Qudi: a modular python suite for experiment control and data processing

Binder, Jan M.; Stark, Alexander; Tomek, Nikolas; Scheuer, Jochen; Frank, Florian; Jahnke, Kay D.; Müller, Christoph; Schmitt, Simon; Metsch, Mathias H.; Unden, Thomas

Total number of authors:
15

Published in:
SoftwareX

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

SoftwareX 6 (2017) 85–90

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Qudi: A modular python suite for experiment control and data processing

Jan M. Binder [a], Alexander Stark [a,b], Nikolas Tomek [a], Jochen Scheuer [a], Florian Frank [a], Kay D. Jahnke [a], Christoph Müller [a], Simon Schmitt [a], Mathias H. Metsch [a], Thomas Unden [a], Tobias Gehring [b], Alexander Huck [b], Ulrik L. Andersen [b], Lachlan J. Rogers [a,*], Fedor Jelezko [a,c]

[a] Institute for Quantum Optics, Ulm University, Albert-Einstein-Allee 11, Ulm 89081, Germany
[b] Department of Physics, Technical University of Denmark, Fysikvej, Kongens Lyngby 2800, Denmark
[c] Center for Integrated Quantum Science and Technology (IQst), Ulm University, 89081, Germany

ARTICLE INFO

Article history:
Received 25 November 2016
Received in revised form
30 January 2017
Accepted 2 February 2017

Keywords:
Python 3
Qt
Experiment control
Automation
Measurement software
Framework
Modular

ABSTRACT

Qudi is a general, modular, multi-operating system suite written in Python 3 for controlling laboratory experiments. It provides a structured environment by separating functionality into hardware abstraction, experiment logic and user interface layers. The core feature set comprises a graphical user interface, live data visualization, distributed execution over networks, rapid prototyping via Jupyter notebooks, configuration management, and data recording. Currently, the included modules are focused on confocal microscopy, quantum optics and quantum information experiments, but an expansion into other fields is possible and encouraged.

Code metadata

Current code version	0.6
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-16-00092
Legal Code License	GNU General Public License v3
Code versioning system used	git
Software code languages, tools, and services used	Python3
Compilation requirements, operating environments & dependencies	Environment: Anaconda, Python 3.4+, Python packages: comtypes (Windows only), cycler, fysom, gitpython, influxdb, IPython, jedi, jupyter-client, lmfit, lxml, manhole, matplotlib, numpy, PyDAQmx, pycallgraph, pyqtgraph, PyQt4, qtconsole, qtpy, RPi.GPIO (Raspberry Pi only), rpyc, ruamel.yaml, scipy, spidev (Linux only), statsmodels, traitlets, visa, pywin32 (Windows only), zmq
If available Link to developer documentation/manual	https://ulm-iqo.github.io/qudi-generated-docs/html-docs/
Support email for questions	qudi@uni-ulm.de

* Corresponding author.
E-mail address: lachlan.j.rogers@quantum.diamonds (L.J. Rogers).

http://dx.doi.org/10.1016/j.softx.2017.02.001

Software metadata

Current software version	0.6
Permanent link to executables of this version	https://github.com/Ulm-IQO/qudi/releases/tag/v0.6
Legal Software License	GNU General Public License v3
Computing platforms/Operating Systems	Linux, OS X, Microsoft Windows
Installation requirements & dependencies	Environment: Anaconda, Python 3.4+, Python packages: comtypes (Windows only), cycler, fysom, gitpython, influxdb, IPython, jedi, jupyter-client, lmfit, lxml, manhole, matplotlib, numpy, PyDAQmx, pycallgraph, pyqtgraph, PyQt4, qtconsole, qtpy, RPi.GPIO (Raspberry Pi only), rpyc, ruamel.yaml, scipy, spidev (Linux only), statsmodels, traitlets, visa, pywin32 (Windows only), zmq
If available, link to user manual — if formally published include a reference to the publication in the reference list	https://ulm-iqo.github.io/qudi-generated-docs/html-docs/
Support email for questions	qudi@uni-ulm.de

1. Motivation and significance

Modern scientific experiments typically rely on multiple hardware devices working together in a coordinated fashion. In many instances, the hardware devices are commercial products with programming interfaces for direct control via custom software. The unique combination of such devices is then specific to a given experiment. Efficient control of such experiments requires software that is capable of coordinating the operation of multiple devices. In addition, data interpretation is facilitated by rapid data processing and visualization.

