ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-TUPD024

A PYTHON-BASED SERIAL COMMUNICATION FRAMEWORK FOR LEGACY PMAC CONTROLLERS

P. Sreelatha Devi *, A.Balzer †, Helmholtz Zentrum Berlin, Berlin, Germany

Abstract

Many beamlines at BESSY-II still operate using VME-crate-based PMAC motor controllers that rely on proprietary Windows software for communication and configuration. However, these programs often require old licenses and are not compatible with modern operating systems, making maintenance increasingly difficult. To address this, we have developed an open-source Python tool that communicates with legacy PMAC controllers over serial interfaces. The tool uses a modular manager-client architecture where multiple client programs can send commands concurrently without conflicts, using a managed queue and locking system. Dedicated clients have been created for terminal interaction, batch upload of command files, and a watch window monitoring of PMAC variables, with configurable fetch and display intervals. The programs are lightweight, installed via Python package management, and accessible through simple command-line interfaces. While the serial communication is not real-time, it is sufficient for configuration, monitoring, and motion program uploads. Extensive logging is provided for traceability. A GUI interface and integration into the broader Geobrick framework is implemented and tested. This tool provides a modern, open-source alternative for maintaining legacy motion control systems and ensures continued support without reliance on outdated commercial software.

INTRODUCTION

A recent cyber attack at the BESSY-II facility has accelerated the obsolescence problems of software dependencies, including security issues of Windows operating systems. There are continuous attempts to modernize the existing infrastructure [1–3]. However, the long operational history of BESSY II has resulted in a diverse and legacy infrastructure, with many beamlines continuing to rely on discontinued VME based motion systems, some based on Delta Tau PMAC VME controllers [4]. These controllers, though functionally reliable, are increasingly difficult to support due to their dependency on proprietary Windows-based software environments, such as Delta Tau's IDE. These environments are no longer actively maintained, often require outdated license mechanisms, and are tightly coupled to specific operating systems and hardware configurations.

From a system maintenance perspective, this creates significant liabilities. The legacy software is incompatible with modern Linux-based beamline workstations and poses additional risks in terms of cyber-security. In particular, updating or re-configuring these systems often involves manual steps, non-portable tools, and physical access to a Windows machine with the appropriate license—a process that contradicts the goals of modern control system standardization and automation.

To mitigate these limitations and extend the lifecycle of critical beamline infrastructure, we present a lightweight, modular, and platform-independent Python software for interacting with PMAC motor controllers over serial interfaces. The design emphasizes clean abstraction, multi-client coordination, and compatibility with both VME-based PMAC and GeoBrick systems. In contrast to closed IDE solutions, our software is fully open-source, maintainable, and suitable for long-term deployment across the BESSY II facility.

ARCHITECTURE

The developed tool chain follows a modular client-manager architecture, designed for asynchronous serial communication with legacy Delta Tau PMAC motor controllers as illustrated in Fig. 1.

The central component is a Python-based manager process that handles exclusive access to the serial port and mediates all communication between clients and the controller. This approach ensures that concurrent client processes can operate safely without contention or protocol violations.

Communication between clients and the manager is implemented over TCP sockets on the local host. Each client establishes its own socket connection and submits commands prefixed with a functional tag (e.g., TERMINAL, BATCH, WATCHDOG, PLOT). These requests are queued by the manager and serialized using a locking mechanism, ensuring that only one command is processed at a time.

The manager then returns the response back to the requesting client. The backend communication with the PMAC is performed via a lightweight serial protocol using configurable baud rates and port names. A dedicated Python class encapsulates low-level serial I/O, including handling for command-response matching, termination characters, and timeout control. This abstraction allows the system to support multiple PMAC variants (e.g., VME, Geo Brick) without changes to the client programs.

- Each client tool is implemented as a standalone Python script or module:
- The terminal client allows raw command entry and response viewing, with optional decoding of PMAC error/status codes.
- The batch client supports upload of .pmc motion programs, including rudimentary pre-processing of include directives.

