App Note 2036: Designing a Networked On/Off Switch Using the TINI Platform

Tiny InterNet Interfaces (TINI) can be used to control an on/off switch remotely from a web page applet. A small amount of Java code running on a remote TINI provides immediate switch state feedback and control across a network.

The TCP/IP network stack and local control capabilities needed to design an IP-networked relay are provided by the TINI platform. Adding a Java runtime environment greatly reduces the complexity with which small sensors and actuators can be accessed and controlled remotely over a network. The following discussion, which presents an IP On/Off switch constructed with a simple relay circuit and a TINIm390/400 verification module, can be extended to other remote monitoring and control applications as well. Familiarity with object-oriented programming such as the Java language is assumed.[1]

The circuit is controlled with an application called TINIWebServer (slightly modified), which is executed directly by the TINI runtime environment. An applet, served to the host workstation, opens two-way communications back to the TINI runtime environment for commands and status, and displays a graphical user interface for simple remote control of the relay.

System software overview

The class com.dalsemi.tininet.http.HTTPServer enables the switch-control application to implement a simple web server whose only purpose is to transfer an applet to the remote host. Executed within the host's browser, the applet establishes a two-way TCP connection for exchanging commands and status with the TINI application. The applet also provides a graphical user interface for displaying controls and status. Figure 1 represents the overall software system.

