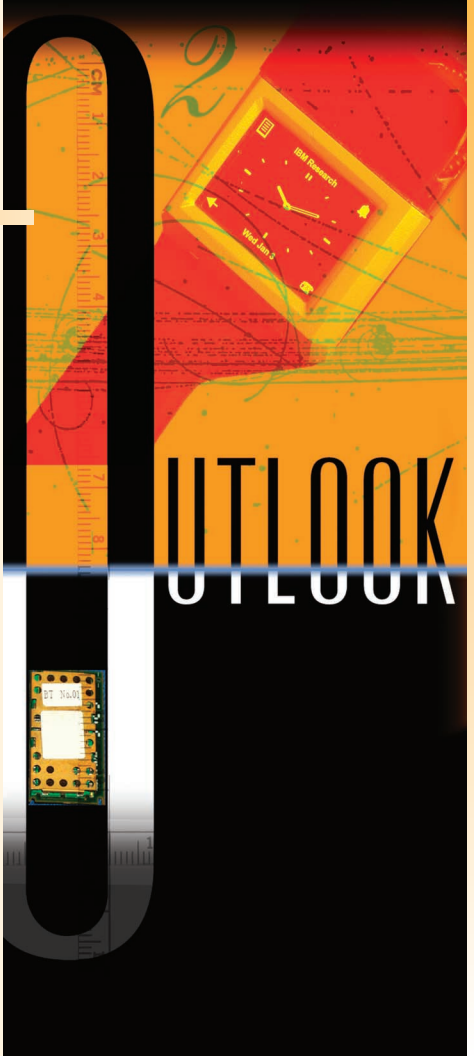
IBM's Linux Watch: The Challenge of Miniaturization



IBM's wristwatch computer looks like an ordinary watch but runs Linux, features X11 graphics, and offers Bluetooth wireless connectivity.

Chandra Narayanaswami, Noboru Kamijoh, Mandayam Raghunath, Tadanobu Inoue, Thomas Cipolla, Jim Sanford, and Eugene Schlig
IBM Research

Sreekrishnan Venkiteswaran, Dinakar Guniguntala, and Vishal Kulkarni
IBM Software Laboratory, India

Kazuhiko Yamazaki
IBM Japan

A businessman carrying a garment bag and briefcase hurries down a Chicago street, looking at his watch. An ordinary event, except that the watch is giving him a traffic report instead of the time.

Noting an accident on the main route to the airport, the watch displays alternative routes, but none will get the traveler there in time for his 10:30 a.m. flight. Touching a corner of the watch face, the traveler brings up his agenda. A roll of the tiny wheel moves the "flight home" entry to 11:30 a.m. and dispatches an e-mail to his secretary in St. Louis.

A few blocks later, the watch vibrates, indicating a new message—e-mail from his son: "Dad, don't forget my Chicago Cubs cap." He also has a message from his secretary, who has gotten him a seat on the 12:20 p.m. flight and bumped his afternoon meeting with the marketing director to 4:00 p.m. Speaking into the watch, he sends a message to his wife's cell phone: "Flight delay. Can't pick up Rob after baseball."

At the airport newsstand, the watch vibrates again, this time with a text message from his wife saying that their son has a ride home with a friend. He heads for the gate with the Cubs cap in his briefcase, thinking how complicated life would be without his watch.

The businessman and his family are neither spies nor actors, but ordinary people—and scenarios like this one have been foundational in our vision of a communications device that would change lives. For nearly four years, we have seen this vision evolving as the IBM Linux watch, a complete computer system that runs Linux, displays X11 graphics, and has wireless connectivity.

Figure 1 shows two versions of this watch, one with an organic light-emitting diode (OLED) display and the other with a liquid crystal display (LCD). The watch fits in a case that could pass as a slightly unusual analog timepiece with a somewhat odd shape and an extraordinarily brilliant face.

The watch interface has a touch screen with symbols in the four corners signifying corresponding actions. Touching the top left quadrant takes the user to a start menu; the bottom left is a back button similar to the one found on Web browsers; the top right is a shortcut to setting alarms; the bottom right is a shortcut to a list of phone numbers. The dial on the top right lets users navigate between applications.

Although the watch is still considered a research prototype, we have already run some personal information management applications such as calendars, address books, to-do lists, and e-mail summaries, and we have used it to interact wirelessly with other computers running specific applications. We have also used the watch to communicate with PCs, personal digital assistants (PDAs), and other

0018-9162/02/$17.00 © 2002 IEEE

Figure 1. IBM wristwatch computer with (a) OLED and (b) LCD display. The slightly larger LCD watch features auxiliary card housing under the wheel.

wireless-enabled devices, viewing condensed e-mail and receiving pager-like messages. Eventually, users will be able to access various Internet-based services, such as information about weather, traffic conditions, the stock market, and sports.