These challenges are exemplified when studying color centers in diamond as solid state quantum emitters for sensing, spin manipulation and quantum information technologies. It is typical for such experiments to be performed on a "home-built confocal microscope" [1–5]. As evidenced by the 2014 Nobel Prize in Chemistry, these techniques have expanded beyond the context of physics and now this kind of microscope is pushing advances in biology [6–8] and nanotechnology [9,10]. A wide range of hardware is used for such experiments, but there is a paucity of mature and flexible lab control software to operate the apparatus.

Here, we present Qudi, a Python software suite for controlling complex experiments and managing the acquisition and processing of measurement data. Despite being developed in the context of quantum optics laboratories, the core Qudi framework is broadly applicable to many scenarios involving coordinated operation of multiple experiment devices. The free and open-source nature of Qudi makes it possible for anyone to use and modify the software to fit their research needs, and the modular code design simplifies this task. Qudi continues to be actively developed, but it is already mature enough for reliable laboratory use [11].

2. Software description

2.1. Why Python?

Python was chosen as the programming language for Qudi because of its conceptual synergy with the goals of the project. As a dynamic, strongly typed, scripting language, Python has become a popular choice for scientific programming [12,13] as the importance of scientific software increases [14]. Python's high level of abstraction makes it human-readable and concise, providing a direct advantage for laboratory programming typically performed by scientists rather than dedicated software developers. Source code availability under an open-source license, the built-in modular structure of Python and good community support lower the initial hurdle to learn the language. Additionally, most laboratory hardware has at least an application programming interface (API) specified for the C programming language, which can be accessed by Python.

Scripting languages cannot replace established compiled programming languages for tasks where processing performance or memory efficiency is required but they are very useful to glue together different components in order to benefit from the advantages each of them can offer [15]. This is closely aligned with the concept of Qudi "gluing" together various devices and control methods for specific complex experiments.

2.2. Qudi design

The Qudi suite consists of a collection of modules that are loaded and connected together by a manager component according to settings given in a configuration file as shown in Fig. 1(a). The program startup code and manager were initially derived from similar elements contained within the neurophysiology software ACQ4 [16]. Startup is initiated by a single executable python file, and the manager component provides core functions for logging, error handling, configuration reading, and remote access. Additionally, the manager also administers the other modules by providing functionality for module loading, module dependency resolution and connection, concurrent execution and network access to modules running on other computers. This core infrastructure makes it easier to rapidly develop modules for new experiments by providing structure and starting points.

A typical Qudi session will proceed as follows. On startup, the supervisor process, for example an IDE, creates a Qudi process. In this Qudi process, the manager component reads the configuration file, sets up the log file and loads the modules designated in the startup section of the configuration file. Typically, the startup section will – but does not have to – contain at least the Manager GUI and the tray icon module. Laboratory operation and experiment control are performed by science modules, which are specified in the configuration file along with any hardware-specific parameters. Science modules can be loaded for the desired measurement from the Manager GUI or a Jupyter notebook. Some of the science modules in Qudi were inspired by the pi3diamond software [3–5,17–19].

The science modules are divided into three categories: hardware interaction, experiment "logic", and user interface. These categories and the relationships between them are illustrated in Fig. 1(b). The division into hardware, logic, and interface represents a clear separation of tasks that improves reliability and flexibility of the Qudi code. It also simplifies the implementation of new experiment modules. The fundamental three-fold distinction is at the basis of Qudi's adaptability, and makes Qudi an experiment control software in contrast to a general software framework.

