

Project Proposal

Networked Display

03/04/2008

Nicholas Mackie-Jones, nmjgwu@gmail.com

Project Abstract

This is a small status display, showing information acquired over a network connection. Ideally this would be a multicolor graphical display and use a wireless networking connection (802.11g or n). For practicality and cost reasons, the initial design will use a LCD character display and wired ethernet. The data used to populate the display will be pulled from a coordinated webserver, via http over tcp/ip.

Strategy

Description of overall design: See above.

What platform: Zilog ZNeo Z16F chip and the ZNeo Contest Kit as a platform

What external:

Character Display Device – A device capable of multi-line character output. Currently anticipate using the PalmOrb software (<http://palmorb.sourceforge.net/>) on a Palm III-family device to emulate a Matrix Orbital LK204-25, a 4x20 (in chars) display, connected via the serial port. This allows the use of a larger, more complex display without the expense of buying a real one.

Network Module – A module to handle a networking connection that will manage at least some portions of the networking stack itself. As mentioned, the plan is to use a wired ethernet connection. Currently designing around the WIZnet NM7010B+ Wired Ethernet module, which uses a W3150A+ chip to manage a TCP/IP stack.

What capabilities: GPIO, serial IO, timers

What sort of evaluations:

Display: I had originally be hoping for a full, graphical display, but upon doing some product searching this was quickly scaled back to a character display for feasibility and cost reasons. Low-end character devices tended to be 2-line, which would meet my requirements but would not allow for much flexibility, variability, or creativity in the displayed content layout. Larger devices more capable of “artwork” tended to be in the ~\$90 range, which gave me pause. By a combination of conversations and googling, the PalmOrb software was found, which seemed like the best of both worlds. The required hardware (PalmIII-family device) is already owned, and the PalmOrb software aims to emulate one of the '~\$90' larger, more capable devices I had been looking at.

PalmOrb on a PalmIII brings with it a variety of boons and banes. PalmOrb appears to only support RS-232, whereas the real device also supports I²C. PalmOrb has the

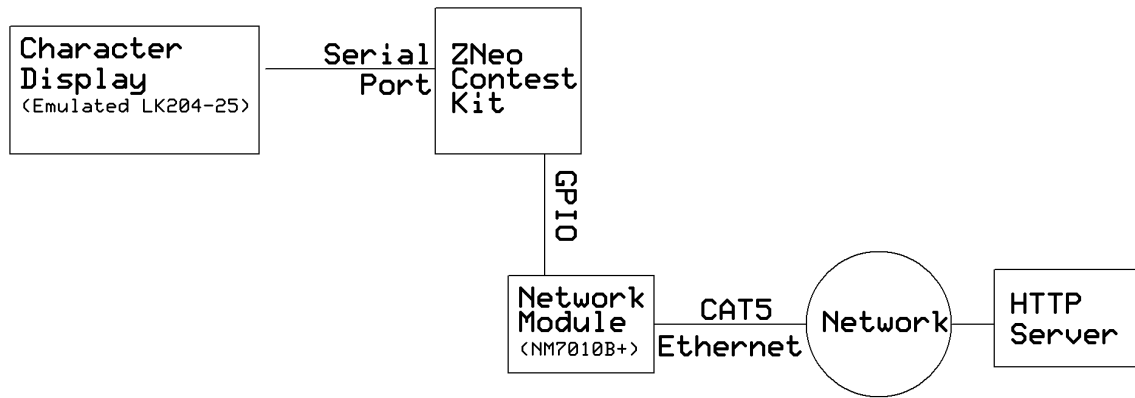
advantage of working on hardware that I want to gain more familiarity with for hobby/hacking purposes, but the disadvantage of bringing with it PalmOS serial bugs which will complicate the coding and the debugging. The PalmIII devices can be powered either via the serial cable or via batteries – while the PalmIII seems to require a 3v supply, it is still unclear whether the rest of the details (amperage requirements vs ZNeo output amperage limits, PalmOrb pin usage) will support powering the display via the ZNeo. Regardless, powering via batteries remains a viable option.

