Título em Português:	Implementação de um widget Qt de uso científico e de código aberto para facilitar a inserção e manipulação de observáveis físicos
Título em Inglês:	Implementation of an open source Qt widget for scientific use to facilitate the insertion and manipulation of physical observables
Autor:	Breno Henrique Pelegrin da Silva
Instituição:	Universidade de São Paulo
Unidade:	Instituto de Física de São Carlos
Orientador:	Alberto Tannus
Área de Pesquisa / SubÁrea:	Tecnologia e Inovação
Agência Financiadora:	Outros

Implementação de um widget Qt de uso científico e de código aberto para facilitar a inserção e manipulação de observáveis físicos

Breno Henrique Pelegrin da Silva

Daniel Cosmo Pizetta

Alberto Tannús

CIERMag - Instituto de Física de São Carlos / Universidade de São Paulo

breno.pelegrin@usp.br

Objetivos

O *framework* PyMR é um ambiente de controle, gerenciamento e desenvolvimento de sequências de pulsos de ressonância magnética (RM) que utiliza a biblioteca gráfica Qt [1]. No software, é comum inserir diversos aspectos observáveis como tempo, potência e frequência. Todavia, os *widgets* nativos do Qt carecem de recursos de edição voltados ao uso científico. Esse trabalho tem por objetivo solucionar este problema, implementando um *widget* Qt, utilizável pela comunidade Python, que possibilite a inserção de valores com unidades e precisão arbitrária, realize a conversão entre multiplicadores da unidade, que interprete notação científica e facilite a manipulação e visualização dos valores.

Métodos e Procedimentos

O *widget* foi desenvolvido em Python, utilizando bindings do Qt para Python 3. Para o versionamento, foi utilizado o Git. A biblioteca QtPy foi utilizada para permitir a compatibilidade com diferentes implementações do Qt para Python. O projeto consiste em três camadas principais: a lógica gráfica (*Widget*), a lógica do tratamento de observáveis (*Backend*) e a lógica de validação do texto (*Validators*). A manipulação e validação de texto na camada *Validators* foi implementada utilizando expressões regulares. Na camada *Backend*, foi criada uma interface que permite comunicar com diferentes implementações de tratamento de observáveis, possibilitando substituir a implementação sem afetar o funcionamento. Por padrão, é utilizada a biblioteca Pint, devido a sua popularidade na comunidade Python [2].

Na camada *Widget*, foram implementados: a edição de algarismos por meio das teclas *Arrow Up/Down*, interpretação de notação científica, separadores de milhar e conversão entre multiplicadores da unidade. O uso de cada recurso pode ser configurado na inicialização do *widget*. Na Figura 1, é exibido um diagrama do fluxo lógico do projeto e a interação entre as camadas.

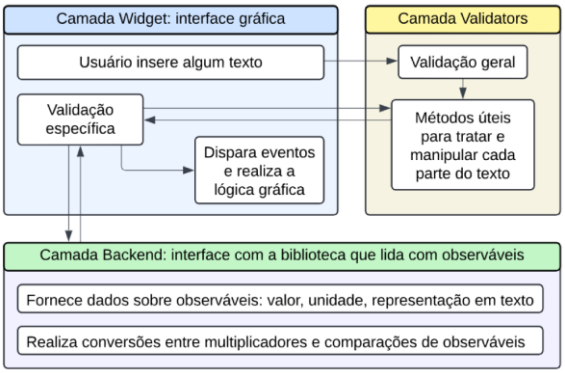


Figura 1: Visão geral do fluxo lógico

Além da implementação do projeto, foram criados ambientes de teste, documentação e implantação utilizando o gerenciador de ambientes Tox e *pipelines* de Integração e Implantação Contínuas (CI/CD) no servidor GitLab. Foram criados testes unitários, documentação automática hospedada no ReadTheDocs, e foi realizada a publicação do projeto no repositório PyPI com o nome de scientific-spinbox.

Resultados

Na Figura 2, é possível visualizar a interface gráfica do projeto, e nela são ilustrados os casos de uso mais comuns das funcionalidades implementadas.