* parvathi.devi@helmholtz-berlin.de
† andreas.balzer@helmholtz-berlin.de

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-TUPD024

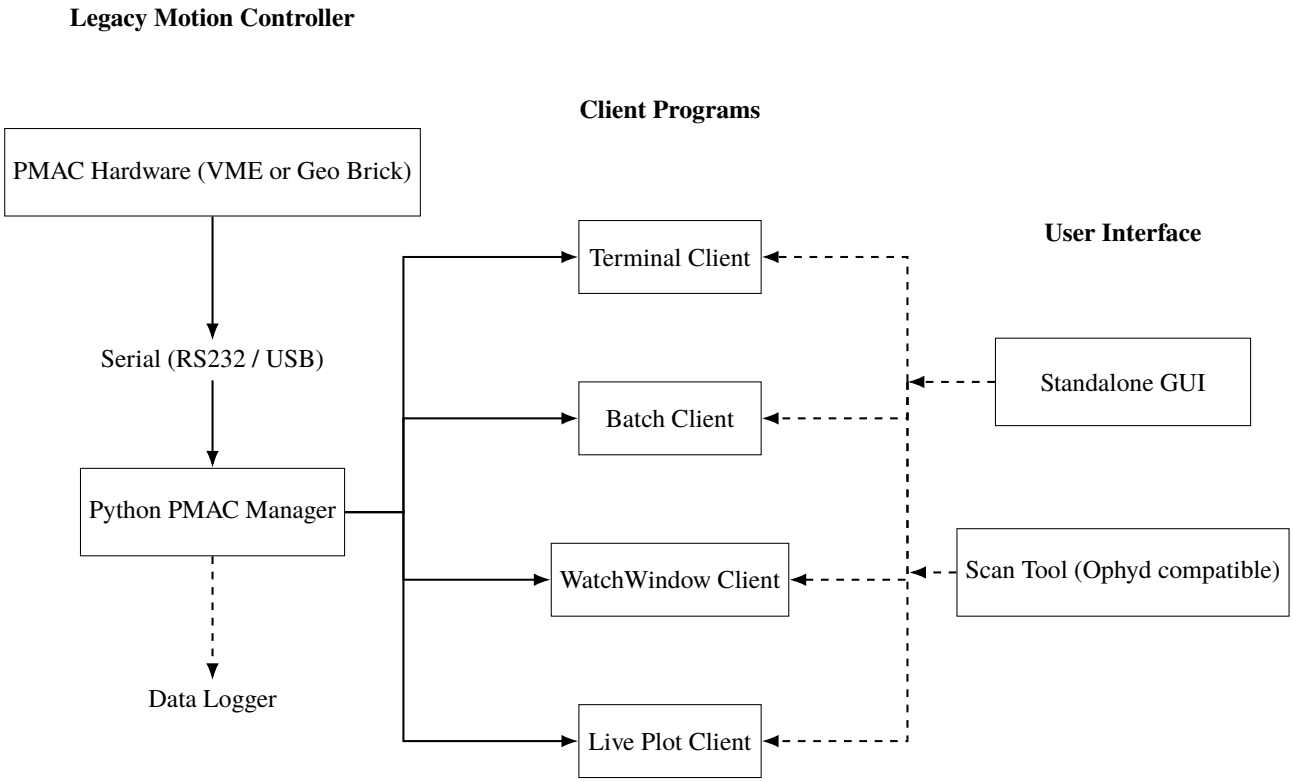


Figure 1: Software architecture.

- The watchdog client reads predefined PMAC variables at fixed intervals, suitable for live monitoring of internal states.
- The plot client requests data periodically for real-time visualization of motor positions or control variables.

Additionally, a graphical interface (based on Tkinter and Matplotlib) integrates all functionality within a single multi-tabbed application. This GUI communicates with the same manager process using separate socket threads per functional module, preserving modularity and preventing cross-interference.

The entire toolchain is dependency-light, with all components developed in Python 3 using only standard libraries and widely available packages. It runs on modern Linux systems and does not require proprietary drivers or runtime environments.

FEATURES AND FUNCTIONALITY

Client Software

The presented Python toolchain aims to provide a robust, maintainable, and user-friendly interface for interacting with legacy PMAC motor controllers at BESSY II. It offers both command-line and graphical utilities designed to support the operational and diagnostic needs of beamline scientists and engineers. Below, we outline the core functionality implemented.

1. Terminal Client

   The terminal client provides a raw command interface, mimicking the traditional Windows-based PMAC terminal programs but with enhanced portability and logging. All user-entered commands are prefixed and routed through the manager, ensuring thread-safe communication. Responses from the controller are decoded using a customizable error parser to improve readability. A scrollable log interface and time-stamped log files assist in diagnostics and auditing. The GUI for terminal is as shown in Fig. 2a.

2. Live Plotting Tool

   A lightweight plotting tool is included to visualize variable trends over time or as XY plots. Users can choose any two variables available in the watchdog list, and real-time data is plotted using Matplotlib. Although the system does not guarantee deterministic real-time sampling, the update frequency is sufficient for general monitoring and system identification tasks. Options for saving data as .csv or figures as .png are also provided. The GUI for plot is as shown in Fig. 2b.