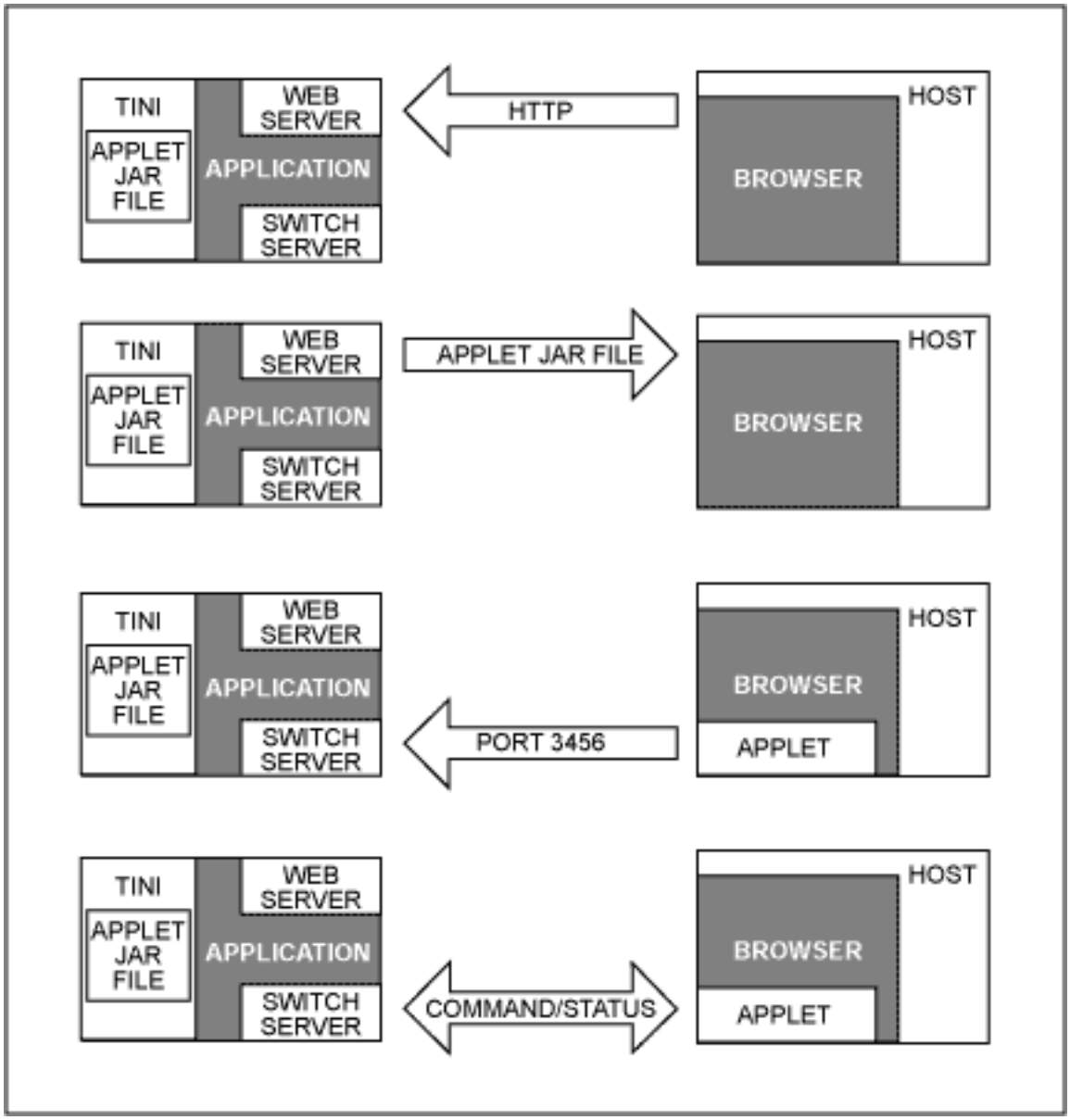

Figure 1. The web server application software executing on the TINI runtime environment uses an HTTP connection to transfer an applet to the host and sets up a two-way connection for transferring command and status data.

System hardware overview

In Figure 2, an On/Off control circuit forms the interface for a TINIm390/400[2] verification module. The TINIm390/400 provides Ethernet network connectivity and controls the switch through port pin P5.0 (other port pins work equally well). An n-channel power MOSFET controls the relay by switching current from the relay to ground. You can accommodate various current and voltage requirements by resizing the relay and FET, and the relay can be omitted altogether if you don't need to isolate the external circuit from the TINI verification module's power supply. The diodes protect against voltage spikes from the relay coil while the switch is changing state. To make possible new services such as networked switch control, the hardware and software components have been integrated in the TINI chipset reference design with standards-based

Internet technologies.