CONCEPT DEFINITION

The idea for a wristwatch computer evolved from a study of how wearable computers would affect people and from the general desire across IBM to create a cutting-edge technology. Everyone agreed that cell phones and PDAs were useful but too easy to lose or misplace. As groups exchanged ideas and information, the idea of a watch computer began to gel, and people were getting excited. Many agreed that this kind of device would be extremely wearable, likely to be with you, and easy to access.

The idea took off in June 1998, when a team member created an attention-grabbing graphical animation of the smart watch concept and presented it at an in-house conference on pervasive computing. People outside our group left thinking, "What if we could really build this?" Seiko's announcement in mid-1998 that it was building the Ruputer,[1] their version of a watch computer, became our line in the sand.

QUESTIONS AND MORE QUESTIONS

In December 1998, IBM Research experts in system design, operating systems, computer graphics, displays, batteries, user interfaces, data synchronization, and mechanical packaging assembled in Hawthorne, New York, for a day-long kickoff meeting. The exchange left several people eager to build the watch without delay.

Knowing that a "make it and they will come" attitude alone wasn't going to lead to success, we tried to first answer some basic questions: Who were the target users? What would they do with the watch? What input devices would we need? Who would write the software? What other devices would interact with the watch? These questions led us to examine relevant technologies—display, CPU, connection mode, memory, operating system, battery life, and data storage.

Personal information management, wireless information retrieval from the Web, and location-based services emerged as the most wanted applications. People called the watch everything from a generic pager to an information appliance. It became clear that different people wanted different applications, which meant that we had to make the watch programmable.

We examined several high-function watches, pagers, PDAs, and cell phones to see what was already available. Seiko's Ruputer, which ran DOS, was one of the most general-purpose watch computers. It did a reasonable job, but we felt we could improve on its user interface, which consisted of several buttons for navigating the watch's functions, a joystick for input, and a 102×64 pixel reflective LCD display. We came away with two important goals: emphasize aesthetics and improve on user interface intuitiveness.

By the end of January 1999, we had completed a preliminary specification describing high-level goals, key research areas, target applications, and user interface and input device guidelines. We had also completed our survey of currently available high-function watches and pervasive computing devices.

BASIC DESIGN DECISIONS

As Figure 2 shows, we ended up with three boards—main, communications, and display. The LCD version has the main board and the communication board; the OLED version has the main board and the display board.

On the main board is an ARM processor from Cirrus Logic (EP 7211), which because of its availability in bare die form, took much less space than a standard-package processor. The EP 7211 came with on-chip ROM for initial boot; an integrated LCD controller; an infrared controller; serial communication, sound, and other interfaces; and two serial ports for connecting to other devices such as PCs and PDAs.

The communications board evolved from our decision to include not only infrared but also radio frequency wireless access. Including infrared access was straightforward because the EP 7211 had a built-in infrared module. We also wanted radio frequency access because it is omnidirectional and

could be used for communication without actively involving the user. Because we did not know for certain if the size and power constraints made it feasible to add a Bluetooth radio, we opted to put the Bluetooth module on a separate card.

The display board contains the OLED, which we chose over the LCD with little hesitation, aesthetics being an important design goal. We had planned to accommodate the OLED display on the other side of the main board. Unfortunately, only the processor was available in a bare die, and many parts destined for only one side of the main board overflowed to the other side, which meant there was no room for the OLED chip.

No existing display met our need for low power consumption, high pixel density, and controls to trade off brightness for power, so we decided to build one jointly with eMagin Corp.[2] We wanted the OLED display to reside on a silicon chip that the processor would see as static RAM. Every high bit in the static RAM would then turn on one of the LEDs in the array. Thus, unlike LCDs, the display would not need refreshing. The OLED display would not be available until several months after the main board, however, so we decided to use a 96×120 pixel reflective LCD display in the interim so there would be two versions—an LCD and an OLED.

Our decision to use a touch screen and roller wheel as input devices took a little longer. Touch screen proponents argued that the absence of buttons made the watch elegant and simplified the user interface. Those who were opposed argued that the smudges on touch screens would look awful and decrease the display's readability.

We decided that the touch screen should have just four distinguishing quadrants, corresponding to the display's four corners. If this proved too limiting, we could add a fifth zone at the display's center; anything finer would be very hard to use. To navigate the applications and data, we added a roller wheel; to experiment with a speech interface, we included a speaker and a microphone.

Once we had the board designs, we began estimating power consumption and drain. It soon became clear that we needed a rechargeable battery to supply the peak current that the watch would draw.

