UNIVERSITY OF NOTRE DAME
NDnano
CENTER FOR NANO SCIENCE AND TECHNOLOGY

ND*nano* Undergraduate Research Fellowship (NURF) 2013 Project Summary

1) Student name: Katherine Loughran
2) Faculty mentor name: Dr. Mark Wistey
3) Project title:

4) Briefly describe any new skills you acquired during your summer research:

This summer I learned to program in Molly, a growth control software, in order to create interfaces for the MBE machines inside the semiconductor processing and device fabrication cleanroom in Stinson-Remick. The Molly language is a derivative of C, so I attained a basic understanding of C programming. In order to enter the cleanroom, I was trained in proper gowning techniques and etiquette inside the cleanroom. I also learned the basics of soldering wires.

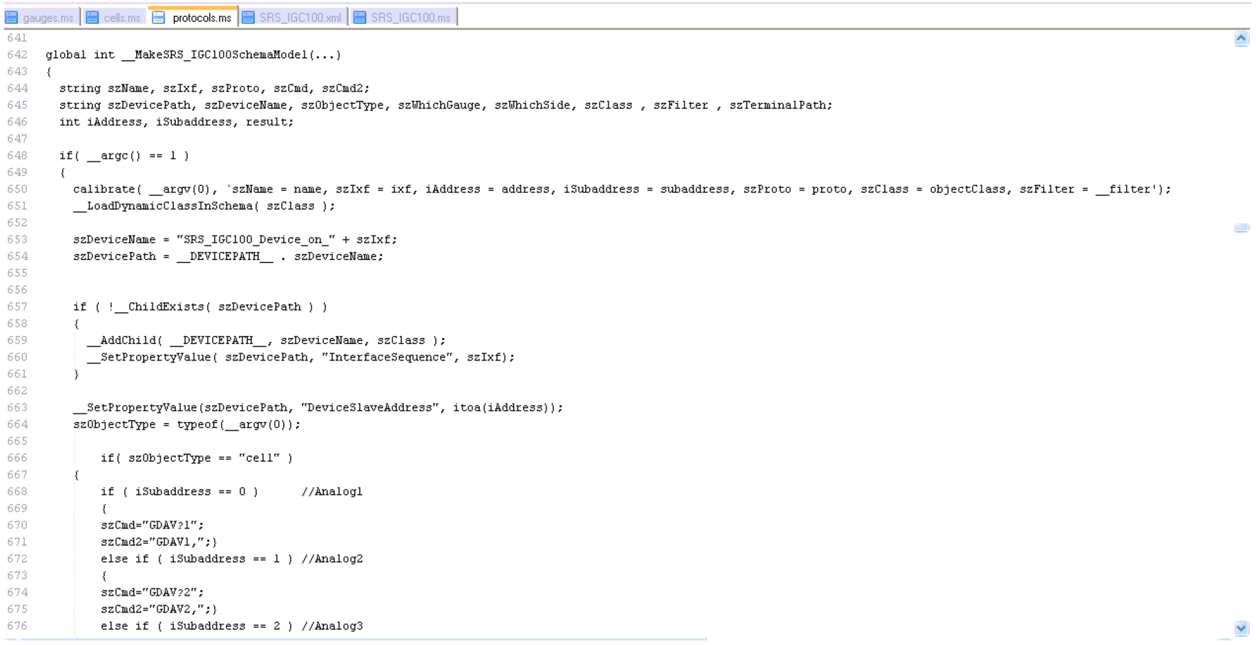
5) Please briefly share a practical application/end use of your research:

My work this summer allows embedded controllers in the Molecular Beam Epitaxy (MBE) machines in the cleanroom to be monitored remotely. The interfaces I created can read and control instruments, make decisions based on the status of the experiment, and send an email if something goes wrong. This will be useful for Dr. Wistey's team as it takes some of the work they would have to do by hand, and allows them to do it remotely via Molly's interface.

Project summary:

I spent my summer creating interfaces from the MBE machines located inside cleanroom to the outside world. There is an ion gauge controller (SRS IGC100) for each of the two MBE machines used by Dr. Wistey's team, and these ion gauge controllers were not remotely controllable. The team uses Molly, software that controls the MBE machines. My goal was to study and use scripts already used by Molly to create a new set of scripts to control the SRS IGC100 by means of Molly's graphical user interface. Although the Molly language is similar to C, Molly's language is particular and had to be learned. How all the scripts Molly uses to back its user interface function was not clear or documented. The plan was to figure out how Molly's scripts work and create my own to control the SRS IGC100 controller.