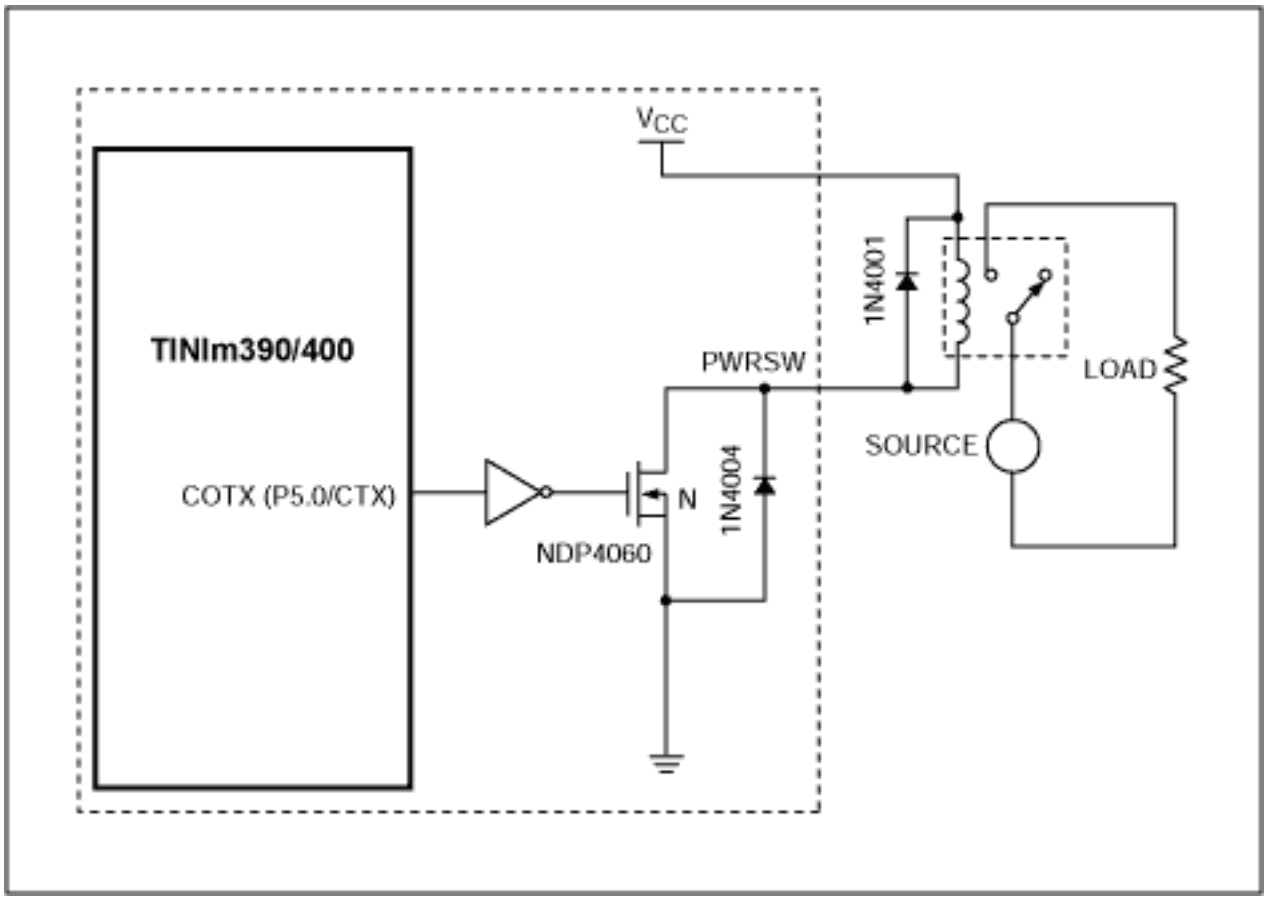

Figure 2. The TINI chipset reference design controls the switch, which is a simple transistor and relay circuit with protection diodes, using any available port pin.

The TINI switch-control application

Four classes make up the switch-control and web-server-interface portions of this application. The PowerSwitch class interfaces directly to the hardware using the TINI class com.dalsemi.system.BitPort API class. The WebWorker class comes directly from the TINIWebServer example in our Software Developer's Kit (TINI SDK), and is responsible for servicing the arriving HTTP connections. The SwitchWorker class manages all command and status communications between the applet and the TINI application. The TINIWebServer class drives the application by binding together operation of the individual classes.

The applet establishes a two-way TCP connection for exchanging commands and status with the TINI application.

The PowerSwitch class is an interface to the hardware, creating a BitPort object for pin P5.0 in its constructor. Two methods are implemented in this class. The On method sets the state of the port pin used to apply voltage to the relay's coil; the Off method removes voltage from the coil by clearing the state of the port pin. The single pole, double throw (SPDT) relay in Figure 2 can

operate in the normally closed or normally open position, or it can switch the external voltage source between those two positions. The On and Off methods assume the circuit is normally open, and must be modified to accommodate the normally closed circuit. To indicate whether on/off corresponds to the BitPort set or clear method, an extra boolean variable (invert) can be added to this class. Another method (setInvert) would be needed to initialize the invert variable. The hardware diagram illustrates a normally open circuit.

The WebWorker class provides an interface between the network and the application. It simply sets up and drives an object (com.dalsemi.tininet. http.HTTPServer), which in turn spawns a thread that services each incoming HTTP connection. This class is essentially unchanged from the TINIWebServer example mentioned above. Accessible from any location on the network, the HTTPServer can prompt for a password, or accept other forms of control that limit access to approved users only.