We had yet to select two major items in the specification: the operating system and graphics library. The choice of operating system led to interesting discussions ranging from "Who needs an operating system for a watch?" to "We need to provide an industrial-strength operating system."

We looked at several embedded operating systems such as pSOS, vxWorks, EPOC 32, and QNX and spoke to others who had studied this question under different constraints. We soon learned that it is difficult to license a commercial operating system for a research project without well-defined product plans or volume projections. The development environments were also rather costly.

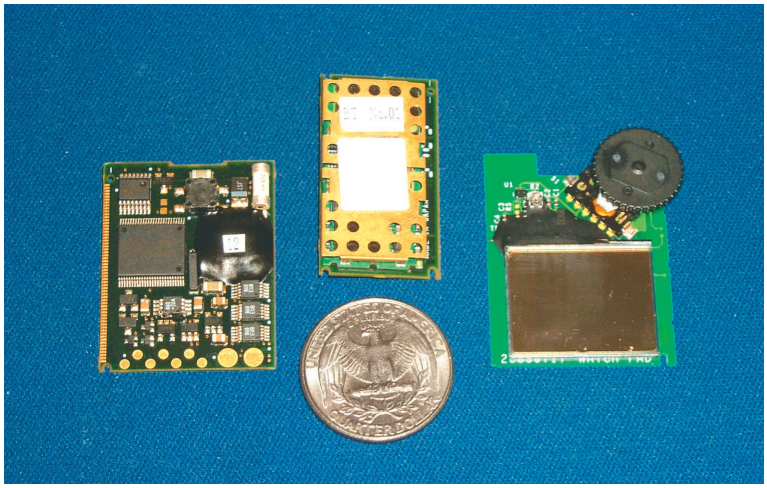


Figure 2. The three boards designed for the wristwatch computer. The main board (left) contains the processor, the communications board (middle) houses the Bluetooth module, and the display board (right) contains the organic LED display.

Inside IBM, effort was being directed at Linux, which was a good reason to choose it. Another compelling reason was cost: The source code and the tools were free. Moreover, many of us were familiar with running Unix on workstations. There was still a question of whether we could run Linux on a device this small, but since others had run Linux on devices with less memory and computational power, we were reasonably confident that it would run on the watch. People outside the project shook their heads and told us we had just sounded its death knell.

About the time we were debating the operating system question, we began to address the issue of graphics libraries. Given the small display and the limited amount of resources available on the watch, having a full-fledged graphics library initially seemed like overkill. Nevertheless, we believed that supporting a standard, well-understood graphics library would be beneficial because it would help simplify application development. Of the available choices, X11 was very stable. Though it was large, we knew that it would fit, and we knew that its developers were working on making it smaller.

Table 1 summarizes the evolution of the hardware specifications, which were completed in June 1999. The watch would have an ARM EP 7211 processor, 8 Mbytes of flash memory, 8 Mbytes of DRAM, a touch screen, a roller wheel, infrared and Bluetooth wireless interfaces, X11 graphics, and a speaker and microphone. We were ready to start designing and building the three boards.

COORDINATING DEVELOPMENT

In parallel with building the boards, we wanted to study user interface issues and work on the software libraries and drivers that would help us bring up the hardware. To do this, we needed some pro-

Table 1. How the IBM Linux watch specification evolved.

Hardware feature	Candidates	Final specification	Reasons
Processor	Motorola's DragonBall EZ and MPC 823, Strong ARM, ARM 7Series (Cirrus Logic PS 7111/EP 7211), NEC V4111, Hitachi SH 7709a, VLSI VWS 22100	ARM 7 (Cirrus Logic EP 7211)	Required little additional logic; available in bare die form; low power consumption; offered several power modes; had proven success in running popular operating systems
Memory	As large as possible	8 Mbytes of flash memory, 8 Mbytes of DRAM	Compatibility with operating system footprint, application size, and data storage on ARM architecture; memory configuration of existing devices
Wireless access mechanisms	Infrared, Bluetooth, cellular, pager systems	Infrared on main card, Bluetooth on separate card	ARM 7 chip has infrared module; too many unknowns to put Bluetooth on main card
Display	Backlit or reflective LCD, organic LED	640 × 480, 741-dpi OLED	Easier to read than reflective or backlit LCD; consumed less power than a backlit LCD
Input device	Buttons, touch screen, jog dial, rocker, joy stick	Touch screen, jog dial	Ease of use, more elegant appearance
Power medium	Lithium, Zi-Air, and alkaline batteries; solar, self-winding; thermocouple-based energy	Rechargeable, lithium polymer batteries	Ability to cope with the watch's peak current draw; solar and other self-winding methods did not generate sufficient energy
Operating system	None; embedded systems: Linux, EPOC32, QNX, vxWorks, DOS	Linux	Availability of source code and tools; worked on a system with limited memory and computing power
Graphics	X11, direct frame buffer access, Microsoft Windows	X11	Most complete, stable, and general library; available with source code

