are close—on the same PCMCIA card, for example—AFH may not be enough. A strong Bluetooth signal in the ISM (industrial, scientific, and medical) band close to but outside the WiFi band passing through the WiFi wideband filter can cause front-end overload when the low-noise amplifier boosts the signal to get it through the IF stages. The components that measure the amount of power in the band, however, do so after the IF stages, yielding a distorted signal for which you cannot tell that the low-noise amplifier was responsible.

Devices in close proximity must employ a TDM (time-division-multiplexing) scheme, in which WiFi and Bluetooth take turns sending data. Instead of guessing when the other radio is sending, one radio could inform the other when it wants to transmit to maximize the available bandwidth. However, getting the networks to cooperate with each other is a challenge. The Bluetooth baseband and WiFi MAC (media-access controller) must exchange information about their actions through a real-time-signaling channel. For laptops, however, given that Microsoft is not expected to support the passing of channel information between WiFi and Bluetooth when it adds Bluetooth support with only limited profiles to Windows XP, you’ll need either a hardware trace between chips or special drivers that post signaling information in a “public” place. Hardware traces, for their part, are extremely proprietary, and special drivers could be a nightmare considering the number of possible driver combinations.

Having both technologies residing on the same chip does solve the collocation problem. Mobilian, for example, offers the TrueRadio chip set, which can send Bluetooth while receiving WiFi and vice versa, using active cancellation. Subtracting the transmitted signal from the received signal allows Bluetooth and WiFi to share the same antenna.

In the meantime, both the WiFi and the Bluetooth standards groups are developing coexistence standards. At press time, the 802.15 Task Group 2 was scheduled to meet to discuss issues of coexistence for wireless PANs. Visit grouper.ieee.org/groups/802/11/ for more details.

YOU REAP WHAT YOU SOW

Everyone knows that LANs (local-area networks) are difficult to set up, and wireless LANs are even more difficult, so when it takes you a few hours to set up a WiFi network, you aren’t surprised. Plugging in cables, on the other hand, is easy and a task we’re all familiar with.

Few tasks are easier than plugging in a printer, but setting up a Bluetooth network is not among them. This statement...
isn’t to say that Bluetooth won’t be better than cables, but it adds the complexity of wireless network to the equation. Environmental noise might prevent devices from pairing easily. Bluetooth portable devices have an extra battery—headsets have separate batteries from cell phones—that can drain and kill a connection. Bluetooth also adds complexity to security. With cables, devices are always paired, and you can send power to a device so that you never have a dead battery. Also, cables are inherently more secure than transmissions, and they are completely transparent to applications; Bluetooth currently is not.

The Bluetooth SIG has done an excellent job of promoting how easy Bluetooth will be to use as a cable-replacement technology. Users have a high level of expectation, so devices need to be idiot-proof. As most engineers know, it is easier to solve a tough technical problem than it is to create a device that no one can intentionally misuse. However, engineers need to design devices that will meet the expectations that all the hype generates. Creating easy-to-use Bluetooth devices poses several challenges, including installation, profile interoperability, matching and discovery of devices, transparency of Bluetooth to applications, and security.

**INSTALLATION**

The Bluetooth SIG has tried to address usability issues through its “5-Minute Ready” program. For example, devices need to avoid startling or confusing messages, such as, “It is now safe to turn off your computer,” that imply problems where there are none or required actions when none are necessary. In most cases, most of the five minutes required to connect a Bluetooth device involve loading the appropriate drivers, because Microsoft Windows doesn’t yet support Bluetooth.

Some devices appear to work without drivers because they aren’t native Bluetooth devices but actually USB dongles. The Bluetooth mouse from Logitech, for example, appears as an HID (human-interface device) and configures itself when you plug it in. The mouse uses Bluetooth

---

**FOR MORE INFORMATION...**

For more information on products such as those discussed in this article, go to www.edn.com/info and enter the reader service number. When you contact any of the following manufacturers directly, please let them know you read about their products in EDN.