The TINIWebServer allows remote control of the switch by tying together the network and hardware interfaces (Figure 3). The drive() method, for instance, sets up the web server by creating a WebWorker thread and creating the web page "index.html." The primary purpose of the web page is to download and run the applet on the host workstation. The application would not have to create the web page if the index page contained only static information. Rather, the index page could simply be copied to the web server's root along with the jar file containing the applet.

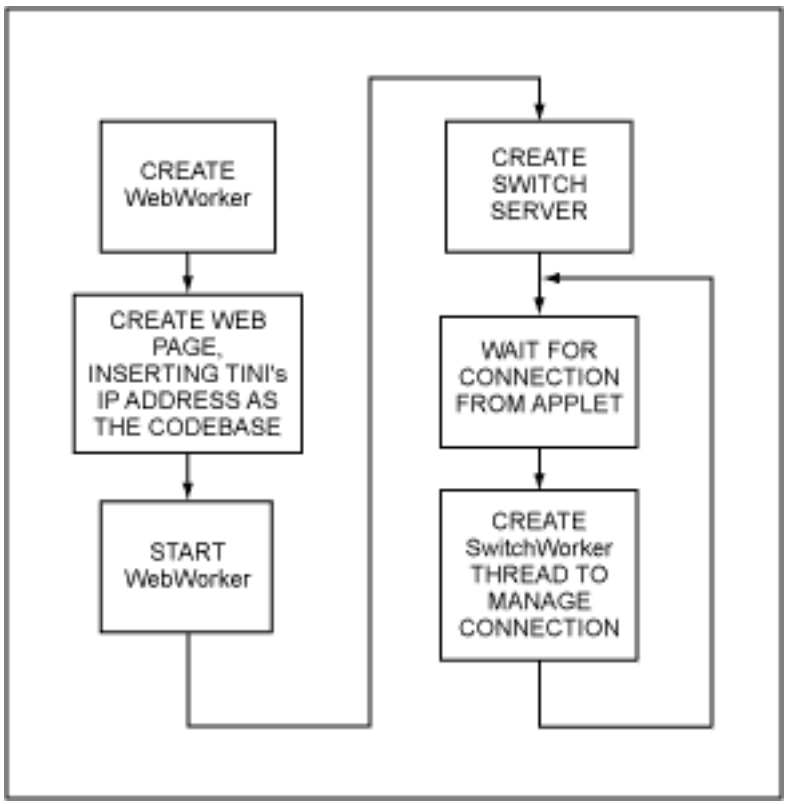

Figure 3. The TINIWebServer class creates a webpage and starts a webserver before creating the switch server to service incoming command and status requests.

The one parameter in the web page that changes on every TINI chipset design is CODEBASE. The applet uses that information to connect back to the TINI application on a separate server socket. A custom web page could be created and uploaded to each TINI chipset reference design that is installed in the field. An easier approach has the page created by the application each time it is run. The createIndexPage method creates the file index.html and inserts the IP address into the CODEBASE section using three writes:

1) index.write(indexTop.getBytes(), 0, indexTop.length());
2) index.write(InetAddress.getLocalHost().getHostAddress().getBytes());
3) index.write(indexBottom.getBytes(), 0, indexBottom.length());

The first and third writes copy the static portions of the web page into the file, and the second write copies the IP address to the CODEBASE section of the file. After the application sets up the web server and creates the page, it starts the web server. It creates a server socket to handle incoming connections from the downloaded applets, and calls the serviceConnection method each time an applet connects with the TINI application. The serviceConnection method creates a new instance of SwitchWorker and passes the socket to this class. The SwitchWorker constructor creates a new thread to manage the connection between the host applet and the TINI application. The serviceConnection method also handles the next incoming connection and then transfers control to the drive method.

The one parameter in the web page that changes with every TINI is CODEBASE.