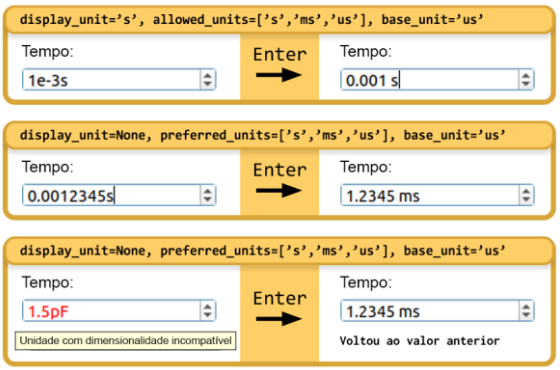


Figura 2: Casos de uso mais comuns do widget

Ao inicializar o *widget*, pode-se escolher entre dois modos de uso: *allowed_units* ou *preferred_units*. O *allowed_units* define uma lista de unidades de mesma dimensão que serão aceitas. O *preferred_units* habilita a conversão automática das entradas para o multiplicador mais próximo em uma lista de unidades preferidas de mesma dimensão. A funcionalidade *display_unit* habilita a conversão automática das entradas para uma unidade específica ao exibir na tela, e pode ser utilizada em conjunto com o *allowed_units*.

Utilizando interfaces gráficas de teste, foi possível observar um aumento na facilidade de inserção de valores, além de uma melhor usabilidade e experiência do usuário. Após implementar os testes unitários e os *pipelines* de CI/CD, foi possível notar que o desenvolvimento se tornou mais ágil e previsível, tornando possível criar novas funcionalidades sem prejudicar o funcionamento de outras partes do código.

Conclusões

A criação do *widget* proporcionou uma melhora significativa na experiência do usuário e facilidade de manipulação de observáveis. A disponibilização pública do projeto no repositório PyPI beneficia a comunidade científica, que pode empregar o *widget* em diversas aplicações, reduzindo o tempo de desenvolvimento delas. Tal implementação trará benefícios para a interface de controle do PyMR, facilitando a interação e evitando possíveis erros de conversão.

Além disso, a utilização das boas práticas de codificação e a configuração realizada dos *pipelines* de CI/CD garantiram um padrão de qualidade para o código e facilitaram a manutenção e continuação do projeto. Futuramente, pretende-se submeter o projeto a equipe do Qt para integrá-lo ao *framework*.

Agradecimentos

Agradeço o financiamento oriundo do acordo entre IFSC/USP e Universidade de Minnesota (UMN), e-convênios 41778, Processo USP/IFSC nº 2017.1, ao Prof. Dr. Alberto Tannús e ao Dr. Daniel Cosmo Pizetta pela oportunidade, direcionamento e orientações.

Referências

[1] Pizetta, D.C. "PyMR: A Framework for Programming Magnetic Resonance Systems". Instituto de Física de São Carlos, Universidade de São Paulo. São Carlos, Brasil. 2018. DOI: <https://doi.org/10.11606/T.76.2019.tde-06052019-103714>.

[2] McKeever, S. et al. "Unit of Measurement Libraries, Their Popularity and Suitability". Universidade de Uppsala. Uppsala, Suécia. 2020. DOI: <https://doi.org/10.1002/spe.2926>.

Implementation of an open source Qt widget for scientific use to facilitate the insertion and manipulation of physical observables

Breno Henrique Pelegrin da Silva

Daniel Cosmo Pizetta

Alberto Tannús

CIERMag - São Carlos Institute of Physics / University of São Paulo

breno.pelegrin@usp.br

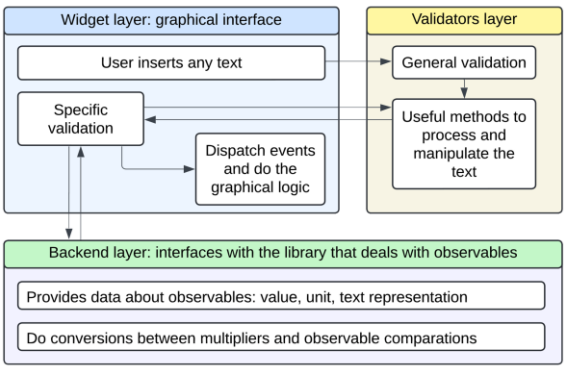
Objectives

The PyMR framework is an environment for controlling, managing, and developing magnetic resonance (MR) pulse sequences that uses the Qt graphics library [1]. In the software, it is common to insert several observable quantities such as time, power, and frequency. However, Qt's native widgets lack editing features for scientific use. This work aims to solve this problem by implementing a Qt widget, usable by the Python community, that enables the insertion of values with units and arbitrary precision, performs conversion between unit multipliers, interprets scientific notation, and facilitates the manipulation and visualization of values.