2.2.1. Logic modules

Logic modules control and synchronize a given experiment. They pass input parameters from the user interface to the respective hardware modules, and process measurement data in the desired way. These modules control the information exchange

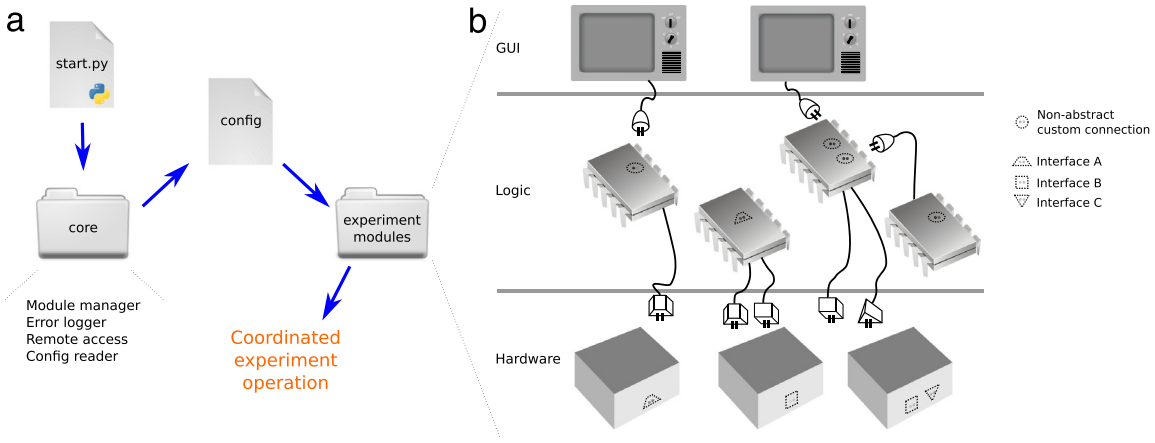

Fig. 1. Qudi functional and structural design. (a) A user launches Qudi by running the start.py file, which loads the core components that take care of configuration parsing, module management, error logging, and remote network access. The module manager reads a configuration file to determine how a set of experiment modules should be configured for the specific laboratory apparatus. Hence, the experiment can be carried out. (b) There is a strongly-enforced three-layer design for all Qudi experiment modules. Specific measurements are written as logic modules, including the required tasks and data analysis. These logic modules connect down to hardware modules via well-defined interfaces, meaning that the experiment itself is hardware agnostic as long as the hardware can fulfill the minimum requirements. GUI modules connect to the logic and provide a way for a user to operate the experiment, as well as a means to display data and calculated results. The careful separation of the GUI from the logic means that it is equally easy to operate experiments in a "headless" scripted manner.

between different hardware modules and perform all necessary computations and conversions.

Logic modules are the only type of modules that are allowed to interact with each other. They are also the only type of module that has its own thread and event loop. Therefore they are the place where concurrent execution of tasks and synchronization of different devices is handled. All steps from the start of a measurement to its end, including data evaluation and storage are performed by the logic. This goes as far as producing "publication ready" plots of data that are saved together with the raw data and which provide a good overview or can be sent to collaborators without post-processing.

2.2.2. Hardware abstraction via interfaces

Today it is possible and even necessary to control most experiment hardware remotely. Unfortunately, the command structure, grammar, measurement units and connection methods differ widely between device models or devices from different suppliers of experiment hardware. To get the most re-usability out of logic modules, it must be possible to interchange hardware modules for measurement devices that provide similar functionality, but work and communicate differently. It is the task of the hardware modules to overcome these problems by translating the commands given by the logic into the "language" of the specific hardware.

The problem is solved by defining an interface, a set of functions that a hardware module of a given type must implement, in order to make a certain measurement work. This set of functions is defined in a class (named ...Interface in a file in the interface folder) where the default implementation of each function raises an exception, if it is not replaced in the device-specific implementation. This class is then inherited by the actual implementing hardware module and all inherited functions must be overwritten.

Hardware modules can represent virtual dummy or mock hardware, which emulates the functionality of a device. Those dummies could load recorded measurement files, create arbitrary data or may perform real physical simulations of measurements, where the result is prepared according to the interface commands which the logic can access. One of the most significant uses of dummy hardware modules is to test the experiment logic without being connected to any actual hardware.

2.2.3. Advanced abstraction via "interfuses"

Building on the abstraction of interfaces, Qudi introduces an additional concept to facilitate the reuse of modules. This ability is provided by interfuse modules which interconnect (or fuse) different hardware or logic modules to modify their interface behavior or to achieve a task for which these modules were not originally designed.

