Computer-Controlled Neutron Diagnostics

Kyle Gibney

Computer Controlled Neutron Diagnostics

K. Gibney
Advised by Dr. Christian Stoeckl

LABORATORY FOR LASER ENERGETICS
University of Rochester
250 East River Road
Rochester, NY 14623

Abstract

In order to upgrade current neutron diagnostic software, and solve problems found within that software, a replacement program has been built using Java. This application is designed to acquire data from FASTBUS time-to-digital converters and export it to a file to be analyzed at a subsequent time. Data is produced using the MEDUSA neutron diagnostic system which is designed to record arrival times of individual neutrons that are emitted in inertial confinement fusion experiments. Neutrons are allowed to pass through an opening in the wall of the target area and strike scintillator detectors housed in a separate building approximately 19 meters away. The signals produced by the scintillators are sent to CAMAC discriminators and stored as encoded data in the FASTBUS TDC's. Former versions of this data acquisition program, written in Visual Basic, have become outdated and incompatible with current versions of the

language. They have also depended on single pieces of hardware that can no longer be purchased, so it was vital to construct a program that could still function on external hardware if the internal card were to become unusable. These problems have been successfully remedied, and the program has been improved internally to allow for easy modification and real time updates of hardware status and the data acquisition process.

Introduction

The University of Rochester's Laboratory for Laser Energetics (LLE) attempts laser driven inertial confinement fusion (ICF) in which deuterium and tritium are compressed to high density and high temperature. As a result of the fusion of these two isotopes an intermediate particle is formed that immediately breaks down into an alpha particle and a neutron. By studying the released neutrons the burning fuel region can be characterized and primary and secondary yields can be measured.

The neutron diagnostic system at LLE, known as MEDUSA (Multi-Element Detector Using a Scintillator Array), depends on software developed in-house to acquire data from hardware specially made for high energy physics applications. Neutrons strike scintillators on MEDUSA that send signals through CAMAC (Computer Automated Measurement and Control) discriminators and end up being stored as encoded data in FASTBUS time-to-digital converters. This data acquisition software, originally written in Visual Basic 4.0, has been heavily updated since the introduction of MEDUSA to the laboratory in 1998 to be used with Visual Basic 5.0, and though it can still get its job

done, it has become obsolete and extremely difficult to maintain. Therefore, a major overhaul was warranted. Knowing the rate at which technology changes, it goes without saying that there have been several other languages introduced in the time that has elapsed, and many of these are much better suited for this type of application than Visual Basic.

One of these emerging languages is Java. Introduced in 1995 under the name OAK, Java has become one of the most widely used development tools for new projects in recent years. Developers have grasped Java for good reason. It introduced many built-in features that were until then left up to the programmer to implement. Among these are multi-threading (parallel processing), portability, garbage collection, and automatic memory allocation. So, since Java has many features that could be utilized in this type of application, it was the logical choice for the language of the replacement program.

MEDUSA Neutron Time-of-Flight Array

The MEDUSA neutron diagnostic system consists of 824 specially made photomultiplier/plastic scintillator detectors that are used to measure fast neutron spectra from laser driven ICF experiments.

MEDUSA is located in a small building roughly 19 meters from the center of the target chamber. The wall around the target chamber has an opening designed to allow neutrons to pass through and strike the scintillator detectors on MEDUSA. The detectors consist of scintillating plastic connected to a photomultiplier tube. Neutrons hit the

scintillating plastic causing protons to be elastically scattered. The protons then pass through the plastic causing photons to be emitted and hit the photomultiplier tube. The photomultiplier tube is designed to turn a small light signal into an electrical signal by taking advantage of the photoelectric effect, in which light hitting a surface causes electrons to be ejected. A sequence of electrodes arranged at increasing voltage provides an amplifier, which turns this initially small current into a measurable pulse of charge that can be recorded.