totyping platforms. The first was an x86-based PC running Linux that would let us look at the Linux source code and tools. To simulate applications that would run on the OLED watch, we used a backlit, color LCD display (27 × 20 mm), which we had used as part of an earlier project. The display had a 640 × 480 pixel resolution but was slightly larger. It was useful in prototyping different screens to determine how easy or difficult it was to view them and what was readable on the display. As Figure 3 shows, the display had an interface module that would send standard monitor output from a laptop to the LCD display.

To bring up Linux and build some of the device drivers, we procured development boards from Cirrus Logic that housed the EP 7211 and directly attached the reflective LCD displays. We planned to first mount the circuit boards for the watch onto 10 × 15 cm (4 × 6 inches) test benches, such as the one in Figure 4, to make hardware debugging easier. After some initial hardware testing, we planned to assemble the boards into watches. The first watches would then have the lower-resolution reflective LCD display.

When the OLED displays were finished, we planned to put them on the test benches first and then assemble watches with OLED displays. We also planned to build docking stations, or cradles, so that users could charge the watch batteries and easily transfer data from a PC to the watch. The platforms helped ensure that the developers stayed focused on a common vision, feeding what they learned into the hardware and software designs.

A team from Tokyo led the hardware design, selected the parts, and built the boards. The same team led the industrial design, contributed to the mechanical design, ported the Bluetooth protocol stack, and built Bluetooth-related applications. A team from Yorktown worked on the software architecture, user interfaces, prototype applications, Linux driver architecture, and overall project management. A second team from Yorktown did the mechanical packaging work. Yet another team from Yorktown collaborated with people from IBM Zurich and eMagin Corp. to build the OLED display. A team in Bangalore, India, worked with the team from Yorktown on the Linux drivers, ways to port the X11 library, and the hardware test code.

BOARD DESIGN

The board design was challenging, and a reduction of even 0.5 mm was considered well worth the work. We needed additional switching voltage regulators for the OLED display, and we had to make sure that the power supply noise would not affect the Bluetooth module. Because of these constraints, we tried to eliminate all the wires to the touch panel, roller wheel, accessory card, and battery, using elastomer connectors and gold-plated spring contacts instead.

We carefully selected all the board parts and designed them to meet these requirements.

Unfortunately, however, we could not get all the parts as specified, and we ended up with several manual wire connections in the LCD watch—a problem we rectified for the OLED watch by using an elastomer connector.

We were particularly concerned about the power consumed in standby mode. We had to consider future functional expansion using auxiliary boards and how those boards would interface to the main board for other types of devices. In light of this, we opted for fine-pitch expansion tabs at the card edge to support various external expansion interfaces. We used IBM's surface laminar circuit packaging technology to design a multilayer circuit board with extremely fine-pitch connections.

We bonded the bare-die processor chip directly onto the board and sealed it with epoxy. Using a chip-stacked memory module for the flash memory allowed us to accommodate variations in memory size and types in the same board design. To save real estate, we attached the chip-size package DRAM to the board in a flip-chip fashion. Many of the other components were extremely small, and the team put them on the main card at IBM's advanced packaging facility in Yasu, Japan. The smallest chip resistor was $0.3 \times 0.6 \times 0.3$ mm, requiring a microscope and an ultrafine solder iron for touch up, not to mention excellent fine-motor skills and steady hands.

Since the processor was a bare die, we could not test it exhaustively before assembly, so we added an external in-circuit emulator and a diagnostic LED for debugging. We also had to be extremely careful in verifying the design and assembly because errors could render the entire board unusable.

The expansion board concept was key even though the system could consume more power. We believed that the benefits of additional functions, radio frequency communications, and sensors were a good tradeoff.

PROTOTYPING EXPERIMENTS

Although we knew the LCD watch would appear first, we started with application and user interface design for the VGA resolution OLED display[3] because we felt that it afforded a richer set of applications. Initially, we used slide-presentation software to create screens to study different font types and font sizes to see what was readable. We also experimented with using slide-show animations to display clock faces and other graphical content, but we very quickly discovered the limits of what you can do with slide shows. We switched to writing actual code on a Linux system to model user interactions and how the screens would change in response. To model the interactions, we simulated the touch screen and the roller wheel.