After becoming accustomed to Molly's graphical user interface, my first activity was writing code to get Molly to send an email based on evaluating the state of an experiment. I then, after connecting the gauge controller to the computer by serial cable, started going through Molly's scripts and wrote code that allowed the user to read the pressure of the ion gauge through Molly's interface. I started on Firehole, one of the MBE machines, and worked my way over to Green, another of the MBE machines in the cleanroom. Working with Green, we wanted a baratron gauge to be read remotely as well. We soldered a cable to connect this gauge to an analog port on the Green's IGC100 controller to be read by Molly. I also wrote code that would troubleshoot the gauge controller and error if anything went wrong. The portion of the project that took the most time was writing code to write to the gauge in order to set the output remotely. Multiple different approaches were taken, starting over and trying new ideas. Trial and error lead me to further understanding of the scripts, and eventually one of my attempts functioned correctly. One of the most time consuming activities of the summer was troubleshooting the code, decoding error messages and determining the root of the problem. Figure 1 below is a screenshot of one of the scripts I was working with. Figure 2 is a screenshot of Molly's user interface on Green.



```
global int __MakeSRS_IGC100SchemaModel(...)
{
  string szName, szIxf, szProto, szCmd, szCmd2;
  string szDevicePath, szDeviceName, szObjectType, szWhichGauge, szWhichSide, szClass , szFilter , szTerminalPath;
  int iAddress, iSubaddress, result;

  if( __argc() == 1 )
  {
    calibrate( __argv(0), `szName = name, szIxf = ixf, iAddress = address, iSubaddress = subaddress, szProto = proto, szClass = objectClass, szFilter = __filter');
    __LoadDynamicClassInSchema( szClass );

    szDeviceName = "SRS_IGC100_Device_on_" + szIxf;
    szDevicePath = __DEVICEPATH__ . szDeviceName;


    if ( !__ChildExists( szDevicePath ) )
    {
      __AddChild( __DEVICEPATH__, szDeviceName, szClass );
      __SetPropertyValue( szDevicePath, "InterfaceSequence", szIxf);
    }

    __SetPropertyValue(szDevicePath, "DeviceSlaveAddress", itoa(iAddress));
    szObjectType = typeof(__argv(0));

        if( szObjectType == "cell" )
    {
        if ( iSubaddress == 0 )      //Analog1
        {
        szCmd="GDAV?1";
        szCmd2="GDAV1,";}
        else if ( iSubaddress == 1 ) //Analog2
        {
        szCmd="GDAV?2";
        szCmd2="GDAV2,";}
        else if ( iSubaddress == 2 ) //Analog3
```

Figure 1. Screenshot of a MollyScript File

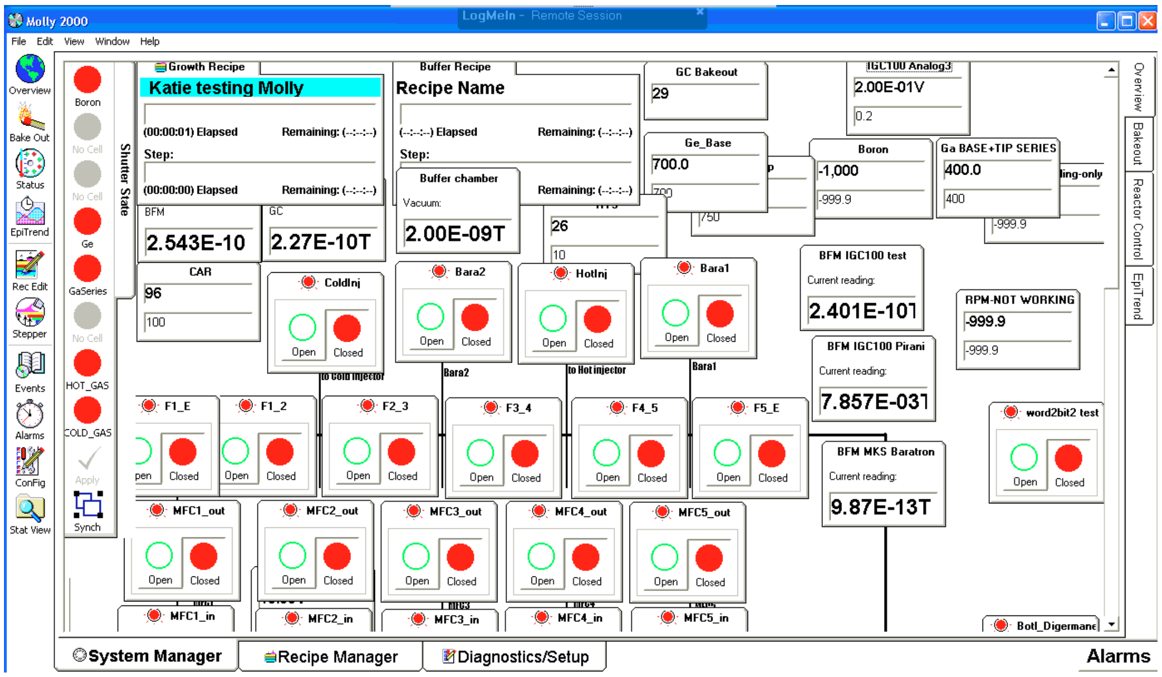


Figure 2. Molly's GUI