- **Accelerated Technologies**
  - www.accelerated technologies.com
  - Enter No. 320
- **Agere Systems**
  - www.agere.com
  - Enter No. 321
- **Agilent**
  - www.agilent.com
  - Enter No. 322
- **Alcatel**
  - www.alcatel.com
  - Enter No. 323
- **Analog Devices**
  - www.analogdevices.com
  - Enter No. 324
- **Anritsu**
  - www.anritsu.com
  - Enter No. 325
- **Atinav**
  - www.atinav.com
  - Enter No. 326
- **Bandwidths**
  - www.bandwidths.com
  - Enter No. 327
- **Brain Boxes**
  - www.brainboxes.com
  - Enter No. 328
- **Brightcom**
  - www.brightcom.com
  - Enter No. 329
- **Broadcom**
  - www.broadcom.com
  - Enter No. 330
- **CATC (Computer Access Technology Corp)**
  - www.catl.com
  - Enter No. 331
- **Cetecom**
  - www.cetecom.com
  - Enter No. 332
- **Conexant**
  - www.conexant.com
  - Enter No. 333
- **CSM (Cambridge Silicon Radio)**
  - www.csm.com
  - Enter No. 334
- **Cypress**
  - www.cypress.com
  - Enter No. 335
- **Datastic Systems**
  - www.datastic.com
  - Enter No. 336
- **Etenna**
  - www.etenna.com
  - Enter No. 337
- **Ericsson**
  - www.ericsson.com/bluetooth
  - Enter No. 338
- **Extended Systems**
  - www.extendedsystems.com
  - Enter No. 339
- **gigaAnt**
  - www.gigaant.com
  - Enter No. 340
- **KC Technology**
  - www.kctechnologyinc.com
  - Enter No. 341
- **IAR Systems**
  - www.iar.com
  - Enter No. 342
- **Inphene**
  - www.inphene.com
  - Enter No. 343
- **I/O Gear**
  - www.iogear.com
  - Enter No. 344
- **IVT Corp**
  - www.ivtcorporation.com
  - Enter No. 345
- **Mezoe**
  - www.mezoe.com
  - Enter No. 346
- **Microsoft**
  - www.microsoft.com
  - Enter No. 347
- **Mitsumi Electronics**
  - www.mitsumi.com
  - Enter No. 348
- **Mobilian**
  - www.mobilian.com
  - Enter No. 349
- **Motorola**
  - www.motorola.com
  - Enter No. 350
- **National Semiconductor**
  - www.nsc.com
  - Enter No. 351
- **NewLogic**
  - www.newlogic.com
  - Enter No. 352
- **Oki Semiconductor**
  - www.okisemi.com
  - Enter No. 353
- **Parthus**
  - www.parthus.com
  - Enter No. 354
- **Philips**
  - www.philips.com
  - Enter No. 355
- **RF Micro Devices**
  - www.rfmd.com
  - Enter No. 356
- **Silicon Wave**
  - www.siliconwave.com
  - Enter No. 357
- **Smart Modular Technologies**
  - www.smartm.com
  - Enter No. 358
- **Socket**
  - www.socketcom.com
  - Enter No. 359
- **STMicroelectronics**
  - www.st.com
  - Enter No. 360
- **TDM Systems**
  - www.tdmsystems.com
  - Enter No. 361
- **Taiyo Yuden**
  - www.taiyo-yuden.com
  - Enter No. 362
- **Texas Instruments**
  - www.ti.com
  - Enter No. 363
- **Toshiba**
  - www.toshiba.com
  - Enter No. 364
- **Ubicom**
  - www.ubicom.com
  - Enter No. 365
- **Widcomm**
  - www.widcomm.com
  - Enter No. 366
- **Xemics**
  - www.xemics.com
  - Enter No. 367
- **Yokogawa**
  - www.yokogawa.com/tm/BuyBluetooth/EnterNo.372
- **Resources:**
  - Bluetooth SIG (Special Interest Group)
  - www.bluetooth.com
  - BlueUnplugged.com
  - Web store and resources
  - www.blueunplugged.com
  - ExpandSys
  - www.expandsys.com
  - PaloWireless Wireless Resource Center
  - www.palowireless.com
  - TDK Wireless Interoperability
  - www.no2wires.com

**SUPER INFO NUMBER**

For more information on the products available from all of the vendors listed in this box, enter No. 373 at www.edn.com/info.
to talk to the dongle, which passes data over the USB. The disadvantage of such a configuration is that the dongle becomes a point-to-point device that can connect only with the mouse. There may also be pairing issues, such as two similarly dedicated connection devices in the same proximity—for example, his and her computers in the same room or on opposites sides of the same wall.