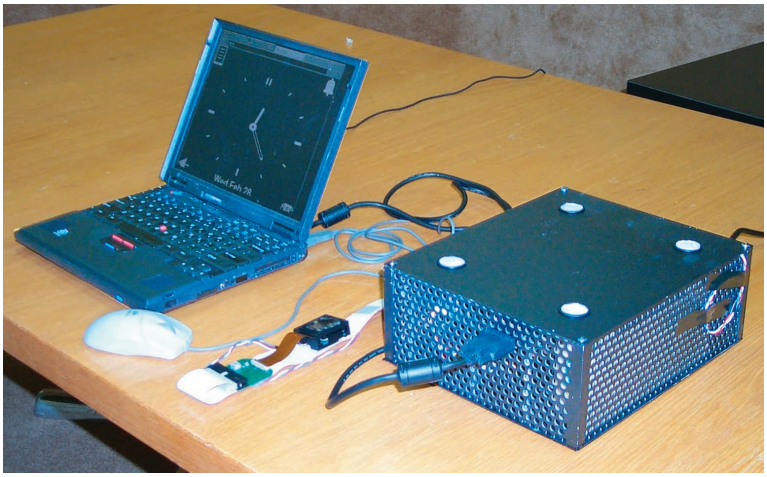


Figure 3. Backlit LCD display used in application and user interface prototyping for the watch with the OLED display.

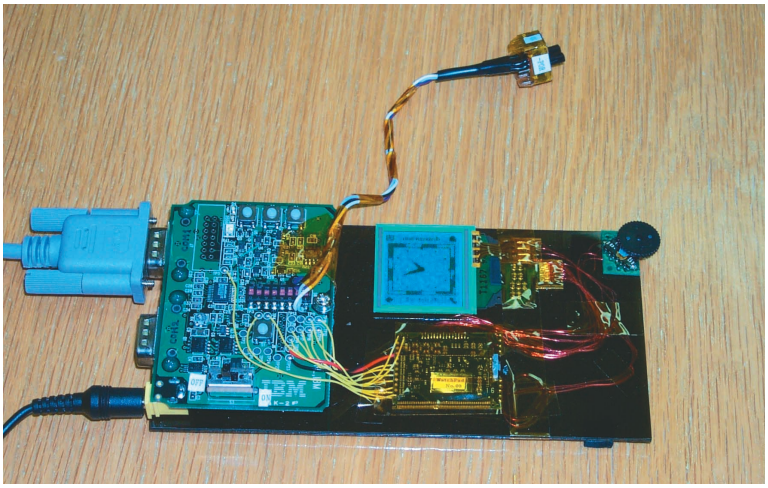


Figure 4. Test bench for hardware debugging. The bench is roughly 4 × 6 inches.

Because we had no touch screen for the display, we used four keys on the laptop's keyboard to emulate it. Our roller wheel was similar to that on a wheel mouse, so we connected a wheel mouse to the laptop to emulate the roller wheel. With these input mechanisms, we prototyped many potential screens and captured the navigation between them. To enable rapid prototyping and easy code changes, we used T/cl/Tk, a high-level scripting language and graphics toolkit, to write the code.

We began showing the prototype to people and asking how easy or difficult they found navigation between screens. When we showed our demos, the large, heavy interface module that connected the backlit LCD display to the laptop became a source of amusement. People often asked if we had to carry this large box to use the watch, to which we quipped, "No, it contains the watch's batteries."

Figure 5. Cirrus Logic development board with liquid crystal display. The LCD was used first on the development board and then on the test bench before finally moving the code to the actual watch.

During this time, we had to decide whether to orient the OLED display in landscape or portrait mode. Wristwatches are generally in portrait mode, but we were not sure if we wanted to use this option for our watch. To see which was better, we cut a rectangle the size of the OLED display from a piece of paper and placed the cutout on various text and graphic images in both orientations. When we placed the rectangle on a phone book to emulate text in a small font, we preferred the landscape orientation because fewer lines with more characters per line were easier to read than fewer characters and more lines. For graphic images, the orientation did not seem to make much difference.

To select the color of the OLED display, we used a 600-dpi printer and printed transparencies using text in various colors on a black background. To get an approximation of what the actual OLEDs would look like, we placed the transparencies on a fluorescent screen and adjusted the color saturation to achieve equal brightness. Yellow text was more readable than blue or green because of the higher contrast. This was good news because yellow OLEDs used a reasonable number of lumens per watt. Green used the fewest lumens, but yellow was not far behind and was far more readable.

After we built the application prototype for the VGA resolution OLED display, we converted everything to fit the 96 × 120 pixel resolution LCD display. Because we had written the code in a high-level scripting language, conversion was easy, but we found that we could not perform some functions on the LCD display. Luckily, the dot pitch on our laptop LCD displays was comparable with that of our LCD. In the initial stages, we simply developed the prototype using a 96 × 120 pixel window on the laptop screen. We then used the actual LCD, first on a Cirrus Logic development board and then on the test bench, before finally moving the code to the actual watch.