An interfuse is a logic module that implements a hardware interface. In doing so, it pretends to be hardware that can connect to an experiment logic module. This allows the core experiment functions to remain in the logic module, while altering the kind of data that is measured. A tangible example helps clarify this concept. A confocal image (2D array) can represent single fluorescence values from a photon counter for each position (x, y). An interfuse makes it possible to replace the counter data with spectrometer measurements at each pixel, allowing fluorescence to be imaged with arbitrary spectral filtering. This practice improves maintainability and prevents code duplication.

The other reason to use interfuses is where a desired feature would require altering an existing interface definition. For example, an interfuse can perform the coordinate transform to correct for a tilted sample in a confocal scan. As a result, the tilted surface appears flat in the confocal image and can then be imaged at a consistent depth.

2.2.4. GUI

Qudi GUI modules create windows on the screen that a user can interact with, allowing experiment control and data visualization. Their purpose is to offer a convenient way for the user to interact with logic modules, however Qudi is fully functional without the GUI modules. The logic can also be controlled by the integrated IPython console or from a Jupyter notebook. For this reason, GUI modules are not allowed to interact with each other or the hardware directly and they do no data processing.

The Qudi graphical user interface (GUI) is built with Qt [20], offering users a familiar appearance. Qt is suitable due to its multi-platform GUI toolkit that provides good Python bindings [21,22] and makes it possible to separate the GUI design from the implemented functionality. Also, Qt's multi-thread ability ensures good scalability and parallel processing, which are essential requirements for complex experiments. Furthermore, Qt implements a signal-slot mechanism [23] that is very useful for

Table 1
Overview of science modules included in the Qudi suite.

Name	Purpose
Confocal (GUI + logic)	Confocal microscope interface for imaging and positioning scanner.
Optimizer (logic)	Automatically center image scanner on a local signal maximum.
odmr (GUI + logic)	Microwave resonance experiments.
Pulsed (GUI + logic)	Pulse sequence measurements (pulsed laser and/or microwave).
Poimanager (GUI + logic)	Point-of-interest manager for keeping track of multiple measurement spots.
Magnet (GUI + logic)	Driving physical magnet on motorized stages to vary applied magnetic field.
Fit (logic)	Obtain fits for data in various common models (Gaussian, Lorenzian, sinusoidal, etc.).
Counter (GUI + logic)	Perform and display counting tasks of binary events either in continuous or gated way.
Wavemeter_logger (GUI + logic)	Record and process data as a function of laser wavelength as measured by a wavemeter.
Spectrometer (GUI + logic)	Record and display spectrometer data.

concurrency, modular design, and interaction between GUI modules and logic modules. On top of this, the Python library PyQtGraph [24] makes it easy to create interactive, frequently updated 2- and 3-dimensional plots.

The user interface can be edited graphically in Qt Designer and is stored as an XML file. For rapid prototyping, this file can be (re)loaded by a running Python program. The GUI design strives to adhere to the KDE Human Interface Guidelines [25], as these stress the importance of interface familiarity and they work well with the default set of Qt user interface elements.

2.2.5. Interactive scripting

Interactive scripting provides a powerful additional user-interface for a flexible software suite. Qudi contains a built-in console with a fully integrated IPython interpreter. In addition, Qudi can be controlled from a Jupyter Notebook. This makes it possible to write a scripted document with incremental execution as well as inline visualization and analysis. Both the console and the Jupyter notebook can control all of the internal states of the Qudi software. These features enable rapid experiment prototyping, since a developer can test different approaches before committing to changes in hardware or logic modules.

3. Impact and reuse potential

The Qudi suite is useful for any small to medium-size computer-controlled laboratory experiment. Its modular design combined with the use of interface definitions makes it easy to integrate new hardware into an existing experiment. Moreover, this design offers the capability to easily reuse existing modules in new experiments. The Qudi core infrastructure is broadly applicable, even beyond the context of confocal microscopy or physics experiments in general.

Qudi is of more tangible impact to the quantum optics community in particular. The existing modules already offer control over confocal microscopes, electromagnets, motorized stages, lasers, (arbitrary) signal generators, and other devices used in this field of research. Table 1 lists the science modules currently included in the Qudi suite. Furthermore, typical measurement protocols and data analysis functions are already implemented. These existing modules make Qudi a ready-to-use Python-based software suite for quantum optics labs, independent from the individual hardware and measurement schemes used by different groups.