Profiles

Profiles are perhaps the most concerning aspect of Bluetooth interoperability because they add a dimension of complexity. Bluetooth devices can connect to each other only if they have the right profile match. This requirement is a problem, because devices need not support all profiles. In some cases, the fact that two devices can’t pair makes sense. For example, it’s difficult to conceive of a reason that someone would try to pair an earpiece with a printer. However, in many cases, determining whether two devices should pair is more ambiguous. For example, many users may believe they can use the earpieces from their cell phones to listen to music on their laptops or PDAs, and they could spend hours trying in vain to connect these devices. In another example, if an OEM designing a cell phone chooses not to support the OBEX (object-exchange) profile for cost-savings reasons, users will be unable to send business cards to the phone.

Pairing scenarios quickly become complex. Is it possible to pair two earpieces with one phone? Can you pair one earpiece with two phones? If you can, which phone makes the call when both are close to the earpiece? In such cases, you can use a priority (dial phone A first, if possible), but doing so adds complexity to the installation process. Determine how much complexity you can expect users to understand and handle.

Another aspect of profiles involves where the brains of a transaction reside. For example, a PC usually processes documents and images before sending them to a printer, which is one reason that printers can be so inexpensive. However, if you want to be able to hold your Bluetooth camera within proximity of a Bluetooth printer and print an image, it’s unclear which device should process the information for printing. It is possible to require a PC in the middle to handle all the transactions, but doing so eliminates peer-to-peer connections, one of the main selling points of Bluetooth.

Bluetooth supports the BIP (basic-image-printing) profile, but it allows little or no image manipulation. If you want to crop the image or remove redeye, you need another level of processing from the camera, the printer, the dongle, or all three. Vendors don’t yet agree on where smarts and horsepower should reside, so problems will arise when a user tries to print an image from an inexpensive Bluetooth camera to an inexpensive Bluetooth printer. The reason for failure probably won’t be easily discernible to users. This problem also exists for audio. Different profiles exist for different sound quality. An 8-kbps-only earpiece for cell phones won’t work with an MP3 player.

Fortunately, as more chip vendors step back from using their own stacks to using the same stacks and profiles as other vendors, interoperability between stacks and profiles should progress. It should become easier to include more profiles when they are all part of a library than when you have to purchase or write them individually.

As part of the 5-minute Ready program, the Bluetooth SIG has also committed to redesigning www.bluetooth.com with consumer education as the site’s primary purpose, offering publication of consumer-use cases and potential product pairing, a searchable consumer-product database, a troubleshooting guide with tips to enhance the experience, and a glossary of common Bluetooth terms.

Listing pairable devices on Web sites, however, is still a far cry from addressing users’ confusion regarding devices that they might reasonably expect to work together but are incompatible. The use of profiles and their optional inclusion with each device will probably be the primary cause of interoperability problems and diminished user experience (see sidebar “Easing setup and connection” on the Web version of this article).

Pairing

Once two devices discover each other and pair to form a connection, Bluetooth can become transparent to the user. However, easily pairing devices is not a given. Bluetooth has found its way into a variety of device types, some of which resemble PCs and some of which have no screens. Thus, no standard manner exists in which to connect Bluetooth devices, and each device may handle connection differently. For example, one vendor may require you to push a button on an earpiece to place it in discoverable mode; another might expect you to flip a switch or hold down a button during the entire process. Additionally, each cell phone may have a different series of menus that you have to navigate to begin pairing. Users might also run into issues if multiple earpieces or cell phones are present during pairing.

Because wireless technology is so young, no culture for it yet exists. No one can know how users want to interact with wireless devices until users have them. The transition resembles the move from command-line interfaces to GUIs. Users will have to weather this phase as the culture evolves. The key to this process is not making users feel stupid.

To connect to other devices, Bluetooth devices must first discover each other. Discovery can be active, such as when the device broadcasts that it is present, or passive, such as when the device listens to hear whether other devices have moved within proximity. Active discovery requires more power than does passive discovery; the device must broadcast. Thus, it makes sense to seek other devices, such as printers, only when you need them. Additionally, because you usually plug in devices such as printers, these devices can afford to bear the cost of broadcasting on a regular basis to discover other devices.

Another level of complication involves device class. For example, many printers are Class 3 devices, meaning they have a range of 10m. Thus, a Class 2 device, with a range of 20m may be able to see the printer without the printer’s being able
to see it. This situation could be useful if no response were necessary from the Class 3 device, such as with an open device that accepts communication from any device and does not need to pair to receive data. However, it presents issues for users. For example, the user of a Class 2 device may wonder why a printer that is within range of his or her device, is unavailable.