We ran the application code on a laptop while another program on the laptop periodically took snapshots of the window and sent it on a serial line to the Cirrus Logic development board, as shown in Figure 5. We used the laptop keyboard and the wheel mouse for the interaction. Later, once Linux and X11 were running on the development board and the test bench, we ran the code on the watch.

We followed the identical steps for the OLED display. Having some demonstrable functionality early on proved valuable. The demos were exciting, and we received valuable input that improved the design. The feedback also helped us measure our progress as we moved the code from one prototyping platform to the next, getting closer to the final hardware.

PACKAGING ISSUES

Until we had put the board, touch screen, and all the hardware into an actual watch case and attached wrist straps, we did not really have a *watch*. We explored several packaging options, including simply buying a watch and replacing its guts with our hardware. We worked on the case for the LCD watch while the boards were being built. Regrettably, we could not find a watch that was the right size and shape, so we had to customize the case and build it.

Our industrial designers and mechanical engineers used 3D tools to model the placement of components on the boards and the positioning of the boards inside the case. Because this was a prototype, we did not devote much attention to many issues that watch designers worry about, such as waterproofing and shock resistance.

For connections between the watch and the spring pins on the watch cradle, we used a flexible circuit wrapped around the battery, which would easily accommodate a thicker battery if we needed it. After a few iterations, we developed a compact cradle design with a docking mechanism that accommodates watches with both the LCD and OLED displays. We used optic fibers to route the status lights on the circuit board to the top of the cradle and designed a special mechanical assembly to bring the push buttons on the cradle board to the top of the cradle.

The watch strap is a custom design so that a user could more easily manipulate the wheel, but it uses standard buckles and band holder pins, which we purchased at a neighborhood watch store. Our repeated trips to get buckles and pins (one trip for each prototype) roused the store owner's curiosity, and she asked if we were building watches. When

we showed her the watch and what it could do, she was eager to sell it.

GETTING LINUX TO RUN

In preparation for our most significant step, we needed to build a cross-development environment, including cross compilers, assemblers, and binary utilities that would let us generate code for the ARM architecture on our x86 desktops. Although there were many resources on the Internet to help in this process, it took some time to get everything set up right. We also had a fairly steep learning curve.

When we began the project, nobody was running Linux on an EP 7211, but patches were available to run Linux on an earlier generation, the CLPS 711x. After finding relevant pieces of source code and patches on the Internet, we wrote some of the basic device drivers and modified the memory maps for our hardware.

The EP 7211 had a good bootstrap mechanism in which the chip boots from an on-chip boot ROM and subsequently loads up the operating system code over a serial line into DRAM or flash memory. This bootstrap mechanism made it possible to bring up Linux fairly rapidly. After a few iterations of debugging both the development environment and the device drivers, Linux would go through the boot sequence and start the kernel, but it would crash the first time a process was forked. By April 2000, we had solved these issues and a "#" prompt from the shell demonstrated that Linux was running on both the development boards and our test bench. To get to this point, we had to get the kernel running on the watch, as well as the basic libraries (libc, libm, and libdl), a shell, and some basic commands.

We then began preparing a demo program to show that we had Linux running on our watch. We first redirected the bootup messages to the LCD. The messages scrolled past quickly because it took less than 10 seconds for the watch to boot up. We had to slow down the boot process to actually see the messages scrolling up on the LCD. After bootup, we started an endless shell loop that executed basic Linux commands such as echo, ls, and mv. The loop demo, shown in Figure 6, demonstrated that we could run Linux and a shell on the watch, but we needed to do more. We endured many jokes about having to type "date" to get the time.

BRINGING UP X11 GRAPHICS

Although we had won a major battle in getting Linux to run, we still needed to get X11 running as well. Most people thought we were crazy, but our preliminary calculations showed that early workstations running X11 had less memory and horsepower than our watch. If X11 could run on those machines, it would certainly run on our device. And we had good reasons for our choice. X11 works on top of Linux. Other graphics libraries such as NanoX, MicroWindows, and FLTK were still in their infancy. We would be leveraging a familiar open source graphics API on top of an open source operating system. Finally, we could run applications written in scripting languages like T/cl/Tk.

Figure 6. Linux shell loop demo. The demo program executed basic commands to show that Linux was running on the watch.

The road to implementation had more than a few potholes, however. When the server came up, a simple program would crash the system, which we discovered was caused by errors in the display driver and pixel-writing routines. When we had progressed to the stage where a drawn line appeared as several disjointed lines, we knew we were close. After ironing out a few more bugs, we were able to run the xclock program. In early July 2000, the prototype watch could display the hour and minute hands and keep time, and our colleagues finally stopped asking, "Yes, but does it show the *time*?"