3. Batch Upload Utility

   The batch client supports motion program upload (.pmc files) to the PMAC controller. It handles command-by-command transmission with inter-command delays, and captures response logs. A pre-processor scans for include directives and expands file dependencies

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-TUPD024

recursively. This batch mode is especially useful for restoring default configurations or uploading updated control logic during commissioning and maintenance. The inbuilt buffer generation program provides dedicated support for legacy PMAC user-written servo filters, enabling parameterization, diagnostic analysis, and automatic generation of configuration data files from command-line inputs.
An example snippet of file for upload looks like below:

```
#include "mono.h"
; &2
q93offset=900 ; offset
q94amplitude=4000 ; amplitude
q95signalPeriod=23000;
q41=SYSTEM_ID_MODE
...
```

Upload lookup tables:

```
wx:40701,-11488
wx:40700,-11408
wx:40699,-11329
...
```

The GUI for batch upload is as shown in Fig. 2c.

4. Watch window Monitor

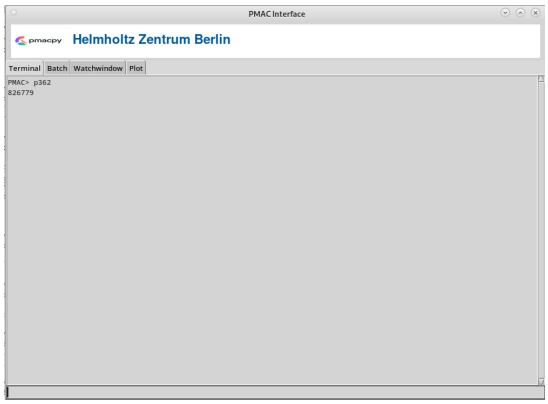
   The watch-window module periodically queries a user-defined list of PMAC internal variables (e.g., positions, status bits, current errors) and displays them in a structured table format. Update rates can be configured, and the readout is live. This facilitates continuous monitoring of critical values without requiring a control system framework like EPICS. The watchdog is also extensible via a text file containing the list of variables to monitor. The GUI for watch-window is as shown in Fig. 2d.

5. Logging and Debugging Support

   All functional clients log their interactions to timestamped log files, categorized by operation type. This enables traceability for debugging and provides a foundation for future integration into higher-level diagnostics frameworks. Errors and exceptions during serial communication are gracefully handled, and clear diagnostic messages are presented to the user.

Graphical User Interface

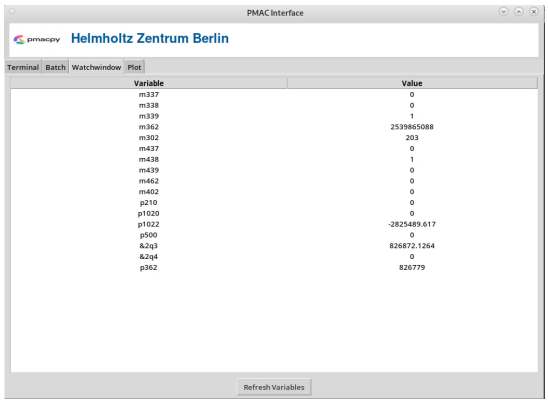
A unified GUI (based on Tkinter) integrates all clients into a tabbed application. Each tab (Terminal, Batch, Watchdog, Plot) runs in its own thread and connects independently to the manager via sockets. The GUI is optimized for clarity and minimal dependencies, making it suitable for deployment on beamline Linux desktops. It also includes institutional branding and supports future extension (e.g., diagnostics, error history, or motion simulation). The GUI windows are as in Fig. 2.



(a) Terminal.

(b) Plot.

(c) Batch.



(d) Watch window.

Figure 2: Graphical user interface windows.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-TUPD024

Hardware Setup

The software was tested with two types of PMAC hardwares used at BESSY-II. The initial attempt was at legacy VME based PMAC2 controller (Fig. 3). The set up with Geobrick LV-IMS-II controller (Fig. 4) was later tested with no further modification on the initial version of the software. Communication was established using serial interface with the PMACpy manager software.

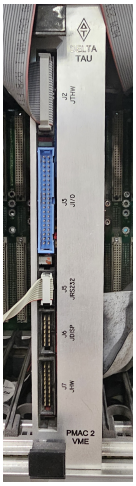


Figure 3: VME-based Delta Tau PMAC2 controller.

