

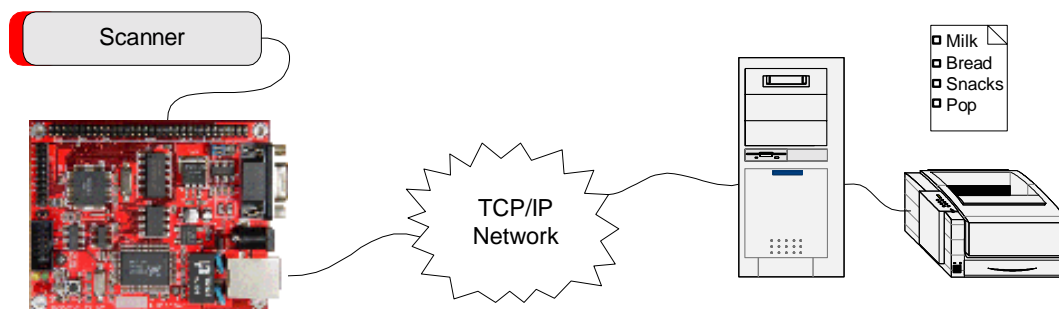
PantryPod, the recycle bins' network appliance

Everything you consume has a barcode. Producers use them to track product in the manufacturing phase. Shipping carriers use them to track product in transit. Retail stores use them to keep track of inventory and to tally your bill. Why can't you use this barcode to make your next shopping list? If you think for a moment about what you throw away and recycle, most of this represents what you'll buy more of. If you could easily scan these items, then you can build your shopping list.

This article describes the creation of a small network appliance which helps do just that. I call it PantryPod. I'll show you how to build one by presenting some of the concepts I explored, a few of the design tradeoffs in my selection process, and the details needed to build one. Along the way, you'll see how many of my requirements were derived. I'll start right here – my first and most fundamental premise was that I wanted to create a network appliance – so it needs to have an Ethernet connection, and be “network friendly” in my home network, meaning it should support standard protocols.

This device I built uses the AtMega 128 microcontroller, and an external Ethernet interface. By leveraging an open source design, and corresponding development tools, I got a quick jumpstart and avoided having to create a TCP/IP stack and services to run on top of it. The AtMega CPU provides more than enough power and I/O for this project. And, it does this with a low power consumption.

The initial implementation is quite simple, as you see from this system architecture.



To create PantryPod platform neutral, standard TCP/IP network protocols were used. To communicate to the home PC, I elected to use standard web server protocols. This made it easy to test each end of the network independently. The home PC is the web server, and PantryPod is the web client. My specific server implementation runs on a Windows XP machine, and uses Perl scripts running as a common gateway interface (CGI) under Internet Information Server (IIS). Since I chose to use flat-file architecture for storage, it should be very little work to port that software to other environments. As a result of these choices, the client software running on PantryPod does not care what server you use so long as the server's network interface is the same.

The hardware is based on a reference design that is popular among many embedded developers, and is a great fit for this particular application, providing hardware support

