Advancing *otsdaq*: Enhancements for Usability, Accuracy, and Robustness

A. Mohammed[1] and R. Rivera[2]
[1]*Elgin Community College, CCI Internship*[a)]
[2]*Fermi National Accelerator Laboratory, Advisor*

(*Electronic mail: amoha210@illinois.edu)

(Dated: 24 July 2025)

High-energy physics (HEP) experiments require data acquisition (DAQ) systems that can orchestrate complex detector operations, high data throughput, and responsive, real-time feedback to operators. Traditional DAQ stacks, which are often bespoke, command-line driven and highly specific, impose large learning curves on users. The Off-The-Shelf Data Acquisition (*otsdaq*) framework was created to address these issues by offering a highly customizable and scalable browser-based 'desktop' environment, in which experiment-specific control and monitoring applications can be easily deployed and integrated. Although the initial development of the *otsdaq* software was aimed at the Fermilab Test Beam Facility, *otsdaq* is now being leveraged for broader deployment, including the upcoming Mu2e experiment, where real-time monitoring of field-programmable gate array (FPGA)-based Data Transfer Controllers (DTCs), Clock and Fanout (CFO) boards, and several other subsystems are critical. We contribute a set of targeted improvements to *otsdaq*: bitmap visualization functionality for configured data, improved and corrected delta-based DTC throughput metrics, version control (VC)-backed source navigation for console messages, custom navigation hooks to eliminate disruptive user interface glitches, and copy-to-clipboard support for macro execution history. These changes improve usability, reduce debugging time, and increase accuracy in performance data as Mu2e moves toward commissioning.

I. INTRODUCTION

Modern high-energy physics (HEP) experiments integrate diverse detector subsystems, custom front-end electronics, and high-bandwidth readout links that must be seamlessly orchestrated through a data acquisition (DAQ) layer capable of configuration management, run control, and live, accurate status monitoring. Historically, these DAQ controls have been exposed through bespoke software and scripts or low-level command interfaces. These controls are efficient in expert hands, but pose accessibility barriers for new collaborators, operators, or visiting test-beam users. The off-the-shelf data acquisition (*otsdaq*) project was developed to provide an "off-the-shelf" alternative: A reusable DAQ control environment with a web based, desktop-like user interface that lowers the barrier required to initialize, operate, and debug complex systems.

The *otsdaq* software was designed to be as modular and flexible as possible due to the broad experimental nature at the Fermilab Test Beam Facility (FTBF), and is being integrated with other current and future HEP applications, such as the CMS Outer Tracker production test stand for the CMS experiment at CERN [2] and the upcoming Mu2e experiment at Fermilab [3]. The Mu2e Trigger and DAQ (TDAQ) stack depends heavily on accurate monitoring of interconnected field-programmable gate array (FPGA) hardware elements such as Data Transfer Controllers (DTCs) and Clock and Fanout (CFO) boards (Fig. 1). Usability flaws or misleading performance readouts in the DAQ layer can directly translate to lost beam time or compromised data quality. Consequently, as Mu2e approaches deployment, improving stability, accuracy, and the user experience in *otsdaq* has high operational value.

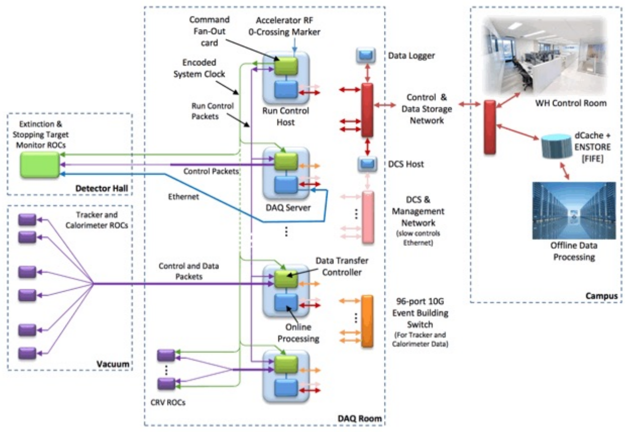


FIG. 1. Complete diagram of the TDAQ sequence [4].

We target high priority performance-relevant bugs and other software gaps observed in ongoing development and test installations. Implementing visualization improvements, correcting performance calculations, strengthening error-navigation tooling, stabilizing user interface (UI) behavior, and making testing results easier to share aim to reduce operator friction and improve diagnostic feedback during both commissioning and running.

