PyDM DEVELOPMENT UPDATE

J. Bellister, Y. Yazar
SLAC National Accelerator Laboratory, Menlo Park, USA

Abstract

PyDM is a PyQt-based framework for building user interfaces for control systems. It provides a no-code, drag-and-drop system to make simple screens, as well as a straightforward Python framework to build complex applications. Recent updates include expanded EPICS PVAccess support using the P4P module. A new widget has been added for displaying data received from NTTables. Performance improvements have been implemented to enhance the loading time of displays, particularly those that heavily utilize template repeaters. Additionally, improved documentation and tutorial materials, accompanied by a sample template application, make it easier for users to get started.

NEW FEATURES

PyDM has had several feature releases during the past year. These releases include new widgets, better EPICS 7 support, and performance enhancements.

EPICS 7

In order to support the EPICS pvAccess protocol, a new data plugin was created for widgets to communicate with. This plugin uses the PVAccess for Python (P4P) [1] wrapper under the hood to perform the standard set of EPICS requests for interacting with process variables (PVs).

To connect a widget to a device with EPICS 7 support is a very simple change to the address set in the widget. Instead of the channel access prefix, ca://, the prefix pva:// is used in the address. This will route communication between the widget and the data source to the P4P data plugin.

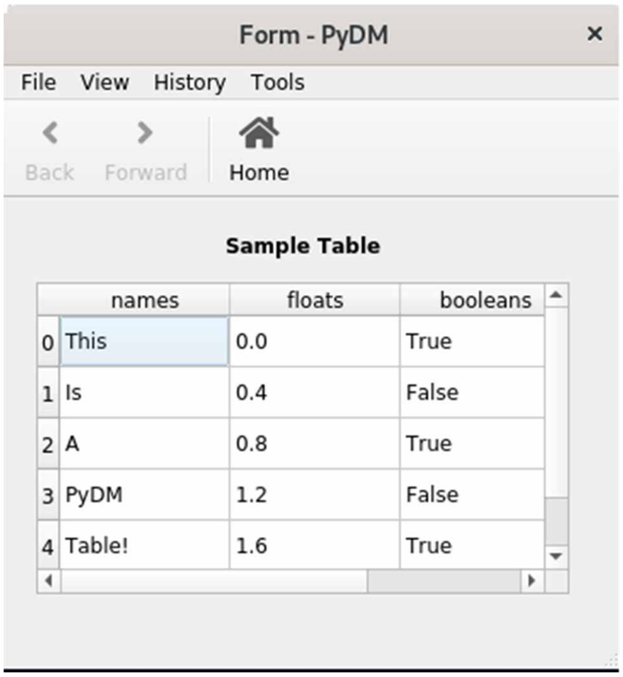
The focus of the new plugin is on the normative types. It also includes an upgrade to the widget for displaying images represented by NTNDArrays. Multiple compression algorithms are supported to provide automatic decompression based on the algorithm specified within the structured data.

New Widgets

One new widget of note is the PyDMNTTable. As can be seen from the name, this widget is for displaying and interacting with data formatted as a table according to the specification in the EPICS 7 normative types document [2]. Like other widgets this table can be used without writing any code, dragging and dropping it into a display from designer. The table can be run in read-only mode, or with writes enabled as well. When writes are allowed, individual cells in the table can be written to and all updates will be reflected in the source PV. It is also possible to pass a subfield of a table to other non-table widgets if only a certain part of the table would be useful to display rather than all of it at once. An image showing a full table is available in Fig. 1.

Another widget that has been added is the PyDMArchiverTimePlot. This widget enhances existing time plots with the ability to communicate with an instance of the EPICS archiver appliance. By setting the timespan on this plot, the widget will call the archiver appliance on startup and request that much historical data to backfill the plot. It is also possible to request archived data while the plot is running by scrolling the x-axis or zooming out on the plot. The signals generated by taking these actions on the plot will invoke calls to the archiver.

To support this new widget, the archiver data plugin was enhanced to make asynchronous calls to the archiver. This prevents calls for large amounts of data from blocking the main PyQt thread and freezing the user interface. The QNetworkAccessManager is utilized to make these non-blocking calls while integrating nicely with the standard signals and slots used by Qt.



Sample Table

	names	floats	booleans
0	This	0.0	True
1	Is	0.4	False
2	A	0.8	True
3	PyDM	1.2	False
4	Table!	1.6	True

Figure 1: The new PyDMNTTable widget.

Usability Enhancements

A number of enhancements have been made to improve the usability of PyDM for both display creators and end users. It is now easier to view EPICS field information on widgets thanks to an upgrade to tooltips. Specifying a field name preceded by a period will cause the tooltip to display the actual value of that field while the display is running.