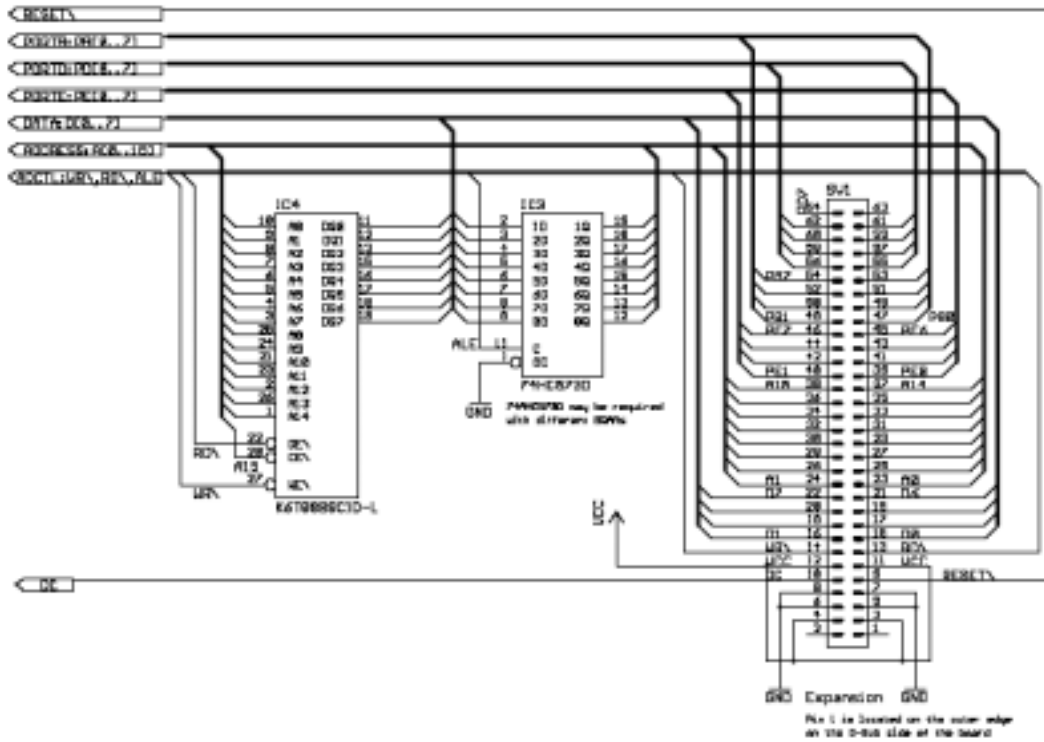


Figure 2 SRAM and Expansion

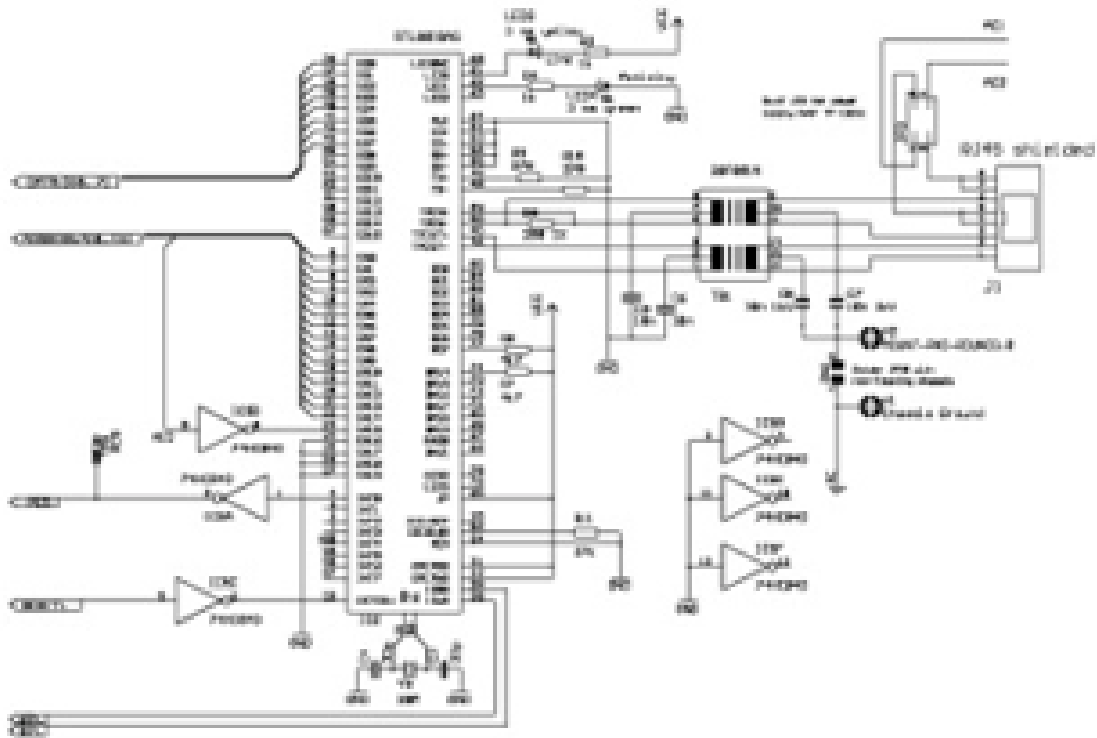


Figure 3 Ethernet Interface

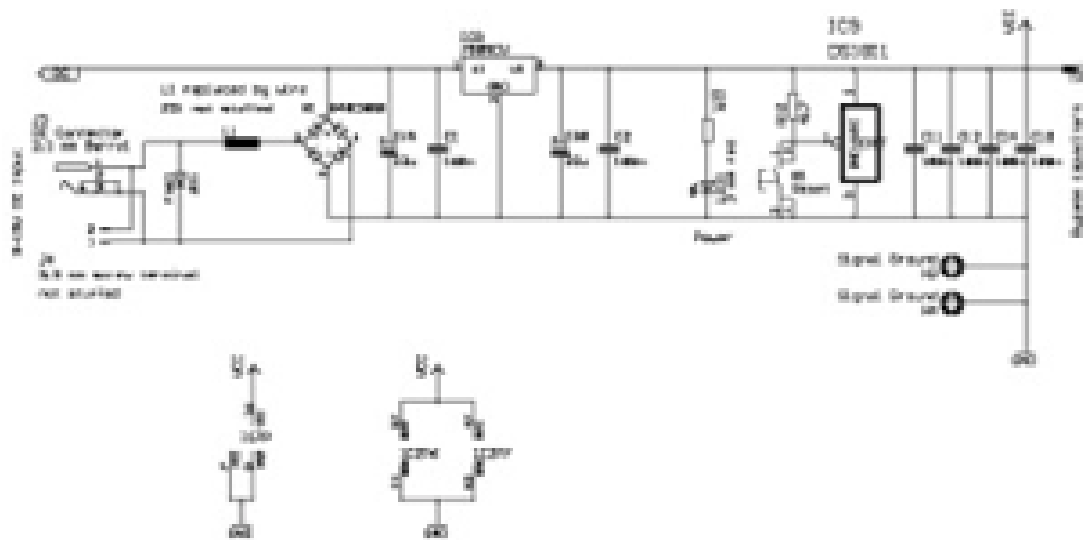


Figure 4 Power Supply

PantryPod hosts a small real-time operating system. So the implementation of threads for various services is easily done. There are 5 basic processing threads as you see in Figure 5.