4. Illustrative example

The measurement of optically detected magnetic resonance (ODMR) on single color centers in diamond [26,27] requires the coordinated operation of a scanning confocal microscope and a microwave source. This section describes how such a measurement is performed with Qudi, illustrating the convenience arising from the software design outlined in this paper. The interested reader can perform this process using the default config distributed with Qudi that loads dummy hardware modules to provide representative data. This experiment makes use of the "confocal" and "ODMR" science modules.

The first step is to find a single color center inside the diamond. The Qudi confocal GUI and logic modules are used to move a diffraction-limited focal spot through the diamond sample in three dimensions [28–31]. This is achieved by scanning hardware that is controlled by the confocal logic. A photon counter records the fluorescence measured by the confocal microscope, and this hardware device sends the data to the confocal logic. The confocal logic produces an image of fluorescence as a function of position, and the GUI presents this image to the user. Fig. 2 shows the confocal GUI with an x–y image on the left and an x–z image on the right.

In order to focus on a single center, the user places the confocal cursor near a promising spot. An optimizer module performs a series of close-range scans around the cursor, and the optimal position of maximum fluorescence is found via a fitting module built on the lmfit package [32]. A 2D gaussian fit is performed on the x–y plane scan and for the third dimension a 1D gaussian fit on the z line scan. These are shown in Fig. 2 on the lower right of the Confocal GUI. Finally, the optimizer module moves the scanning hardware to the optimal position focussed on the desired single color center.

In addition to spatial alignment, a microwave resonance condition has to be matched in order to detect the desired change in optical signal [26,27,33–35]. The ODMR logic module controls the frequency of a microwave source while recording the fluorescence level. The design of Qudi means that the ODMR logic is easily capable of driving a variety of microwave source hardware, increasing flexibility in the laboratory.

An ODMR experiment is performed by sweeping the microwave frequency and recording the fluorescence. Recorded data are shown live on screen in the ODMR GUI as both the fluorescence sum of all frequency sweeps and as a matrix plot containing each sweep (Fig. 2, lower right). ODMR scans of several spots can be measured automatically by saving the color center positions and then using a script to move from spot to spot, optimizing the position on each site and recording an ODMR spectrum.

5. Conclusions and future directions

Qudi is a generally applicable experiment control software suite, with infrastructure to support modular design of experiments, significantly reducing the effort involved in constructing new experiments. Qudi already offers a developed quantum-optics tool set capable of reliable laboratory operation, and a modern user interface.

There is continuing effort to expand the library of available science modules. One priority for the future is to simplify the setup of Qudi by providing a graphical configuration editor. Furthermore, it would be convenient to make Qudi installable from the Python