**TRANSPARENCY**

For Bluetooth to succeed, it must become a transparent technology. For example, you don’t care whether you connect your PC to your printer using parallel or USB technology. Users want to access applications, not interconnects. They don’t want to know whether the device is native Bluetooth or working through a USB dongle.

In this light, instead of a user’s clicking a button to discover Bluetooth devices, discovery should be an integral part of the application, similar to how you can download e-mail regardless of your network connection. Devices can reconnect with previously paired devices without the user’s participation.

To complicate matters, because many Bluetooth devices are mobile, paired devices can move out of proximity of each other or experience serious interference during data exchanges. Applications must also be able to tolerate the idiosyncrasies of the wireless environment; for example, they can’t assume that the connection has been broken as they can when a cable is disconnected. Ideally, the user should not need to know whether a connection is a Bluetooth connection to use it.

**SECURITY**

Wireless connections differ greatly from cabled connections. For example, other devices can listen in on a connection. In a classic example, a vendor with a hotel room adjoining his competitor’s can log onto that competitor’s Bluetooth network. Encryption is available for Bluetooth, but many devices turn it off or default to not using it to conserve bandwidth and power. Given the push to reduce the complexity of creating a connection, many users may be unaware that their data is exposed.

This is one level at which an application may need to be aware of the Bluetooth connection to determine which data to encrypt and which to keep as clear text.

Another area of concern is device pairing. To make pairing an earpiece easier, a cell-phone manufacturer might consider automatically pairing it and the earpiece. However, if several earpieces are within a phone’s proximity, the phone must be able to determine which earpiece is yours. Otherwise, a stranger could make calls on your phone.

Most chip vendors have decided that the hands-free market is the killer app for Bluetooth, because they perceive that this market holds the highest volumes. However, Bluetooth suits a variety of applications (see sidebar “Sensor PANs put data collection in your hand” on the Web version of this article). These niche markets may determine the long-term success of Bluetooth, but they alone won’t drive the volumes necessary to keep Bluetooth under its $5 goal.

Although it’s arguable whether Bluetooth has meet the $5 target its developers originally set for the technology, Bluetooth radios are now available off the shelf; you need not be an RF expert to design a Bluetooth-enabled device. Legend has it that the $5 figure was derived from the $10 cable Bluetooth aimed to replace—$5 for each side of the cable. Today, you can buy chip sets for less than $4 in volume, but they include only software up to the HCI (host-control interface), meaning that you still have to consider the cost of the Bluetooth protocol stack, as well as discretes and an antenna. Note that prices for Bluetooth stacks, as well as various profiles, are elusive unless you have a purchase order in hand. For example, STMicroelectronics charges a 30-cent royalty for its HID profile.

Hitting $5 dollars on the OEM balance sheet—never mind bringing the price to a consumer $5—is a challenge. (The cable costs the consumer—not the OEM—$10.) In any case, prices are low enough that many Bluetooth chip-set vendors claim that handset vendors are including Bluetooth on most if not all of their upcoming offerings. Given the cost-competitiveness of the handset industry, accepting this claim may require a bit of faith.
One PAN (personal-area-network) application suited for Bluetooth is the handheld-sensor data-collection market. Using this application, a person walks through a network of wireless sensors and views current data using a handheld device, rather than going back to a control room or relying on a confederate sitting in that control room with access to the master monitoring system. Datastick develops software to enable Palm PDAs to replace expensive proprietary handheld devices (Figure A).

Using Bluetooth in the sensor market poses several challenges. First, the Bluetooth spec contains no sensor profile. Second, Bluetooth processing is difficult to implement in simple devices, such as temperature or pressure sensors. Third, not everyone wants standards in the industrial-automation and -control markets. Control data is a trade secret, and people perceive wireless transfers as insufficiently secure.

To solve these problems, vendors such as Datastick have focused not on changing sensors but on developing a PAN that works with the devices. Most sensors project data over a serial port. Bluetooth supports a serial-port profile. By attaching a serial-port Bluetooth dongle to sensors, you can now wirelessly send data. Using encryption that the dongle supports, you can also securely send data.

One concern with using serial-port emulation instead of native Bluetooth connections is that emulated transfers can take two to 10 times as long as native transfers, because emulation of serial-port pins eats up bandwidth. The firmware of the sensor also affects throughput; because it is unaware of the technology it is using, it may misuse bandwidth. With emulation, the connection is running but is compromised. Does the lost bandwidth matter? It depends upon how many measurements you take.