II. PURPOSE AND OBJECTIVES

The underlying goal of the *otsdaq* software is to provide a modular and flexible DAQ control surface that can be tailored to specific experiment needs without needing to sacrifice

[a)]Now at The University of Illinois at Urbana-Champaign ease-of-use and functionality with existing DAQ layers, or re-engineering a custom DAQ layer. Within that broader mission, closing usability and reliability gaps uncovered by developers, engineers, and scientists working closely with the *otsdaq* software was heavily exercised in preparation for Mu2e deployment. This work focused on incremental, high-impact improvements, including the optimization of configuration data visualization tools, validation of rate and throughput metrics reported by FPGA-based hardware, enhancement of diagnostic navigation across software versions and installations, and reduction of interface friction that can impede typical usage.

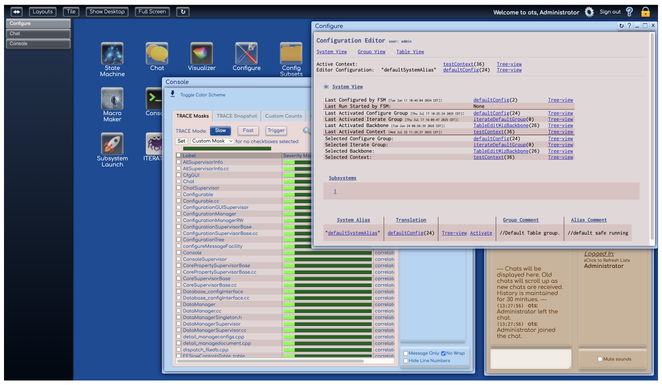
The effort was collaborative across software developers and experiment stakeholders. We partnered with members of the *otsdaq* development team, engineers in the Computational Science and AI Directorate (CSAID) department, and Mu2e personnel representing varied use cases. The completed focused subprojects include bitmap rendering for configuration data, corrected DTC metrics, source lookup for console diagnostics integrated with version control (VC) systems, custom hook behavior to stabilize in-app navigation, and copy-to-clipboard support for macro execution histories.

III. BACKGROUND AND METHODS

A. Overview of the *otsdaq* Framework

otsdaq is a web-based DAQ framework that exposes hardware control, configuration management, run operations, and monitoring through a browser interface designed to decrease the learning curve compared to highly customized DAQ stacks [5]. The FTBF allows scientists and engineers from around the world to work and collect data pertinent to their research, and reducing the onboarding time by common interface convention and on-screen guidance is particularly valuable as it increases the time spent on data acquisition and analysis.

The *otsdaq* software is organized as a web based desktop environment. Each window corresponds to a distinct application, and multiple applications, including multiple instances of the same application, can be active simultaneously (Fig. 2). The software provides a range of tools supporting both basic and advanced usage. Core utilities include configuration editing, run control, and log and console viewing. Advanced tools extend functionality with features such as macro execution, macro building, and data visualization. The modular, application-based architecture allows experiment-specific as well as advanced applications to be added or developed as needed. This enables HEP experiments to create new applications for novel use cases without integrating an entirely new DAQ framework. The *otsdaq* software is currently implemented at the FTBF, the CMS Outer Tracker production test stand for the CMS experiment at CERN [2], and is undergoing extensive testing to prepare for the Mu2e experiment at Fermilab [3]. In addition, the *otsdaq* software can easily be packaged and customized for future experiments around the world.



Configure

Configuration Editor user: admin

System View Group View Table View

```
Active Context:                          testContext(36)    Tree-view
Editor Configuration:  "defaultSystemAlias"  defaultConfig(24)  Tree-view

» System View

Last Configured by FSM [Tue Jun 11 10:45:04 2024 CDT]:        defaultConfig(2)          Tree-view
Last Run Started by FSM:                                      None
Last Activated Configure Group [Thu Jul 17 10:25:24 2025 CDT]: defaultConfig(24)         Tree-view
Last Activated Iterate Group [Thu Jul 17 16:09:47 2025 CDT]:   iterateDefaultGroup(0)    Tree-view
Last Activated Backbone [Tue Jun 24 08:56:39 2025 CDT]:        TableEditWizBackbone(26)  Tree-view
Last Activated Context [Wed Jul 23 11:29:27 2025 CDT]:         testContext(36)           Tree-view
Selected Configure Group:                                     defaultConfig(24)         Tree-view
Selected Iterate Group:                                       iterateDefaultGroup(0)    Tree-view
Selected Backbone:                                            TableEditWizBackbone(26)  Tree-view
Selected Context:                                             testContext(36)           Tree-view

Subsystems
  1
```