The SwitchWorker class manages all communications between the applet and the webserver (Figure 4). Until the connection is dropped, it loops continuously, performing the following steps:

- Block on read(), waiting for a command byte from the applet.
- If the command byte is 0, turn the switch off. If the byte is 1, turn the switch on.
- Read the current switch state and send it back to the applet.

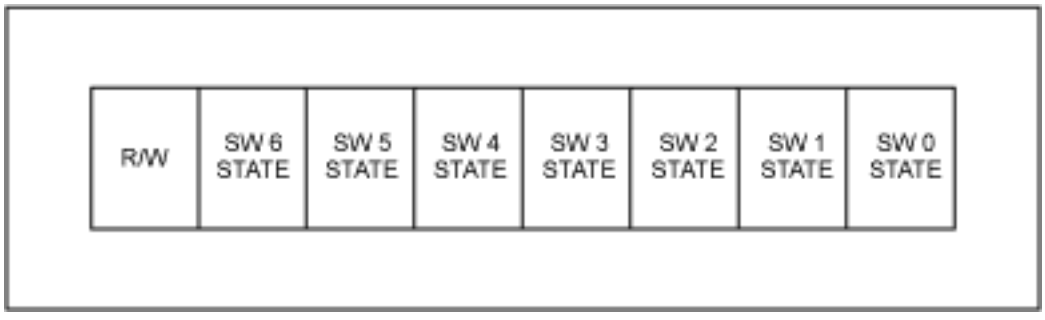

Figure 4. The SwitchServer executing on the TINI runtime environment sets up the listener and loops indefinitely, waiting for commands, setting the switch state based on those commands, and sending the updated state back to all listeners.

The algorithm can be extended to multiple switches by allocating each of the seven lower bits of

the command byte to represent the state of a separate switch (Figure 5). The most significant bit is reserved to indicate a read-only operation. The algorithm can also be extended to allow multiple applets to connect to a single TINI webserver application at the same time. The SwitchWorker simply keeps a "vector of listeners." Each time an applet sends a command to change a switch's state, the webserver sends the status back to all of the applets currently connected.