Network: While there was a minimal “selection” process, I still attempted to evaluate the NM7010B+ module for feasibility/practicality, largely as an aspect of determining the overall feasibility of this project. The module's PNY manages all of the physical details, and the W3150A+ manages the majority of the networking stack. The W3150A+ actually exceeds what I was expecting to find, as it implements TCP, managing the Transport and Session OSI layers.

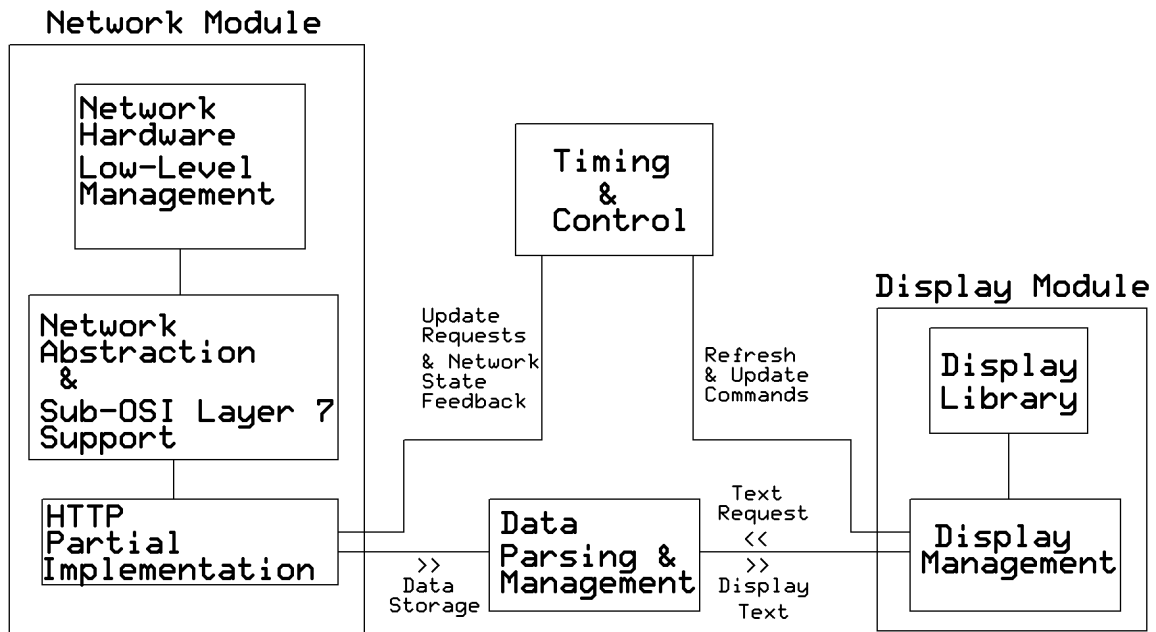
While the NM7010B+ package has 56 physical pins, 11 are NC, 2 are power, and 9 are ground, translating to 34 data signals (56 pins – 11 NC – 2 power – 9 ground). This is within the range of the ZNeo Contest Kit's GPIO ability. The module wants 3.3V VCC, which the ZNeo board can supply. The module supports both a MCU bus and an SPI interface, either of which could be used with the ZNeo.

What Software Modules: The software will be broken into three basic categories: Networking, Control, and Display. The networking module will likely be broken down into three internal segments, reflected in the milestones – a module to manage the hardware requirements, a module to manage the HTTP communication, and a module to bridge between the two. The display module will likely be broken into two parts – a generic display driving library, and a module that will use that library to manage the display for this program. Lastly, the Control module will consist of a timing & control segment to dictate usage of the network connection, react to networking events, and instruct the display. The Control module may also have a separate data management module, to bridge communication between the data received via the network and the data needs of the display, including managing timing conflicts and formatting issues.