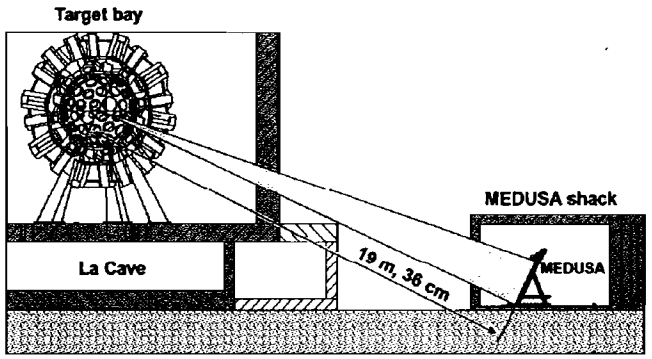

Figure 1

MEDUSA Diagram

The scintillators themselves are powered using a programmable LeCroy 1458 High Voltage system connected to a personal computer using a serial interface (RS-232). The hardware connected to the scintillator array is made up of five CAMAC crates holding 54 LeCroy 4413 discriminators along with a single FASTBUS crate holding nine LeCroy 1877 time-to-digital converters (TDC's). The signal produced by a photomultiplier is sent through a CAMAC 16-channel discriminator with a threshold set to 200 mV and then transferred to a FASTBUS TDC where it is encoded by a Monolithic Time Digitizer, which is an 8-channel, 16-bit dynamic range TDC circuit. At

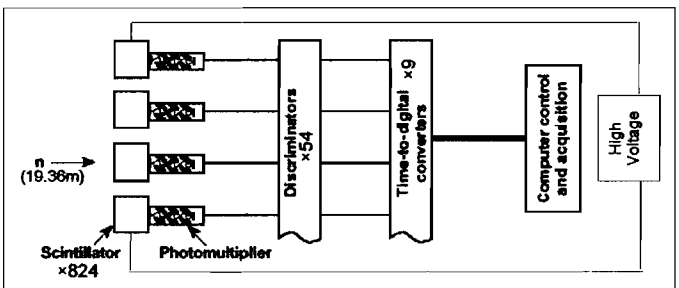

Figure 2

MEDUSA Hardware Setup

this point, after the signals have been digitized, they are ready to be read using the data acquisition program.

Program Function and User Interface

The MEDUSA data acquisition program is meant to read the data from the FASTBUS crate and export it into a file that will be analyzed by a data analysis program. When the program is loaded it automatically checks the hardware that is needed (CAMAC, FASTBUS, HV) and alerts the user if any problems are encountered, such as a CAMAC crate being turned off, or a discriminator module not functioning correctly. Following the hardware check the program remains idle, listening to messages from the OMEGA broadcast server which tells the program at what stage the shot process is in. If the program receives a message it recognizes as a cue, a task will be started that corresponds with that message. For example, a message of "POSTSHOT" would cause the program to begin reading data from the FASTBUS TDC modules and make a file containing this data, or a message of "CHARGE" would cause the the program to perform a precautionary hardware check to ensure that all crates and modules are still in working condition. This means that the program is fully automated and requires absolutely no user input. Following acquisition, data is printed to a file on the local disk and/or UNIX server that matches the format required by the analysis program and the acquisition application returns to an idle state.

The graphical user interface (GUI) was made to be as user-friendly and

intuitive as possible. Although the program is designed to require no user input, aside from being able to select the preferred save path and auto-acquire options, it was still crucial to try and make interacting with the program as effortless as possible. It was important to display information about each shot, program history, and hardware status in places on the window that made viewing easy and understandable. The user interface was not drastically changed from the previous Visual Basic version mainly to prevent an unnecessary retraining of all operators. Because there are many people that need to understand how to operate the program, and most of them are comfortable with the Visual Basic interface, it seemed logical to keep the interface as similar as possible.

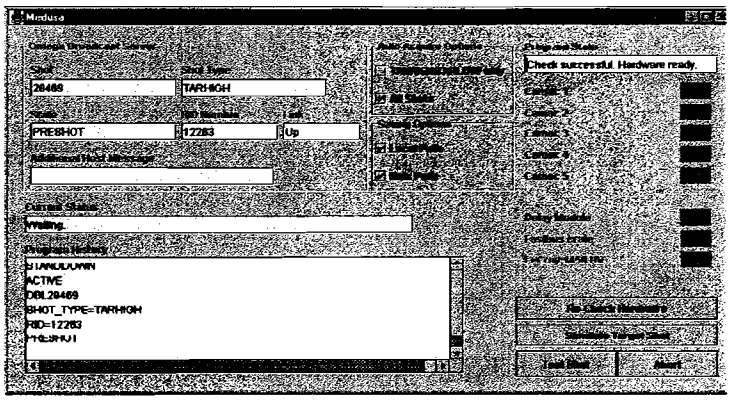

Figure 3

MEDUSA User Interface

Program Structure

One of the big advantages Java brought was the ability to design the program in layers. Layering the classes in Java allows for reusable code and easy modification when it is necessary. Classes written for the data acquisition program can be easily reused as they are in another program that will need similar functions. Easy modification is also important in a program that will need to be maintained by others following completion.