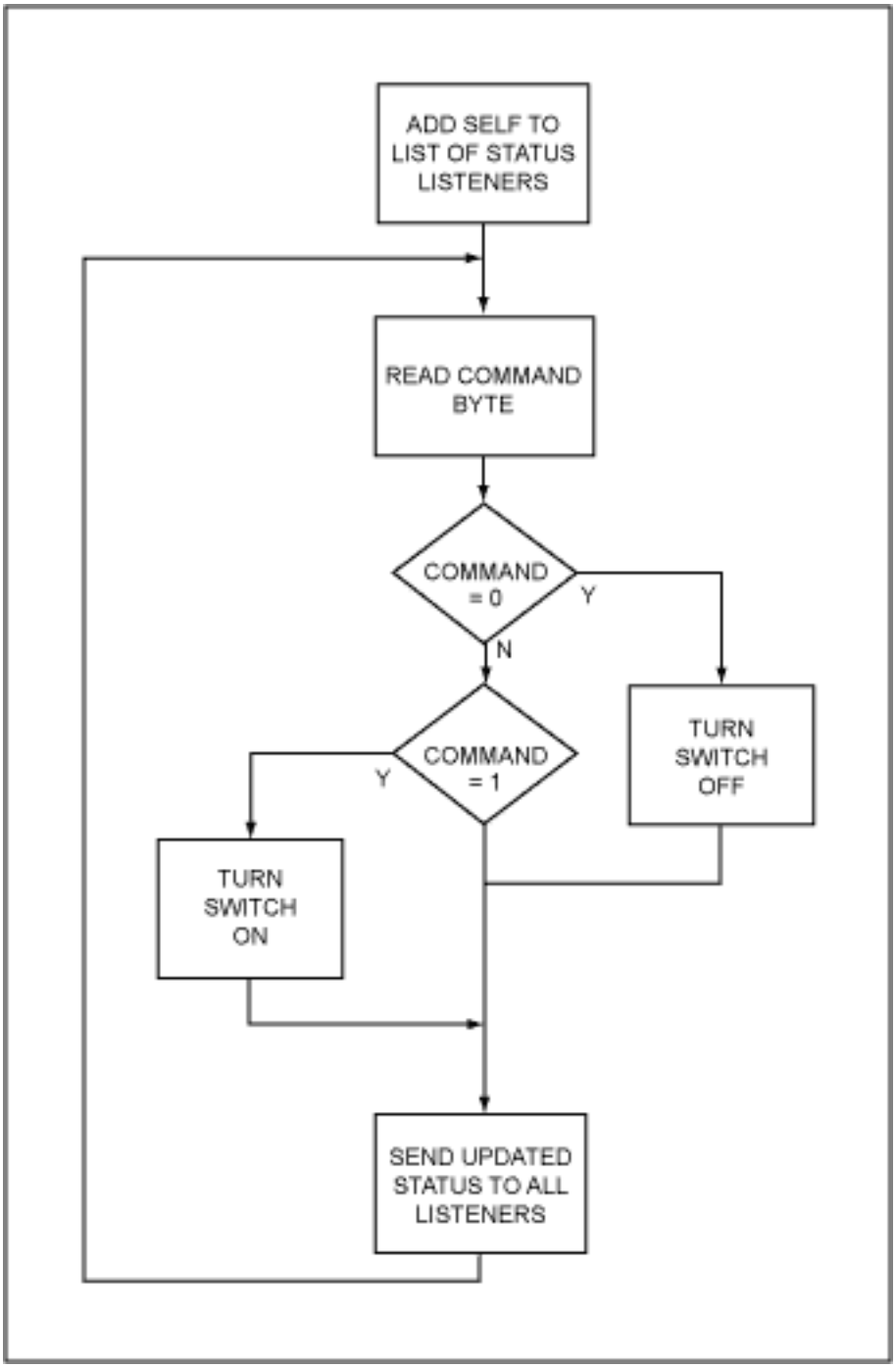

Figure 5. The command byte simply uses one bit per switch for control of up to seven switches and reserves the last bit to distinguish a status request from a command.

The host applet

An applet is used on the host because it provides a rich set of graphical objects for the display of controls and status, and enables (among other things) two-way communications. The TINI class com.dalsemi. tininet. http. HTTPServer class is fast and small, but it supports only the HTTP GET operation. The resulting data can flow only in one direction from the TINI application to the host. This application, however, requires data flow in both directions; commands are sent from the host to the TINI webserver and status is sent from the webserver to all connected hosts. Communication between the host and the TINI application incurs no protocol overhead. Its 1-byte commands and 1-byte status allow very quick responses for control and status.

The host applet includes two classes: the primary class (SwitchControl) handles host-side network communications and creates all graphical elements displayed on the web page (Figure 6); the other class (ImageButton) creates a graphical toggle button that displays one of two bitmaps according to the button's status (Figure 7). The toggle button should be sufficient for control and status of the switch, but the behavior of applets varies from one browser to another.