Hardware Block Diagram:



Software Block Diagram:



Unknowns

The difficulty of managing the network interface is a rather large unknown. While I do have a familiarity with the kinds of information used to build, pack, and manage packets, and the W3150A+ chip handles much of the network stack itself, the module still requires a finer level of networking control than I am accustomed to. While this lack of familiarity is part of the point of this project, how much time and effort it will take to fully understand the datasheets, and then how much time it will take to implement code using that interface, is one of my larger unknowns. From my examination of the WIZnet NM7010B+ and W3150A+ (chip used by the NM7010B+ for the hardware TCP/IP stack) datasheets, I fully expect it will be doable – the question is how much time it will take, which determines how much time will remain for other facets of the project.

The connection of the NM7010B+ to the ZNeo Contest Kit is also a unknown. Originally I had presumed it would be a simple connection using a breadboard. However, the pin interface of the module is “Two 2.0mm pitch 2 * 14 header pin”, which translates to two long rows, each two pins wide, so that a single-valley breadboard like the ones we were loaned would still have two pins sharing a single electrically-connected column. One possible solution would be to use double-ended 'socket' wires to connect the module to the ZNeo directly, but that would depend heavily on the relative sizes of the wire 'sockets' and the module's pin spacing. The pin-center to pin-center spacing is listed as 2.0mm, and a manual measurement of the wire 'sockets' we have been distributed yields a size of about 1.9mm. While these measurements indicate it sufficient room, physical variabilities in the 'sockets' might overwhelm these small clearances and prevent it from fitting in practice.

The character display is also a bit of an unknown. Using the PalmOrb display device requires several elements – the PalmIII proper, the serial cable to connect the PalmIII to the PC and the ZNeo Contest Kit, the PalmOrb software to emulate the Matrix Orbital device, and the Palm Desktop and Hotsync PC software to load the PalmOrb software onto the PalmIII. I already possess all of these elements, but some of them are less accessible (storage, etc) which has delayed testing of the process. If there is a problem, then I'd need to find another display device, which would likely take notable time both to acquire (ex selecting and shipping) and to modify the design to deal with I/O differences (serial vs parallel connections, etc).

Both the LK204-25 and the NM7010B+ use serial formats for communication. While the general principle seems simple enough, the difficulty of actually implementing and using these protocols remains an unknown to me. It's unlike any kind of I/O I have ever actually used in the past, which makes it hard to judge without actually attempting it. Since serial I/O is fundamental to my project, I am largely banking on the fact that since serial interfaces are common, they must be reasonably manageable to learn and use.

Implementation Plan

There look to be three major steps in this project: successfully interact with the character display, successfully interact with the network module, and successfully use the network module to do transactions over the network.

In the breakdowns to follow, no explicit milestones for serial interface tasks have been included due to their unknown nature. If they seem appropriate, such milestones will be added as needed during development.

Milestones

- Character Display
 - Install and configure PalmOrb
 - Test PalmOrb using PC LCD driver software
 - Test ZNeo-PalmOrb communication
 - Write utility/abstraction library
- Network Module (Hardware and Hardware-managed networking)

- Verify physical connections and primitive functionality via link status LEDs and pins
- Setup the numerous registers used for network/mode settings
- Initial communications testing in ICMP, TCP Client, or TCP Server mode
- Successfully use TCP Client mode, and gain familiarity with it
- Network Communication (Software-managed networking and data management)
 - Simple HTTP communication (Request line sending, status code parsing, minimal headers) and any required sub-application-layer support structure
 - Reasonable HTTP communication (header sending and parsing, payload parsing)
 - Enabling the HTTP-based payload data to populate the Character Display

Character Display – Once the pieces are assembled, loading the PalmOrb software is a straightforward process. To test whether it is functional or not, third-party PC LCD driving software will be used – the current plan is LCD Smartie (<http://lcdsmartie.sourceforge.net/>) which was recommended by the PalmOrb project. Testing the display using a 'known'-good display driver will help distinguish between problems in the overall Palm/PalmOrb system and problems stemming from my code.

Once assured that the PalmOrb setup works, we can move onto communicating with it successfully from the ZNeo. Once communication has been achieved, we can write a display library to simplify usage. This library will be used both for help debugging other portions of the program and for displaying the program's actual intended output.

