Experimental research control software system

I A Cohn, A G Kovalenko, A N Vystavkin
Kotel'nikov Institute of Radio Engineering and Electronics of RAS, 7-11
Mokhovaya str., Moscow, 125009, Russia

Abstract. A software system, intended for automation of a small scale research, has been
developed. The software allows one to control equipment, acquire and process data by means
of simple scripts. The main purpose of that development is to increase experiment automation
easiness, thus significantly reducing experimental setup automation efforts. In particular,
minimal programming skills are required and supervisors have no reviewing troubles.
Interactions between scripts and equipment are managed automatically, thus allowing to run
multiple scripts simultaneously. Unlike well-known data acquisition commercial software
systems, the control is performed by an imperative scripting language. This approach eases
complex control and data acquisition algorithms implementation. A modular interface library
performs interaction with external interfaces. While most widely used interfaces are already
implemented, a simple framework is developed for fast implementations of new software and
hardware interfaces. While the software is in continuous development with new features being
implemented, it is already used in our laboratory for automation of a helium-3 cryostat control
and data acquisition. The software is open source and distributed under Gnu Public License.

1. Introduction
Currently two different approaches are widely used to automate experimental equipment in small
(single PC) experimental setups. Either the whole system is programmed using commercial software
like LabVIEW, Stateflow, etc., or there is a custom solution that is designated to a single experiment.
Both ways have drawbacks, which complicate the automation task.

Commercial software costs money that has better ways to be spent. The major applications use the
combination of a dataflow paradigm with visual programming. Visual programming significantly
complicates supervision and support while the dataflow paradigm is not studied in computer science
class.

Custom solutions are commonly built using general purpose programming languages. This means
that the programmer has to deal with all the interfaces, including hardware, user interaction, output, -
all on his own. Thus, good programming skills are required as well as a significant amount of labor.
Moreover, this approach lacks flexibility. Slight change in the experiment’s algorithm may cause
substantial program changes. Hardware access collisions and stability issues must be solved each time
from scratch.

2. Goals
Here we present a new software solution that lacks the above drawbacks. To meet the requirements of
an experiment automation system that are imposed in our laboratory, a universal experiment control
software system is proposed. Here are the goals to be achieved:
First of all, there should not be a requirement of strong programming skills to automate an experimental setup. Unfortunately experimental physicists are not the best programmers, especially students.

- The system should be portable, to allow easy transition between different platforms.
- There should be an easy way to extend the software to operate new hardware and software interfaces.
- High flexibility in terms of experimental setup automation is essential. New hardware installation, experiment algorithm change etc. should be easily handled, not affecting experiments in action.
- Automation scripts should be easily reviewed and corrected by a supervisor. That would allow handing automation tasks to students without a worry.

3. Architecture
To accomplish those goals we have come up with the following program architecture: the experiment is controlled by separate scripts that run simultaneously in separate threads. The scripts interact with the hardware using a semaphore synchronized layer. Each script can access a private logging interface. In such an arrangement, operating the hardware and logging is quite straightforward, with no worry for access collision and data mess-up.

4. Tools
As the main programming language we have chosen C++. The reasons for that are good OS multithreading and IPC capabilities utilization as well as hardware access library support. To meet portability requirements we have chosen the wxWidgets C++ framework [1]. It provides all of the cross platform abstraction needed to implement our software architecture. Unlike QT it does not use C++ extensions and promises better portability through the wxUniversal implementation.

To compile our code for Windows operating system we have chosen OpenWatcom compiler [2]. Though it hasn’t been changed much in the last couple of years, it is a free open source C/C++ compiler that produces a nice code and is available to everyone. Unlike the current version of Microsoft Visual C Express it runs under windows XP. The alternative is gcc with MinGW [3].
environment. It is open source and free as well, but there is a glitch with symbols been generated in binary files. The compiler adds an underscore before each symbol, so it complicates run-time dynamic linking. For POSIX platforms gcc is an obvious choice.

As the scripting language we have chosen REXX [4]. It was developed by IBM as a glue language to automate mainframes, now it serves us to automate experiments. It is an imperative language, very easy to learn, very forgiving. It has natural means of software integration [5]. Thus there should not be a problem for a student with a beginner programming skill to write a simple script.

5. Features
While the application is in continuous development, today it can already operate with NI hardware interfaces, especially GPIB, via the NI VISA library [6]. There is a separate handler for the serial port as well. Data logging uses CSV format for easier parsing. There also is a simple graphical output facility to display the measurement results in real-time. User scripts can communicate with each other by a messaging system. Finally, a script can start other scripts.