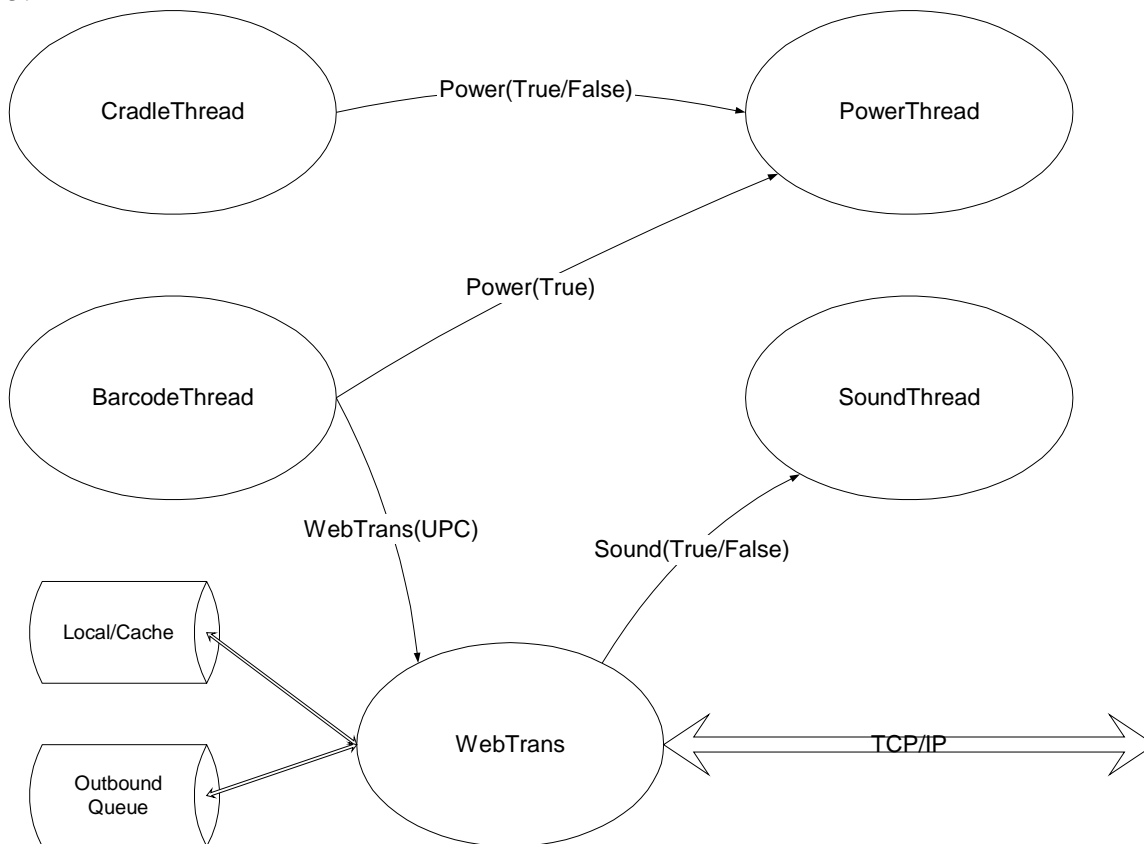


Figure 5 Thread Interactions

There are a number of interactions between PantryPod and the host PC, which are shown in the diagram in Figure 6.

PantryTech Interactions

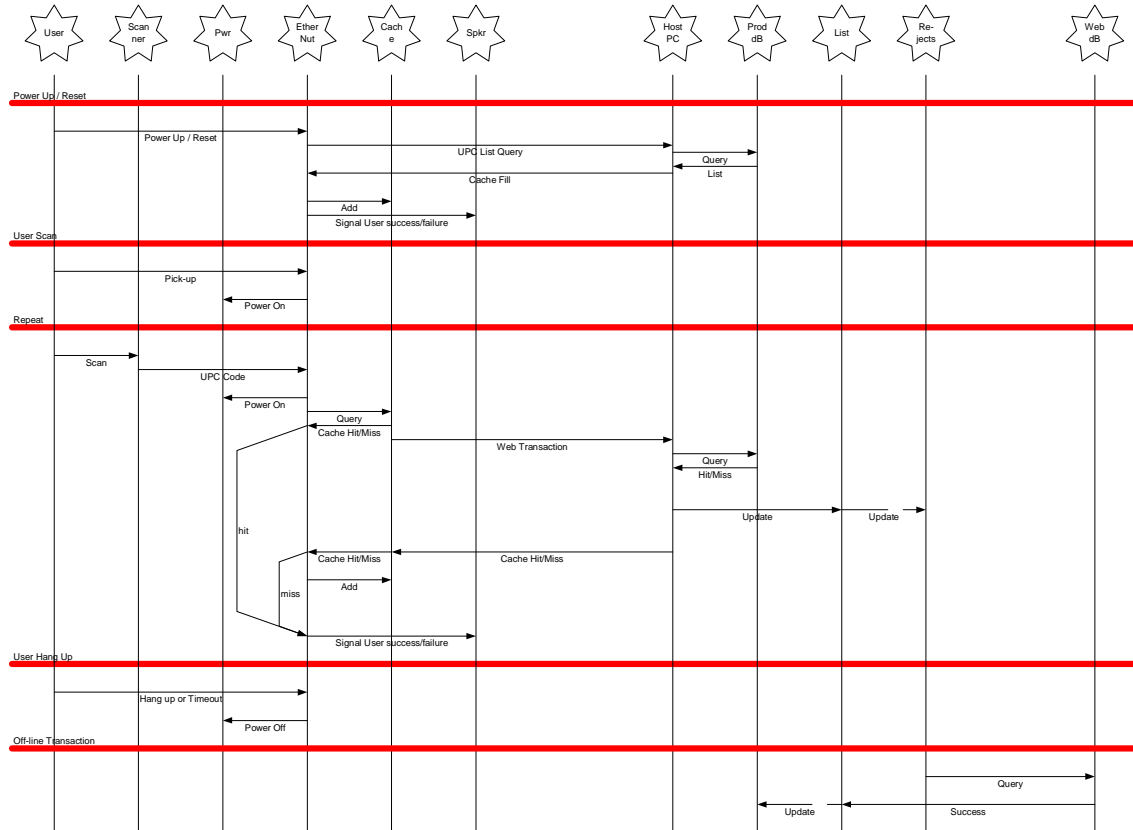


Figure 6 Communication Interactions

When packaged and installed, PantryPod mounts on the wall and it quite easy to use. The scanner parks in a small bay that is part of the package.