Network Module – Initially the NM7010B+ module must be physically connected to the ZNeo. Initial confirmation can be done using the link status LEDs (confirming the module is powered and functional) and the link status pins (confirming that the ZNeo is communicating successfully, and that the pin signals match the integrated LED display). Once the connection has been confirmed, we can move to populating the many registers needed for basic setup – the source MAC, source IP, source subnet mask, and gateway address. This is followed by attempts to communicate in ICMP, TCP Client, or TCP server modes to verify the correct settings. The specific mode to use will be determined based on further examination of the hardware, and which seems the most productive choice at the time. Regardless of the mode used for initial testing, we must manage successful TCP Client mode communication next, as that is the mode required for the following milestone. Essentially at this stage, the goal is to master the layers of network communication which the W3150A+ chip on the NM7010B+ manages for us.

Network Communication – With the hardware-managed layers working, we then move on to the software-managed layers built on top. It is expected that this will primarily involve managing the HTTP communication. Initially we will seek signs of accurate communication with an HTTP server by writing and examining only a subset of the HTTP header fields. Additional header fields and response payload will be added incrementally. At some point during this process, a mode to display the actual desired content will be added (ie the Live/Production display mode, as opposed to a debugging

mode). The parsing of incoming header fields will likely initially be hardcoded for the specific network server used in development – dynamic header parsing will follow if resources permit.

Resources

1x ZNeo Z16F Contest Kit – university hardware. Initial plan is to simply use the one already loaned to me for the course. Considering the number of connections I will likely be making to the ZNeo board, a request for a second university kit to dedicate to this project might be prudent.

1x Ethernet module – hopefully a borrowed NM7010B+ hardware module. Failing that, I will need to purchase my **own** instead, likely the NM7010B+ or a similar module due to the time already invested in learning about the NM7010B+ and W3150A+.

1x Display System – hopefully already owned. For the Palm-based emulator, I already have several PalmIII-family devices (largely PalmIII-c if memory serves; a mix of battery types, some using typical batteries and some using internal rechargeable batteries), and several styles of serial cable cradles. The main issue so far has been that they were packed into a relatively inaccessible location, and so I need to invest some time in digging them out. The emulator software to run on the Palms (palmorb.prc) has been acquired, and the software to load the prc onto the Palms (Palm Desktop & Hotsync) is already owned.

If the Palm-based system fails, then I have three tiers of fallback. First, there was a character display mentioned in the hardware available to be borrowed. If it's still available at the time, I could potentially borrow module. Failing that, I could buy a LCD character display device myself – though that would likely be a parallel rather than serial device (for cost reasons), which would complicate the project. As a final fallback, I could use the LED display on the ZNeo Contest Kit (likely using a scrolling marquee or somesuch to enable messages longer than 4 characters to be displayed), though that would be an undesired simplification.

1x PalmOrb Display Test Software – Already acquired. LCD Smartie (<http://lcdsmartie.sourceforge.net/>), a free SourceForge project, will be used to fill this role. If there proves to be an issue with LCD Smartie, my fallback is LCDC (<http://www.lcdc.cc/>), a pay-for program with a shareware trial that is aimed at Matrix Orbital-style devices. Both were suggestions from the PalmOrb documentation.

1x Networked Server – Already owned. My intent is to use a simple webpage served by an Apache 2.2 Http Server, which is already configured and running on a locally- and web-accessible machine. Failing that, a simple custom Java-based server could be used instead.

1x Ethernet Cable – Already owned. (to connect the Ethernet module to the network containing the Networked Server)

1x Network Packet Sniffer – Useful but not required. May or may not actually acquire, depending upon the difficulty of the debugging process, and the difficulty of acquiring a packet sniffing setup. I am aware of free software for Linux, though I do not have a functional Linux machine at present, and I am aware of pay-for software for Windows. Ideally I would find free packet sniffing software for Windows, but failing that I would likely go the Linux route, via a LiveCD distro at worst.