Code needs to be as readable as possible to allow for modification, and again the layering becomes beneficial because classes can be altered independently of the modules above and below, meaning that a slight change in one class will not cause major problems in others around it. Because this program is written in three different languages it is important that it is adequately documented. The majority is done in Java, but the functions needed to control GPIB (General Purpose Interface Bus) needed to be done using the Java Native Interface and C, and assembly microcode was used to control the FASTBUS embedded microcontroller.

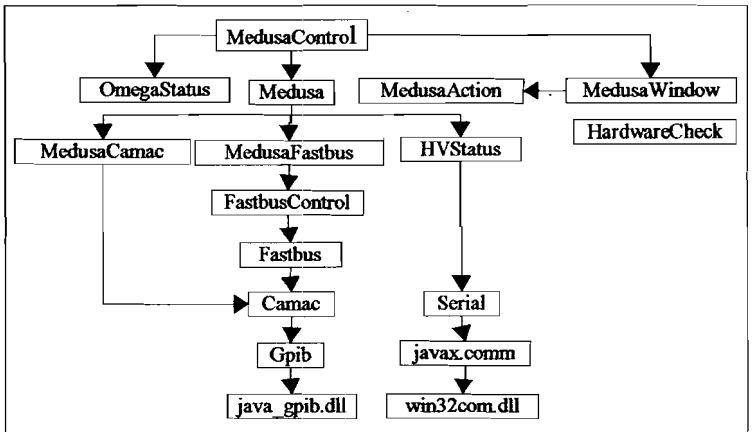

Figure 4

Program Structure (objects in gray provided by Sun)

ADVANTAGES

The MEDUSA data acquisition program, completed in Java, has many advantages that were not found in the Visual Basic version, and it has many possibilities for future use. The updated program takes advantage of multi-threading, so long-running tasks are able to run simultaneously with the user interface with no obvious effect on performance being seen the user. This allows users to see updates in real-time, as tasks are being processed in the background. Within the program updates were made to the error reporting process. If problems are found with hardware, or a save path is non-existent, an

explanatory message is given to the user so the problem can be solved quickly. If a problem with hardware is found, data acquisition is canceled until the errors have been resolved.

One of the major problems with the Visual Basic version of MEDUSA was compatibility. Visual Basic is not backward compatible, so recent versions of Visual Basic (VB 6, VB .NET) are not compatible with the version that MEDUSA was originally written in. Java does not have this problem, so updates to the language are generally well received and easily adaptable.

Another large problem was hardware dependency. One of the goals of the program from the beginning was to avoid being dependent on a single piece of hardware. These devices can no longer be purchased because companies have stopped manufacturing them due to lack of demand, so it is very important that the program can be easily told to utilize different pieces of hardware. If one were to break, it is vital that there is a backup that can be used as a replacement, instead of having no secondary option and having no way to acquire data at that point.

Future Uses

The program is not only designed for use on MEDUSA at LLE, but also possibly on the newly added 1020 Array. The hardware setup for this new detector is very similar to that of MEDUSA, so modifying the program to take data from this device should be fairly simple. Java is also a National Ignition Facility (NIF) approved language,

so if they have a similar setup the data acquisition program could be utilized there as well, again with minimal modification, after the completion of the NIF in 2008.

Conclusion

The development of the replacement program has been successful. The program is currently being used in place of the Visual Basic program while being debugged for errors. This switch to Java has brought many advantages that have made further development of the program much easier, and has made possible porting the program to be used with other systems. The 1020 Array and possibly NIF applications are now options that have been opened up by the development of this new program. Also, most issues found within the previous version were successfully remedied, including hardware dependency and compatibility problems.

Acknowledgments

I would like to thank Dr. Stephen Craxton, director of the Summer High School research program at LLE for giving me this wonderful opportunity and allowing me to work here the summer of 2002. I would also especially like to thank Dr. Christian Stoeckl and Dr. Vladimir Glebov for their amazing patience and willingness to assist me with whatever I asked of them. There is no question that the completion of this project would have been impossible without their explanations and hints.