6. The handlers mechanism
In order to provide a convenient way to add new functionality, such as new hardware support or new logging capabilities, there is a software interface based on the concept of separate handlers. A handler is an object that serves a specific task. If it handles a hardware interface that should be shared among user scripts, it would be called “a device”. If it handles a logging mechanism it is called “a logger”.

All handlers are dynamically loaded from shared libraries and are identified by short names. The handler for the serial port is identified with ”SERIAL”. The handler for the graphical logging mechanism is identified with “PLOT”. The GPIB bus as well as VXI, etc. are accessed through the “VISA” handler.

One can build his own handler to replace existing functionality as well as add a new one. The shared library should provide C++ objects inherited from an interface base class. Thus, not more than a basic understanding of virtual function is required to build such a library. The software interface uses pure C++, and the wxWidgets framework is not needed to build a handler library.

7. Operation example
As an example of how our software works, let us take a look at the experimental setup operating in our laboratory. We study superconductor samples at low temperatures, about 0.3K. To reach such temperatures we use a He-3 absorption cryostat. The cryostat cool down process is carried out in several stages. We use a chain of scripts to operate the cool down process. Each script is designated to a separate cool down stage. When the conditions to proceed to the next stage are met, the script starts another script, which operates the next stage. A script that records all of the cryostat sensors data runs in parallel with the cool down control.

We have installed a blackbody radiation source [7] into the cryostat. A separate script runs a PID regulation cycle to set the blackbody temperature at the desired point. When the cryostat reaches the working temperature point, we launch another script that, for instance, measures the IV curves of our samples. As a set of IV curves with different blackbody radiation temperatures is required, the experiment script communicates the blackbody script to set different blackbody temperatures via the messaging system. The cryostat control devices are connected to the computer by the GPIB bus. The IV measurement is done by two devices that use the GPIB bus as well. The blackbody source is controlled via a USB ADC/DAC board. All that communication with different commands to different devices is handled by the application so no hardware access collisions occur.

8. A comparative look
Let us have a closer look at the differences between our software and Labview. A typical Labview programs represents a set of interconnected instruments drivers and program control elements, passing data to each other.
To compose such a program, one should acquire the necessary instrument drivers for his equipment, figure out how to represent the experiment’s algorithm in the dataflow paradigm and connect all the elements in the appropriate way.

In the case of our software, the program is a set of scripts. Instead of creating or downloading an instruments driver, one just has to look for the corresponding command in the device’s manual. In case of Labview the effort is just the same as searching for the appropriate instruments driver in the Internet. The scripts are imperative that makes algorithms implementation straightforward.

One can states that visual programming offers supreme usability. However, studies show [8] that text-style programming is in fact more effective.

9. Further developments
Concerning further developments, we consider the following development objectives to be the most interesting:

- The graphical user interface should be separated from the program core. It is not clear yet, whether it is better to have a separate build with a console-style control, a program interface that loads UIs from a shared library, or some kind of a network protocol that can attach different UIs. Anyway that would give new opportunities to the user, such as using an ARM board for automating the experiment.
- As the dataflow paradigm is quite popular with experimental scientists, we think that a sort of descriptive JSON [9] based dataflow scripting could be implemented.

10. Summary
To summarize, an open source software system has been developed to aid researchers in experiment automation. The program has been designed with an intention to make the experiment automation tasks easy for inexperienced users to accomplish, and clear for supervisors to review. Common hardware interfaces have been implemented and an API is provided that allows the implementation of a new hardware support, as well as introduction of additional output methods.

The software is available on Source Forge http://tirelessworker.sf.net

Reference
[1] http://wxwidgets.org/
[2] http://openwatcom.org/index.php/Main_Pagehttp://www.beercapital.ru
[3] http://www.mingw.org/
[4] http://www-01.ibm.com/software/awdtools/rexx/
[5] http://publibz.boulder.ibm.com/cgi-bin/bookmgr_OS390/BOOKS/E04A2A01/CCONTENTS
[6] http://www.ni.com/visa/
[7] Andrey V. Uvarov, Ilya A. Cohn, Sergey V. Shitov, Alexander N. Vystavkin, Oleg V. Koryukin, Alla G. Kovalenko, Oleg S. Bol’shakov, Vyacheslav F. Vdovin, Cryogenic submillimeter wave blackbody source for tes-bolometer calibration, 2-nd International Conference "Terahertz and Microwave radiation: Generation, Detection and Applications", June 20 - 22, 2012, Abstract book, p.50.
[8] Green, Thomas R. G., and Marian Petre. "Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework." 1996 Journal of Visual Languages & Computing 7 (2) 131-174.
[9] http://json.org/