Materials and Methods

The widget was developed in Python, using Qt bindings for Python 3. For version control, Git was used. The QtPy library was used to enable compatibility with different Qt implementations for Python. The project consists of three main layers: the graphics logic (Widget), the observable handling logic (Backend) and the text validation logic (Validators). Text manipulation and validation in the Validators layer was implemented using regular expressions. In the Backend layer, an interface was created to allow communication with different implementations of observables handling, allowing for the replacement of the implementation without affecting the core functionalities. By default, the Pint library is used, due to its popularity within the Python community [2].

The Widget layer implements the following functionalities: editing numbers using the Arrow Up/Down keys, interpreting scientific notation, thousands separators, and converting between unit multipliers. The use of each feature can be configured on the widget initialization. Picture 1 shows a diagram of the project's logical flow and the interaction between its layers.



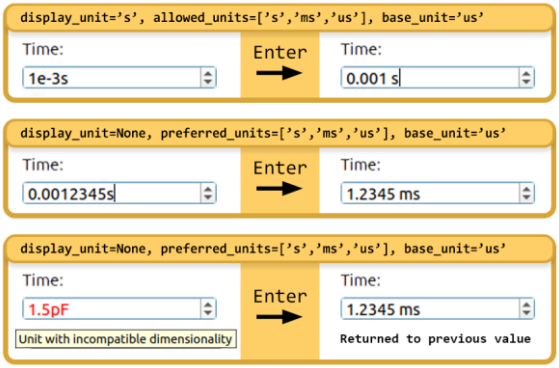
Picture 1: Overview of the project's logic flow

In addition to the project's implementation, the environments for test, documentation and

deployment were created using the Tox environment manager and the Continuous Integration and Deployment (CI/CD) pipelines on the GitLab server. Furthermore, the documentation was created and hosted on ReadTheDocs. Several unit tests were also implemented. The project was published in the PyPI repository by the name of scientific-spinbox.

Results

In Picture 2, it is shown the project's graphical interface, and the common use cases of the implemented functionalities are illustrated.



Picture 2: Common use cases of the widget

In the widget's initialization, one can choose between two usage modes: allowed_units or preferred_units. The allowed_units defines a list of units with the same dimensionality that will be accepted. Conversely, preferred_units enable automatic conversion of inputs to the nearest multiplier in a list of preferred units with same dimensionality. The display_unit functionality enables automatic conversion of inputs to a specific unit when displaying on the screen, and can be used along with allowed_units.

Using test graphical interfaces, it was possible to observe a noticeable reduction in the complexity of value insertion, along with significant enhancements in usability and user experience. After implementing unit tests and CI/CD pipelines, the development process became increasingly more agile and predictable, enabling the creation of new features without affecting the functionality of other parts of the system.

Conclusions

The widget's creation resulted in a significant improvement in the user experience and ease of manipulating observables. The public release of the project in the PyPI repository can surely benefit the scientific community, which is now able to use the widget for various applications, reducing their development time.

This implementation will also bring benefits to the PyMR control interface, facilitating interaction and avoiding possible conversion errors. Moreover, the use of coding best practices and the configuration of the CI/CD pipelines ensures a quality standard for the code and reduces the difficulty of maintaining and continuing the project. For the future, the project is intended to be submitted to the Qt team to be integrated into the framework.

Acknowledgements

I would like to thank the funding provided through the agreement between IFSC/USP and the University of Minnesota (UMN), e-convênios 41778, Processo USP/IFSC nº 2017.1. I would also like to sincerely thank Professor Alberto Tannús and Ph.D. Daniel Cosmo Pizetta for the provided opportunity and guidance.

References

[1] Pizetta, D.C. "PyMR: A Framework for Programming Magnetic Resonance Systems". São Carlos Institute of Physics, University of São Paulo. São Carlos, Brazil. 2018. DOI: <https://doi.org/10.11606/T.76.2019.tde-06052019-103714>.

[2] McKeever, S. et al. "Unit of Measurement Libraries, Their Popularity and Suitability". Uppsala University. Uppsala, Sweden. 2020. DOI: <https://doi.org/10.1002/spe.2926>.