System Alias	Translation		Group Comment	Alias Comment
"defaultSystemAlias"	defaultConfig(24)	Tree-view Activate	//Default Table group.	//default safe running

--- Chats will be displayed here. Old chats will scroll up as new chats are received. History is maintained for 30 mintues. ---
(13:27:56) ots: Administrator left the chat.
(13:27:56) ots: Administrator joined the chat.

FIG. 2. The *otsdaq* UI with open apps.

B. Development Environment and Workflow

The *otsdaq* software stack consists of a C++ backend and an HTML/JavaScript frontend, interacting with each other through a CGI-style interface. Development was performed on the Fermilab *correlator* cluster, with builds utilizing *cmake* and *Spack* package management. Source control and collaboration were conducted on GitHub. Debugging relied on standard browser tools, console logging, and side-by-side validation with DTC hardware when validating throughput metrics. Progress depended on regular and frequent interaction with engineers and scientists in the CSAID department.

IV. CONTRIBUTIONS AND RESULTS

A. Bitmap Visualization for Configured Data

Configuration tables within the *otsdaq* software can store bitmap-structured data types. Typical applications of these data types include masks, threshold maps, sensor patterns, calibration, and rapid state auditing. The original interface rendered these data as large, scrollable numeric matrices that took excessive display area to render and were difficult to scan by eye.

We implemented a dynamic bitmap visualization tool that renders table contents as images, where each matrix element is mapped to a pixel representing its value. The feature was implemented within the existing JavaScript layer and integrated with the C++ backend so that stored configuration values remain definitive. We also increased and added new input validation methods and extended save and export behavior to preserve bitmap states without requiring cross-checks against raw numeric values. Performance checks showed no adverse impact on load or render time compared to the prior tabular view.

By optimizing the spatial structure of the UI, the bitmap mode improves scalability to large datasets and accelerates quality checks. The new functionality was merged upstream and is available for future experiment deployments.

B. Corrected DTC Throughput Metrics

DTC modules in the DAQ chain move high-volume data from frontends toward downstream processing. This functionality will be heavily utilized in the Mu2e experiment. During testing, engineers reported that certain metrics, specifically packet rate metrics, were not properly reporting the correct rates, leading to misleadingly low rate estimates and halting the debugging chain. Upon further investigation, it was found that the rate being calculated was derived from cumulative counters rather than differences between successive reads, and information was being recorded although the DTC was not being sent data.

We refactored the rate calculation to use delta-based calculations. These calculations are computed between consecutive sampling intervals and report a much more accurate and instantaneous rate. In addition, the metrics output was reformatted for additional clarity. This refactor was implemented within the C++ backend and kept the display format consistent so as to easily reintegrate with frontend tools. The implementation was checked against live DTC hardware traffic patterns. The corrected metrics enabled collaborators to identify and tune data paths. Notably, the improved readouts were used to optimize data flow and eliminate packet losses across multiple DTCs in test stands. These changes were merged upstream and will directly contribute to reliable data collection in Mu2e operations.

C. VC-Backed Source Navigation in the Console App

The *otsdaq* console application collects and displays warnings, errors, and other informational messages from various services, and attempts to link each message to its responsible source file using a built-in code editor. In mixed deployments, particularly those built from Spack packages where not all source trees are present locally, these links failed, affecting debugging and confusing new users. Notably, this issue occurred in the Mu2e deployment of the *otsdaq* software.

We extended the console functionality to detect when an error message is generated from a file not included in a source tree, and instead construct a URL to the file's upstream GitHub repository, corresponding to the installed package version. This functionality was implemented within both the C++ backend and the JavaScript frontend. Test injections of synthetic warnings, errors, and messages confirmed that links resolve cleanly and open seamlessly. By closing the gap between diagnostics and source context, this feature decreases troubleshooting time and improves support across sites. These changes were merged upstream and are available for future use.