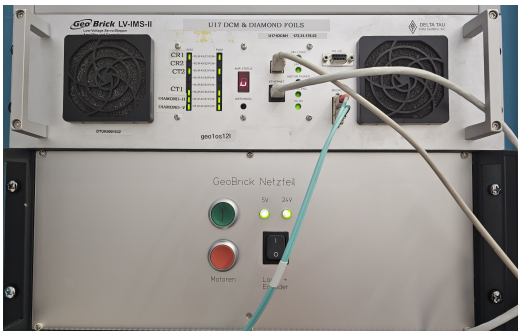


Figure 4: Geo Brick LV-IMS-II controller with external power supply.

DISCUSSIONS AND OUTLOOK

The presented Python-based software tool provides a modular and license-free alternative for interacting with legacy PMAC motor controllers at BESSY II. By abstracting the communication over serial interfaces and offering standalone clients for essential functionalities—such as terminal access, batch uploads, real-time variable monitoring, and graphical inspection—it decouples critical beamline infrastructure from outdated Windows-only software.

From a usability perspective, the tool has been well received by beamline scientists and technicians who benefit from the lightweight installation, transparent logging, and platform independence. Importantly, the GUI application consolidates multiple workflows into a single interface, improving accessibility for non-expert users during commissioning and troubleshooting tasks.

Performance-wise, while the serial communication layer is not suitable for real-time motion control, it has proven adequate for configuration, diagnostics, and monitoring. The communication is also stable when used simultaneously with epics. The managed queue system ensures command integrity across multiple concurrent clients. This architecture avoids race conditions, which is particularly valuable during multi-user debugging or batch uploads.

Looking forward, several extensions are envisioned:

- Support for TCP/IP-based PMAC controllers (e.g., Power PMAC over Ethernet), using the same modular manager-client architecture.
- EPICS integration for selective exposure of PMAC variables into control room systems.
- Waveform acquisition and analysis via Python backends (NumPy/Pandas) to support advanced diagnostics, e.g., following error characterization or tuning feedback gains.
- Include preprocessing in batch uploads, to allow modular and nested .pmc files to be managed more robustly.
- Enhanced security and sandboxing for shared user environments, especially if the tool is to be used on beamline consoles with limited permissions.

This tool demonstrates that even legacy motion systems can be managed with modern, open-source tooling. By reducing dependence on proprietary software and simplifying workflows, the project extends the operational life of legacy hardware while aligning with the broader goals of maintainability and security in accelerator control systems.

CONCLUSION

We have presented a lightweight, open-source Python toolchain designed to support legacy PMAC motor controllers still in use at BESSY II. The system addresses longstanding challenges related to proprietary software dependencies, outdated licensing schemes, and operating system limitations, particularly those associated with Windows-based tools.

By offering a modular manager-client architecture that enables concurrent, conflict-free access to PMAC devices, the tool ensures flexibility for a wide range of tasks including terminal interaction, batch command uploads, real-time variable monitoring, and waveform plotting. These capabilities are further enhanced by a graphical user interface that consolidates workflows for beamline operators and technical staff.

Extensive deployment and testing with both VME-based and Geo Brick PMAC controllers confirm the robustness and portability of the implementation. The project demonstrates how a modern, scriptable, and cross-platform interface can extend the usability of legacy hardware and reduce maintenance overheads in complex accelerator environments.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-TUPD024

Future developments will focus on extending compatibility to Power PMAC over Ethernet, improving batch file pre-processing, and enhancing integration with control system frameworks such as EPICS. In doing so, this software tool aims to offer a sustainable path for continued support of PMAC-based motion systems within evolving accelerator infrastructures.

ACKNOWLEDGEMENT

We would like to thank the beamline scientists at UE112 Low Energy PGM and Normal Incidence Monochromator at BESSY-II for the possible early test of this software solution. We would also like to thank all the engineers in motion controls group who supported this project. This solution is part of the legacy control system support activities at BESSY-II.

REFERENCES

[1] R. Müller *et al.*, "Experimental Data Taking and Management: The Upgrade Process at BESSY II and HZB", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 84-89. doi:10.18429/JACoW-ICALEPCS2019-MOCPL02

[2] W. Smith *et al.*, "Status of Bluesky Deployment at BESSY II", in *Proc. ICALEPCS'21*, Shanghai, China, Oct. 2021, pp. 1064–1068. doi:10.18429/JACoW-ICALEPCS2021-FRBR03

[3] He, Huiling *et al.*, "Bluesky web client at Bessy II", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 518-521. doi:10.18429/JACoW-ICALEPCS2023-TUPDP014

[4] A. Balzer *et al.*, "Diagnostics and Optimization Procedures for Beamline Control at BESSY", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper O1_014.