DEBUT AT LINUXWORLD

We were ready to show our watch at a public forum, and we knew it was newsworthy. After some discussion, we decided that the LinuxWorld Expo in mid-August would be the ideal place.

We already had the endless loop of shell commands and the clock program, but to prove to skeptics that Linux was running on the watch, we had to provide a method for typing in Linux commands that the watch could execute and for displaying results. The cradles arrived just in time for us to set up a demo in which an interactive shell program accepted commands from the serial interface on the watch and executed them in response to the user. A serial cable connecting the cradle to a laptop running a terminal emulator completed the picture.

On 7 August 2000, IBM issued a press release[4] stating that we were successfully running Linux on a wristwatch prototype. The *Wall Street Journal* and *IEEE Spectrum,* among others, covered the

Adding Bluetooth connectivity to the LCD watch was a major development step.

announcement, and the response was overwhelming. The exposure set off discussions in various online forums, such as Slashdot's "News for Nerds, Stuff That Matters."

When we arrived to set up the watches at the LinuxWorld Expo, our booth was already crowded, and the universal reaction was "way cool." It was a heady experience. Linus Torvalds showed up and said he would like one. Jim Gettys, one of X11's principal authors, and Eric Raymond, an advocate of the open source movement, also stopped by. Parents who saw the watch brought their children in to see it the following day. The team from Compaq that had built Itsy[5] seemed interested in how we had managed to pack so much into something as small as a watch.

FROM LCD TO OLED

Shortly after our LinuxWorld debut, the first OLED displays became available, and they were more impressive than we had imagined. Relative to the backlit LCDs we had used for software prototyping, the OLED display seemed positively dazzling.

Building the OLED displays had been complex. Our fabricators had first built silicon circuits on the wafers and then deposited multiple layers of organic materials on top. In the first set of displays, the fabricators had discovered the need to seal the chip to keep the organic materials from deteriorating. They had also encountered various deposition and packaging issues.

The OLED and LCD designs were different,[6] but we still got the OLED displays to work on the first pass. To verify that it was up and running, we had the OLED display show different patterns. Instead of building custom test circuitry to show the patterns, we used a video card with digital output. The video card scanned the contents of its video RAM and output one word at a time. By creating different sequences of 1s and 0s in the video RAM, we generated digital waveforms of different shapes and timing patterns. We used this technique to generate the appropriate addresses, read and write cycle timings for the OLED chip, and image data. In addition to encoding many images to determine display quality, we used images from the application prototyping to see how they would look on the watch.

Connecting the OLED display to the main card was not easy because it involved making about 50 connections at a fine pitch. We used an elastomer connector, but it was difficult to make this connection reliable. Finally, we changed the elastomer and increased the compression to ensure proper contact.

Our last challenge was to bring up Linux and X11 on the OLED watch, and we ran into another interesting problem. The OLED had a 16-bit-wide data bus and expected short word stores to write data into it. When we built the X11 server code, we had not specified the options that told the compiler it was okay to generate short word stores. As a result, the compiler converted each short word store into two-byte stores, resulting in a garbled display. We kept staring at the source code and found nothing wrong. When we finally decided to look at the object code, it became obvious that we needed to specify the compiler options.

ADDING BLUETOOTH CONNECTIVITY

About the same time we were working on interfacing the OLED display to the main board, we were tackling another major development step: adding Bluetooth connectivity to the LCD watch. Again, we benefited from our decision to use Linux. A sister group at IBM Watson Research had already written the Bluetooth protocol stack and drivers for Linux for an Intel processor. We partially modified their code, which they called BlueDrekar, and fought some gcc compiler idiosyncrasies to make it work on the ARM processor. Overall, however, the task was simpler than writing the code from scratch.

In December 2000, we demonstrated the Bluetooth-enabled watch with the LCD display at the Bluetooth Developers Conference in San Jose, using it to advance the foils for a keynote presentation to about 3,000 people.

Because the human body easily absorbs and affects the 2.4-GHz wave, we were careful about locating the antenna. We succeeded in impressing the keynote attendees, who were surprised by the watch's communication range.

We are confident that it is possible to build a highly functional and elegant wristwatch computer. Although we are not there yet, we are several steps closer than when our vision began. Our current focus is on software components, power management issues,[7] and other ways people can use the watch.[8] For example, they may prefer using it to display data from their other devices, even though they are carrying them. It is easier to see an agenda or a map on your wrist than it is to fish around in a briefcase or purse for your PDA. We are also prototyping services tailored to the user's current location and context and looking into services that leverage connectivity to the Internet.