D. Macro Output Copy-to-Clipboard

Macros in the *otsdaq* software allow operators to automate multi-step operations across both hardware and software components. The macro application in the software runs macros and records execution histories and status traces. Previously, sharing this history required manual text selection. This led to formatting complaints from users and occasionally led to errors. Adding functionality to easily export macro history would allow users to include macro logs in documentation, issue reports, or collaborative chats.

We added a copy-to-clipboard action that exports the full macro history in a clean and transferable format. This feature was implemented in the HTML and JavaScript frontend. Validation involved creating synthetic macro runs and comparing the copied test against each of the expected sequences. This enhances the user's ability to collaborate with colleagues and gives increased reproducibility in debugging workflows. This functionality was merged upstream and is available for future deployments.

E. Scroll and Navigation Stabilization

The *otsdaq* software presents multiple apps within framed browser panels, and deep navigation within long configuration pages occasionally triggers abrupt frame shifts, overscroll, or blank whitespace when relying on native HTML anchor behavior. Many users reported unwanted jumps and visual artifacts on several pages. Reducing these visual bugs was critical, as they can not only impede workflows, but potentially can introduce errors into any input fields.

We replaced native anchor-based navigation with a custom JavaScript hook that computes offsets and performs bounded scrolling. The approach was designed to be as universal as possible, such that any future occurrence can easily be fixed by replacing the native functionality with the custom script. This feature was implemented in the JavaScript frontend, and instances of problematic elements in the HTML were replaced during testing. Additional manual testing across several representative pages confirmed the disappearance of the whitespace artifact and stabilized the navigation. The enhancement was merged upstream and is available for future deployments.

V. CONCLUSION AND FUTURE WORK

The *otsdaq* software received a set of focused, user-driven improvements to the *otsdaq* DAQ framework as it advances towards full deployment in the Mu2e experiment. Bitmap visualization, corrected DTC metrics, GitHub source linking, macro log export, and stabilized frontend behavior each address real user complaints surfaced directly from development and testing. The cumulative effect is a more scalable, accurate, and user-friendly control environment for DAQ operations.

Building on these improvements, future development may include automated GUI testing to ensure interface stability, broader visualization modes, settings, and functionalities to accommodate varied use, and integrated dashboards that unify metrics from multiple DAQ subsystems. These enhancements would further support usability, debugging efficiency, and scalability across complex detector subsystems.

Modularity, flexibility, and scalability have increased significantly in the application. Although the *otsdaq* software is designed with a HEP mindset, the software could very easily be applied to different needs. The lightweight nature of the application allows for use in environments where resource efficiency is critical, and the modular behavior allows for users to pick and choose exactly what they need. The *otsdaq* software has potential to be used in several industries, such as manufacturing to revolutionize process automation by providing highly adaptable and efficient monitoring and control solutions.

VI. ACKNOWLEDGMENTS

We gratefully acknowledge Ryan Rivera for his mentorship and guidance throughout this project, and in addition, the CSAID department at Fermilab. This manuscript has been authored by FermiForward Discovery Group, LLC under Contract No. 89243024CSC000002 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics. This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Community College Internships Program (CCI).

[1] S. Hageboeck, A. R. Hall, N. Skidmore, G. A. Stewart, G. Benelli, B. Carlson, C. David, J. Davies, W. Deconinck, J. D. DeMuth, P. Elmer, R. B. Garg, K. Lieret, V. Lukashenko, S. Malik, A. Morris, H. Schellman, J. Veatch, and M. H. Villanueva, "Training and onboarding initiatives in high energy physics experiments," (2023), arXiv:2310.07342 [hep-ex].

[2] S. Chatrchyan *et al.* (CMS), "The CMS Experiment at the CERN LHC," JINST **3**, S08004 (2008).

[3] A. Gioiosa, R. Bonventre, S. Donati, E. Flumerfelt, G. Horton-Smith, L. Morescalchi, V. O'Dell, E. Pedreschi, G. Pezzullo, F. Spinella, L. Uplegger, and R. A. Rivera, "Prototype data acquisition and slow control systems for the mu2e experiment," IEEE Transactions on Nuclear Science **68**, 1862–1868 (2021).

[4] R. Rivera, "Trigger & daq wg report," Mu2e Collaboration Meeting (2018), internal document.

[5] Fermi National Accelerator Laboratory, "otsdaq: Off-the-shelf data acquisition," `https://otsdaq.fnal.gov` (2023), last modified Dec 12, 2023; accessed Jul 22, 2025.