Pairing devices becomes a much different problem when you're dealing with a large network of devices. When you have 50 temperature devices, how do you know whether temp53477 or temp59972 is the one you want? For such a network, you must create a database of names, so that you can reference devices by name (intake_valve_temp, for example) rather than by device ID, as is typical of Bluetooth.

One disadvantage of serial emulation is that it can take several seconds to establish a connection with a device. Connection takes less than a second with a native Bluetooth link. With so many devices, managing discovery poses a challenge. Should the PDA ping the user with a proximity alert every time he walks within range of a sensor? That pinging could become annoying, especially in noisy environments in which links often drop.

Perhaps the key advantage of a native link is that, through connection speed and aware firmware, it allows the data collector to collect data from sensors without making the user aware of that connection. For example, a user could walk through a plant, and the PDA would sound an alert when a sensor exceeded a set threshold, enabling the user to collect and evaluate data without having to proactively query each sensor.
CHOOSING YOUR TESTS WISELY

Many products exceed the ~70-dBm sensitivity that the Bluetooth spec requires. You can maintain a tight design so that the extra decibels, referred to 1 mW, result in better range than advertised; give you wiggle room in the physical environment, such as going through walls or handling noise; or compensate for design errors, such as routing signal lines over the antenna. (If you know nothing about radios, consult with your vendor of choice for a list of RF no-nos).

Developing tests for manufacturing is critical. You don’t want a person to have to participate in the testing process, so you need to provide an internal mechanism to automate testing. Additionally, consider that, on assembly lines, the difference between a five-second test and a one-minute test makes a tremendous difference. You need to understand enough about the radio you purchase to pick tests that ferret out faulty devices, not those that merely exercise functions.

Your chip-set and stack vendors should be familiar with how to test Bluetooth radios. If you aren’t an RF expert, ask about RF-testing resources. Get details in writing of the vendor’s commitment to you for developing RF test for design and manufacturing. Consider the more than 600 tests for qualification only as a starting point. Because exhaustive testing is expensive, ask vendors to identify troublesome or high-value test points.

The receiver subsystem will be more difficult to build than the transmitter subsystem because you don’t know what you’ll get as a signal.

Sometimes, less is more. The Bluetooth protocol analyzer from CATC allows you to expand or collapse the Bluetooth stack to the level you want to observe. In other words, you don’t see a stream of bits; you see an interpretation of those bits as they fit into the Bluetooth protocol.

Note that many of the testing tools can evaluate how a device runs with interference but do not themselves inject interference. You may need to create your own tool to exercise this aspect of performance. You also need to consider the C/I (carrier-to-interference) ratio, or the ability of the radio to extract information from a noise-filled environment.

DEVICE DESIGN

When testing your device’s effective data rate, use a file transfer. Doing so provides you with a more accurate measure of true bandwidth than short transactions with lots of handshaking and overhead. Also, use a local file, not one over a network, to avoid adding unrelated latency.

Consider the amount of data your device needs to send. For example, a mouse could send a great deal of data that yields relatively little value. Consuming bandwidth to incrementally improve performance reduces the overall effectiveness of the PAN when multiple units are present.

If your device connects to a computing device, such as a PC or a PDA, the drivers that come with the Bluetooth chip set consume CPU cycles. Some chip sets can sell for less because they push more of the protocol processing onto the computing platform, possibly slowing the applications running on the computing device and resulting in a diminished user experience.

For the sake of being a good neighbor, develop or use drivers that users can completely remove when they no longer need them. For example, you may insert a dongle on your friend’s computer to create a temporary PAN. When you remove the dongle, however, leftover drivers or DLLs may affect the future performance of the computer if you haven’t been careful.

Most reference-design BOMs (bills of material) are inaccurate. Some omit a critical component, such as an antenna or flash memory, so make sure you account for the entire subsystem. Also ask for a price breakdown by component; your pricing for commodity parts may differ from the vendor’s.

Version 1.1 of the Bluetooth specification does not support real-time data. Quality of service for audio and video will come with 1.2, which is in draft form and is slated for ratification this year. If you’re working with early drivers, you need a mechanism for updating devices in the field.

It’s not all about cost. When you buy a chip set, you also buy software and signal management. A vendor that invests in improving the robustness of drivers and stacks, as well as adding work-arounds, may charge more than a vendor that leaves you to your own devices.

Consider the complexity of the various APIs with which you work. Although a more complex API has a longer learning curve, it provides access to internal variables necessary for hunting down a problem. However, to use this information, you have to sort through the extreme amounts of data to generate and understand which parts are important.