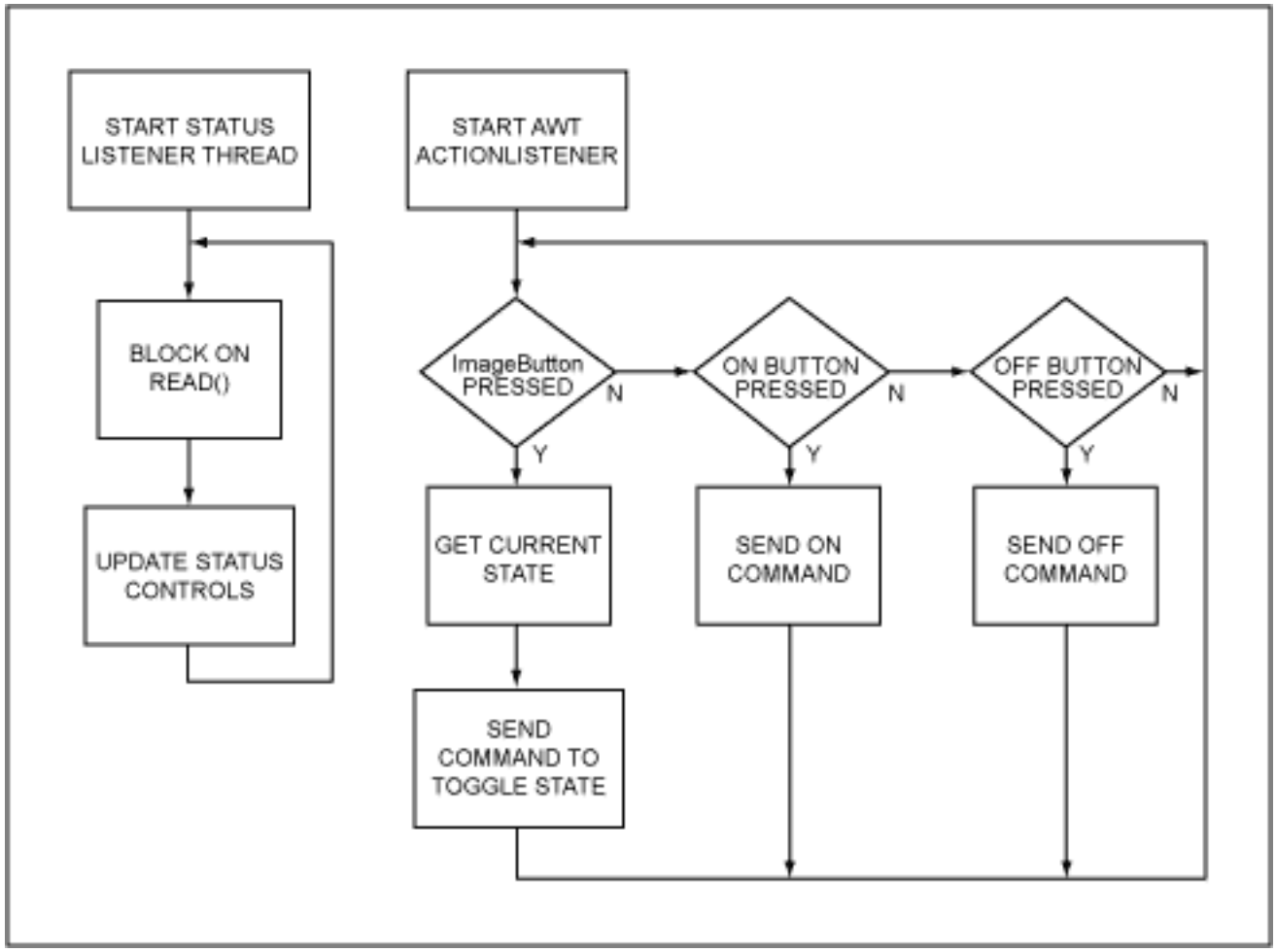

Figure 6. The applet's SwitchControl class uses one thread to read incoming status from the webserver and update the controls. It uses a second thread to detect user input and send the commands back to the TINI webserver.