To further customize their displays, creators can add custom menus to the toolbar to take actions relevant to their displays. An option has also been added to the menu for switching to a new stylesheet while a display is running.

All buttons now support a channel property to make it easier to display alarm sensitive content. This simplifies the process of changing the appearance of related display buttons and shell commands by connecting them directly to a channel, instead of needing to add a second frame widget.

To make displays easier to maintain, notes can now be associated with rules. This makes it clearer what each rule represents. And an option to associate a help screen with any display has been added by just including a text or html file alongside it. When a display is launched with a help file present, PyDM will detect this and automatically add options to both the toolbar and the right-click context menu to display the help file. The QTextBrowser widget is used in order to support both text and html displays.

PERFORMANCE

Improvements to the performance of the application are currently underway. The main focus has been on the time it takes a display to load. One area in which gains have already been made are in displays that make heavy use of the template repeater widget. Caching of the compiled PyQt ui files has resulted in a speedup from 20 to 25 percent.

Further improvements to launch time performance are planned, from providing the option to launch new displays within the same process, to improving the time it takes to establish connections on displays with many PVs.

SUPPORT MATERIALS

An overhaul of the supporting materials associated with PyDM is also in progress. The goal is to make developing and creating displays with PyDM easier at the start, and to provide more useful reference material for more experienced developers.

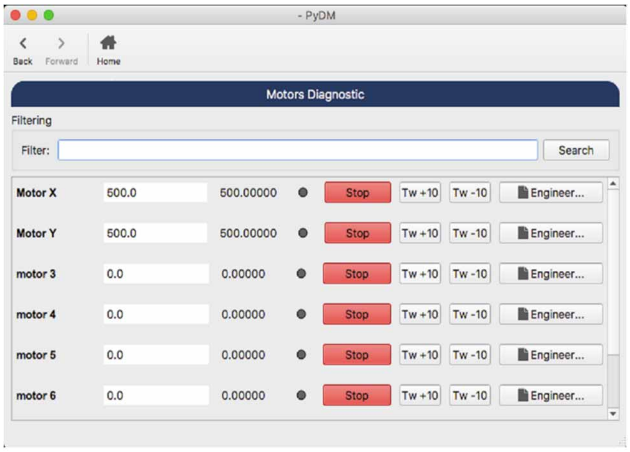
Documentation

A cleanup to make the documentation easier to read has been completed. The methods that are unique to PyDM widgets are now given much more visibility, while links to inherited PyQt methods are provided. Additional examples of how to use each widget are being added so that creating displays with python code is a nicer experience.

Tutorial

The tutorial has been folded in with the rest of the documentation, and revamped to no longer require the download and use of a separate virtual machine. The same environment that runs the tutorial can then be immediately useful in building PyDM displays once the tutorial has been completed.

The documentation for the tutorial itself has been updated to reflect the changes, as well as the images such as the example in Fig 2.



Motor X	500.0	500.00000	●	Stop	Tw +10	Tw -10	Engineer...
Motor Y	500.0	500.00000	●	Stop	Tw +10	Tw -10	Engineer...
motor 3	0.0	0.00000	●	Stop	Tw +10	Tw -10	Engineer...
motor 4	0.0	0.00000	●	Stop	Tw +10	Tw -10	Engineer...
motor 5	0.0	0.00000	●	Stop	Tw +10	Tw -10	Engineer...
motor 6	0.0	0.00000	●	Stop	Tw +10	Tw -10	Engineer...

Figure 2: A screen from the PyDM tutorial.

INFRASTRUCTURE

PyDM now supports Python 3.7 and up only. The migration away from Python 2 was necessary to obtain the latest updates from Python packages that PyDM depends on as more and more projects have dropped Python 2 support in the past few years. The continuous integration (CI) pipelines that PyDM use have also been migrated away from Azure to use GitHub Actions instead.

GitHub Actions

To make future updates to our CI pipelines easier while also requiring significantly less code, we migrated all pipeline files to use GitHub Actions. Because there is a very active developer community around actions, most functionality that is required by PyDM was already available.

PyDM continues to be tested against multiple version of Python, and more than one version of PyQt has been placed into the testing matrix as well. The release process for PyPI was moved to use trusted publishing. And a change to use mamba instead of conda in our building and testing pipelines resulted in a runtime speedup of over 80%.

PyQt Support

In the coming year, updates to the code base will be made to ensure compatibility with PyQt 6. This will allow end users more flexibility in the environment they install PyDM into.

ACKNOWLEDGEMENTS

The authors would like to thank everyone who has contributed code, feature requests, bug reports, and feedback to PyDM.

REFERENCES

[1] P4P, `http://mdavidsaver.github.io/p4p`

[2] Normative Types, `https://docs.epics-controls.org/en/latest/pv-access/Normative-Types-Specification.html#nttable`