Some vendors offer hardware samples, even though their firmware is still incomplete. Determine the impact these samples or incomplete firmware will have on your ability to start designing, as well as any delays to your time to market.
EASING SETUP AND CONNECTION

To measure how easy your device is to use, get someone other than the designing engineers to try the device, using only consumer-level documentation. And make sure that the learning curve for the device is clear to users. For example, a commercial could show a user removing a Bluetooth device from a box and then using it with a PDA. Even live demonstrations give an unrealistic view of user expectations, because the demonstrator has used the device hundreds of times. Time a new user employing the device for the first time. Then, time the second through 50th times. Finally, time an expert who has used the device 100 times. Too often, marketing provides only the expert time, making users feel stupid or that the product is broken and possibly returning it if they achieve different results.

Documentation needs to be easy to understand and heavily illustrated. Better yet, setting up a Bluetooth connection should be so straightforward that it requires no manual at all.

Not every Bluetooth device has a screen. Even devices with screens pair differently from each other. Thus, knowing how to connect two Bluetooth devices doesn’t necessarily mean you can connect other kinds of devices. Bluetooth hasn’t been around long enough for pairing to be as intuitive as clicking on the x in the upper right corner to close a window has become. Loading device drivers is sometimes difficult. Enterprise users may discover that it has locked out their ability to add applications; they can plug in the device but can’t update the necessary Windows drivers. The device may run at a reduced bandwidth if at all, putting your product in a poor light. Over-automating the updating process might mean clever users cannot work around OS-update problems.

At some point, vendors hope, Bluetooth will be a native part of Windows, and you’ll be able to skip the driver update. However, there will for some time be legacy machines that do not natively support Bluetooth. You still need to consider these devices if you want to sell to these markets. Consider the average lifetime of a corporate computer.

Consider selling both sides of a Bluetooth connection in the same box. First, users can bypass the frustrating experience of discovering the sound of one radio ping. Additionally, given the potential pairing problems with different vendor devices, you ensure that at least your part of the PAN (personal-area network) operates.

Consider bundling compatible devices, such as a camera and a printer. You eliminate the uncertainty of whether devices can talk to each other. You can also pair devices beforehand for an out-of-the-box experience. (Note: Pairing devices before consumers buy them may mean that consumers can’t use a camera, for example, with a printer other than the one the manufacturer paired it with at the factory.)

If discovery can be so complex, why not just make it automatic? Consider the case of a cell phone that automatically pairs with headsets. A stranger standing close to the phone could conceivably pair the headset with the phone and start making calls with it. To prevent unauthorized pairing, you need to add some element of user acceptance.

Each Bluetooth device has a 48-bit ID. Thus, once you make a pairing, you can keep the connection available even when the devices are out of each other’s range. One issue worth considering, however, is the spoofing of IDs. If a sniffer can pick up an ID during a connection, it could use that ID to spoof the device in question.

Even if a printer supports a camera, the user might need to load a printer driver to the camera to get best results. You need to provide a mechanism for notifying the user of this need and the means to download. You’ll also need to leave enough headroom for the driver itself. Or is it the other way around? Does the camera support a particular printer? How is the user, especially one who won’t crack open the manual because Bluetooth is so easy to use, supposed to know?

Remember that many first-experience issues will arise. For example, you may want to print to a printer using your PDA over Bluetooth, but you don’t have the right printer driver loaded. Certainly you can get the driver off the Web for subsequent attempts, but your first attempt has been thwarted. Many evangelical Bluetooth moments, in which an early adopter shows someone how useful and easy Bluetooth is to use, will be first experiences.

Provide a mechanism in your device that explains to a user that the device is incompatible with another device the user is trying to connect to. Otherwise, you might find yourself dealing with a lot of unnecessary product returns.

One way to reduce the complexity of pairing is to have the device clearly state what it can do. Instead of confusing users with the profiles that the device supports, you could present the information in terms of function: “Your device allows you to send files to a printer and audio to an earpiece.” Consider pairing devices only when the user needs to use that function. For example, when the user wants to print, ask whether the user wants to pair at that time. Offer the printers with which the device is currently paired with an option to find more. (Perhaps show only printers that are currently within proximity and “gray out” devices unavailable for use.) Bring pairing to a yes-no-or-later level of complexity. One important limitation of this technique, however, is that it clearly defines how a user can pair devices, making it difficult for users to discover ways to use your device that you didn’t consider.