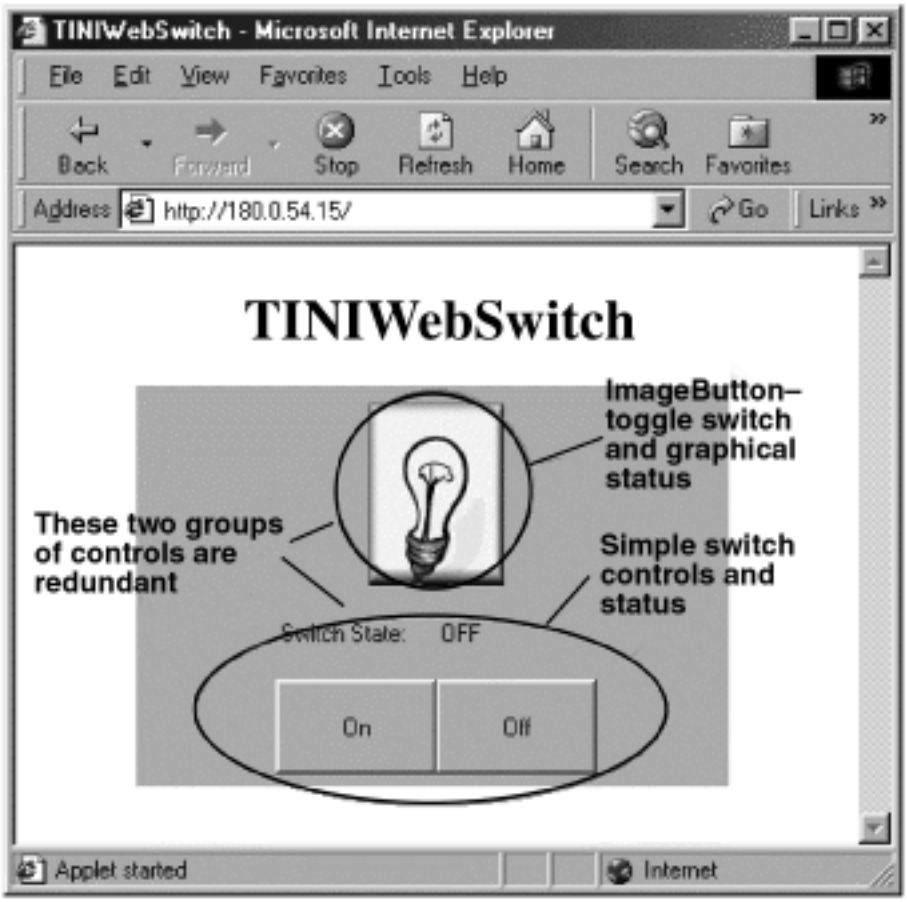

Figure 7. The switch-control applet supplies Off/On buttons and status fields to give remote control and feedback on the physical switch state.

Accordingly, simple On/Off buttons and a text window for status were added to accommodate a wider range of browsers. ImageButton and On/Off buttons perform the same control function, just as bitmaps and the status window perform the same status function. After creating the graphical elements, the SwitchControl class creates a status listener thread. The thread then sleeps, blocked on read, waiting for status from the TINI application. When the thread unblocks, the ImageButton bitmap and the status window are updated and the method loops back to the top of the method to await the next status byte. The applet-event thread drives the actionPerformed method, which is called each time a graphical button is pressed. If a call is triggered by the ImageButton or the On/Off button, it toggles the current state and sends an On or Off command to TINI. If triggered by the On/Off button, it sends an On/Off command. The ImageButton class is simply an exercise in AWT (Abstract Window Toolkit) component programming.

Conclusion

It is easy to implement an On/Off switch controlled remotely over a network with the TINI runtime environment, a Java application, and a simple relay circuit. The large selection of available circuit components makes possible a variety of applications, with control (from any location with

network access) of anything from light bulbs to industrial equipment.

Associated files

Files associated with this article are located at ftp://ftp.dalsemi.com/pub/tini/appnotes/NetSwitch/NetSwitch.tgz.

[1]Readers should particularly understand the terms class, method, object, and constructor in this context.
[2]The TINIm390 data sheet can be viewed at App Note 195: TINIm390 Verification Module Chipset Reference Design.

More Information

DSTINIm400: QuickView -- Full (PDF) Data Sheet