Another important step is the research collaboration of IBM Research and Citizen Watch, announced in October 2001,[9] to further explore the possibility of bringing the watch to a wider audience, conducting pilot tests, and pursuing collaborative research with universities. Updates on our progress are available at http://www.research.ibm.com/WearableComputing.

Our work has convinced us that a lot of functionality can be squeezed into small devices. The availability of the source code, the APIs, the reliability of the software, and the enthusiasm of the Linux community are paving the way for Linux to become an attractive operating system for such devices. As researchers combine high-function miniature hardware with a robust operating system and well-designed applications with intuitive user interfaces, we will see more science fiction gadgetry become everyday tools that people take for granted. ■

Acknowledgments

Numerous people have contributed to the success of this project. We have received much encouragement and enthusiastic support from our management. Several others have given us useful suggestions and provided valuable technical input. We thank the following individuals and apologize to anyone we might have inadvertently omitted: Rachel Bellamy, Tilman Beierlein, Tom Corbi, Steve Depp, Murthy Devarakonda, Derek Dove, Ambuj Goyal, Paul Greier, Erik Harris, Randall Isaac, Michael Karasick, John Karidis, Kohichiroh Kishimoto, Jayadevan Kumanan, Kartik Kumar, Akihiko Mizutani, Robert Morris, Alex Morrow, Michael Olsen, Bob Olyha, Carl Powell, Walter Reiss, Heike Riel, Tiaa Sahni, Omesh Sahni, Alfred Spector, Sunil Sreenivasan, Vijay Sukthankar, and Ken Tamagawa.

References

1. "Ruputer the Wrist Computer," http://janus.free.fr/ruputee.html (current Dec. 2001).
2. eMagin, "eMagin and IBM to Demonstrate New OLED Display at the Consumer Electronics Show," 5 Jan. 2001, http://www.emagin.com/pressreleases/pribmwatch.htm (current Dec. 2001).
3. C. Narayanaswami and M.T. Raghunath, "Application Design for a Smart Watch with a High Resolution Display," *Proc. 4th IEEE Int'l Symp. Wearable Computers*, IEEE CS Press, Los Alamitos, Calif., 2000, pp. 7-14.
4. IBM Research, "Linux on a Wristwatch," Aug. 2000, http://www.research.ibm.com/WearableComputing/factsheet.html (current Dec. 2001).
5. W.R. Hamburgen et al., "Itsy: Stretching the Bounds of Mobile Computing," *Computer*, Apr. 2001, pp. 28-36.
6. J. Sanford and E. Schlig, "Direct View Active Matrix VGA OLED-on-Silicon Display," *Int'l Symp. Soc. for Information Display, Digest of Technical Papers,* vol. 32, 2001, pp. 376-379.
7. N. Kamijoh et al., "Energy Trade-offs in the IBM Wristwatch Computer," *Proc. 5th Int'l Symp. Wearable Computers,* IEEE CS Press, Los Alamitos, Calif., 2001, pp. 133-140.
8. C. Narayanaswami et al., "What Would You Do with a Hundred MIPS on Your Wrist?" tech. report RC 22057, IBM Research, Yorktown Heights, N.Y., May 2001.
9. IBM Research News, "Citizen Watch and IBM Research Announce Research Collaboration on Linux Watch Technology," http://www.research.ibm.com/resources/news/20011011_watchpad.shtml (current Dec. 2001).

Chandra Narayanaswami *is a manager of the wearable computing department at IBM T.J. Watson Research Center, where he is the overall project leader and an architect for the Linux watch.* ***Noboru Kamijoh,*** *a senior researcher at the IBM Tokyo Research Lab, is the watch's lead hardware architect and hardware designer.* ***Mandayam Raghunath*** *is a researcher at Watson and was in charge of the system software and application design.* ***Tadanobu Inoue*** *is a researcher in Tokyo and worked on the design and verification of the hardware.* ***Thomas Cipolla*** *is a senior engineer at Watson and was in charge of the watch's mechanical packaging.* ***Jim Sanford*** *is a senior engineer at Watson and one of the OLED display designers.* ***Eugene Schlig*** *(deceased) was a research staff member at Watson and another OLED display designer.*

At IBM Software Laboratory, India, ***Sreekrishnan Venkiteswaran,*** *a software engineer, helped bring up Linux.* ***Dinakar Guniguntala,*** *a software engineer, helped bring up Linux and X11 for the OLED watch.* ***Vishal Kulkarni,*** *a software engineer, helped bring up X11 for the LCD watch. All three built device drivers.*

At IBM Japan, ***Kazuhiko Yamazaki,*** *a senior technical staff member, was in charge of the watch's industrial design.*

Contact the authors at chandras@us.ibm.com.