Internet Access to an Optical Microscope

A.M. Kuchling
Corporation for National Research Initiatives
Reston, VA 20191-5434
akuchlin@cnri.reston.va.us

Abstract

As part of an effort to establish a distributed MEMS microfabrication environment, CNRI is working on a client/server system for controlling an optical microscope and viewing the results over TCP/IP. The server software is written in Python, and runs on a Linux machine; the client is a Java applet, written using JPython.

1 Introduction

The MEMS project at CNRI is building a distributed network for prototyping MEMS devices; Micro-Electro-Mechanical Systems (MEMS) integrate mechanical elements, sensors, actuators, and electronics on a common silicon substrate. Users will construct designs on their computers, have the designs manufactured at one or more remote fabrication sites, monitor the status of the request, and finally receive the finished devices. It's possible that one single device would undergo manufacturing steps performed at several different fabrication sites.

Users need to be able to examine a device partway through the manufacturing process without having to be at the same physical location as the device. For example, it's possible that one manufacturing step may be destroying a microstructure produced in an earlier step; users need to be able to detect this in order to fix the problem. To this end, we've been working on implementing a client/server system around an optical microscope. In future, we'll work on similar systems for other testing devices, such as wafer probers, profilometers, and scanning electron microscopes.

2 The Automated Microscope

Previous work has been done on remote access to microscopes for surgical pathology ([McCl97]) and inspection of semiconductor wafers. ([Kitt96]). The system we're developing is similar to previous work; it consists of a server that controls the microscope, prevents users from damaging the equipment, and takes snapshots of what the microscope is seeing. Users run a client program that provides a user interface, and displays the images.

The server runs on a Linux machine connected to a Leica INM200 microscope and equipped with a Polaroid Digital Microscope Camera. The server program, written as multithreaded Python code, sends commands to the microscope's hardware over two RS-232 connections, and retrieves images from the camera using the Python Imaging Library and an interface to the SANE scanning library.

We wanted platform independence for the client, so it runs on a Java virtual machine and uses the Abstract Windowing Toolkit for the user interface. Currently the code is written in JPython for ease of development; once the interface has settled down, it'll probably be incrementally ported to Java, simply to shrink the size of the applet that must be downloaded.

At the demo session, you'll be able to see the microscope interface and try it out, though it'll be running over a 33.6 modem connection, which is on the slow side. Performance over a modem connection isn't too bad, at least for a demo; serious users will want to use faster connections, of course.

References

[Kitt96] Somsak Kittipiyakul, *Automated Remote Microscope for Inspection of Integrated Circuits*, MIT Microsystems Technology Laboratories Memo 96-846, October 1996.

[McCl97] Stan McClellan and Tom Winokur, *TelePath: Real-time, remote pathology*, Proc. IEEE SoutheastCon'97, p234-256. `http://etsam2.eng.uab.edu/telepath/sec97.html`