

Networked Objects

Patrick Dwyer

Spring 2006

Week 5 - February 16th

XPort Meet PIC

Now we finally get to interfacing our PIC and our XPort network co-processor. A few details:

- We need to configure the XPort with it's IP address before any of our examples will work.
- Our PIC and XPort communicate using serial communication. Specifically:
 - Port C.6 transmits data from the PIC to the XPort
 - Port C.7 receives data from the XPort to the PIC
 - Our *protocol* for communicating between the PIC and XPort is non-inverted 9600 baud serial. This is the same as the communication protocol we used when configuring our XPort.
 - 9600 Baud
 - 8 Bits
 - No Parity Bit
 - 1 Stop Bit
- The XPort takes longer to start up than our PIC, therefore we need to pause for a few seconds in our PIC code before we try and interface with the XPort. I would recommend between 5 and 6 seconds.
- Just like in Intro to PComp, we'll be using the EPIC programmers, which you can check out from the Equipment Room.
 - To program your PIC chip you need to remove it from your board, just use a small flat head screwdriver to pry it out. The PIC chip is fairly resilient, but still be careful. Just to be safe I would order a few extra 18F252 PIC chips to have on hand.
 - When using the EPIC programming software, you need to manually set the Oscillator preference to **HS** or the PIC won't work properly.
- To test our our PIC and XPort I've setup a special script on the website, which allows us to send data to the site using the PIC/XPort, and check the data using a web browser.
 - The URL to send data too from the PIC is: www.digilutionary.com/classes/netobj/write.py
 - Use the parameter "user" with a value equal to your NetID, or the script won't accept your input.
 - Use the parameter "msg" with a value equal to any message you want to transmit.

- The URL to view with your browser is: www.digilutionary.com/classes/netobj/read.py

IP and Ports

We finally have IP Addresses. Check out the IP Addresses handout.

J2ME, WML & Cell Programming

Few device are more ubiquitous than the cell phone. To access a wide audience of users this makes the cell phone an ideal platform for embedded and wireless device projects. There are a variety of technologies we can use to take advantage of cell phones as a deployment platform, each with their own benefits and pitfalls. We won't go into great depth on any of these in class, but we can research them and discuss their impact on our projects, and directions we can take with any of these technologies if anyone chooses to further explore them.

J2ME

Originally designed for the embedded device market, Java on mobile devices is often referred to as J2ME, or Java 2 Micro Edition. As cell phones and embedded processors have changed rapidly over the last few years, J2ME has splintered into a collection of technologies for creating applications that can be deployed to cell phones. A good introduction to J2ME can be found at:

<http://uberthings.com/mobile/>

This tutorial (by an ITP Alum) will walk you through creating a basic "Hello World" J2ME example that you can run on a Java enabled phone.

J2ME is one of the more powerful technologies we can use for cell phone applications, and has a growing market of supported devices, but can be difficult to develop with. Not all phones support the same functionality, making it hard to program to the lowest common denominator of your audience. The up-side of J2ME is it's ability to work with just about every part of the host cell phone, from networking and SMS, to files, images and sound. Some phones even support 3D graphics in Java.

WML

Phones that support any kind of network interaction will often times have a web browser built into the phone. The quality and support offered by phone browsers varies widely, but most phones support the WML standard. Web Meta Language is similar in concept to HTML, but allows multiple pages (or Cards) to be embedded in a single document. Actions and reactions to form and events is handled differently with WML than with HTML, but WML can be much easier to use that technologies like Java.

The power of WML is quite limited when compared to a technology like J2ME, but if your application doesn't call for low level networking or other features found only in Java, WML may be the answer.

Python

New to the phone programming arena is the Python language. Python is quite common as a server and computer scripting language, but Nokia has released a development kit for creating Python applications on Series 60 phones. You can check which phones belong to this select group here:

<http://www.forum.nokia.com/main/0,,150,00.html?matrixType=s60>

Working with Python tends to be much easier than working with Java, and because this development kit is targeted at a single set of phones, it tends to be much easier to ensure device compatibility with your application. There are fewer resources available for working with Python on the Series 60 phones, but if you have one of these cells, this could be a good option.

Pair Assignment

We work on the Pair Assignment this week. Before the end of class I need the list of people in each group. There should only be one group of three. For more on the Pair Project check out the Pair Assignment handout.

Readings

- Urban Tapestries
 - <http://urbantapestries.net/>
- ActiveCampus
 - <http://activecampus.ucsd.edu/>
- Sonic City
 - <http://www.viktoria.se/fal/projects/soniccity/>
- FLIRT
 - http://www.interaction.rca.ac.uk/research/projects_card/flirt/text.html

Resources

J2ME

- <http://uberthings.com/mobile/>
- <http://developers.sun.com/techttopics/mobility/midp/articles/wtoolkit/>

WML

- <http://www.w3schools.com/wap/>
- http://www.xml.com/pub/rg/WML_Tutorials

Python on Series60

- <http://www.forum.nokia.com/main/0,,034-821,00.html>
- <http://www.onlamp.com/pub/a/python/2005/04/14/s60.html>