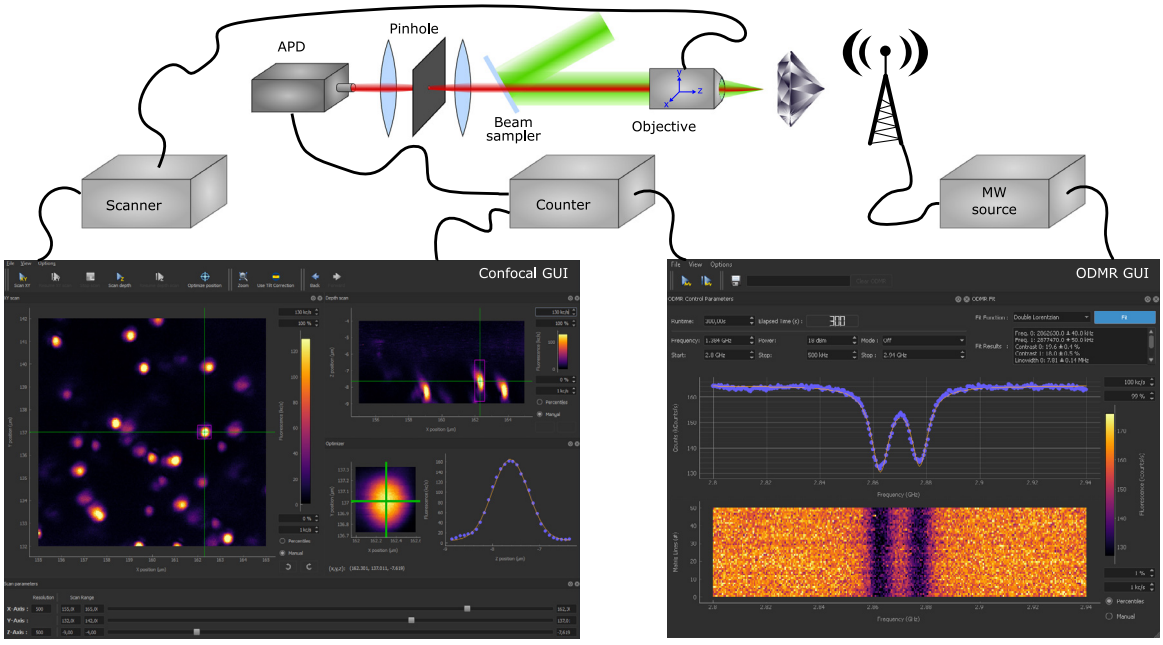

Fig. 2. Simplified illustration of Qudi used to perform ODMR experiments in the laboratory. The experimental setup consists of three main parts. The confocal microscope is used to image the red fluorescence of color centers in diamond using a green excitation laser. The objective can be scanned in all three dimensions by scanner hardware. An avalanche photodiode (APD) detects red fluorescence photons which are counted by digital data acquisition hardware. In addition, a signal generator exposes the color centers to a microwave field which lowers the fluorescence at certain resonance frequencies (ODMR). The Confocal GUI shows the fluorescence images used to position the optical focal spot and the ODMR GUI displays the microwave resonance spectra. This figure illustrates the experience of a user, and does not show the logic modules which perform experimental functions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Package Index. In the context of experiment operation, enhanced automation capabilities are desired to allow a user to rearrange the existing functionality without programming.

Acknowledgments

We would like to thank Boris Naydenov for advocating the use of a software platform that can be used by the whole Quantum Optics institute and for maintaining the predecessor to this software. Furthermore, we would like to thank Ou Wang, Gerhard Wolff, Samuel Müller and Andrea Filipovski for contributions to the software, testing and reporting bugs.

This work was supported by the ERC (ERC-2013-SyG), EU (FP7 Grants 667192, 611143) projects (SIQS, DIADEMS, EQUAM), DFG (FOR 1493 and SFBTR 21), BMBF (Project Q-Com), the Volkswagen foundation, the Innovation Foundation Denmark (EXMAD project no. 1311-00006B and Qubiz) and the Danish Research Council for Independent Research (DIMS project no. 4181-00505B, Individual Postdoc and Sapere Aude, 4184-00338B).

References

[1] Jelezko F, Gaebel T, Popa I, Gruber A, Wrachtrup J. Observation of coherent oscillations in a single electron spin. Phys Rev Lett 2004;92(7):076401. http://dx.doi.org/10.1103/PhysRevLett.92.076401.

[2] Balasubramanian G, Neumann P, Twitchen D, Markham M, Kolesov R, Mizuochi N, et al. Ultralong spin coherence time in isotopically engineered diamond. Nature Mater 2009;8:383–7. http://dx.doi.org/10.1038/nmat2420.

[3] Fedder H, Dolde F, Rempp F, Wolf T, Hemmer P, Jelezko F, et al. Towards T1-limited magnetic resonance imaging using Rabi beats. Appl Phys B 2011; 102(3):497–502. http://dx.doi.org/10.1007/s00340-011-4408-4, URL http://link.springer.com/article/10.1007/s00340-011-4408-4.

[4] Waldherr G, Beck J, Steiner M, Neumann P, Gali A, Frauenheim T, et al. Dark states of single nitrogen-vacancy centers in diamond unraveled by single shot NMR. Phys Rev Lett 2011;106(15):157601. http://dx.doi.org/10.1103/PhysRevLett.106.157601, URL http://link.aps.org/doi/10.1103/PhysRevLett.106.157601.

[5] Dolde F, Bergholm V, Wang Y, Jakobi I, Naydenov B, Pezzagna S, et al. High-fidelity spin entanglement using optimal control. Nature Commun 2014;5: 3371. http://dx.doi.org/10.1038/ncomms4371, URL http://www.nature.com/ncomms/2014/140228/ncomms4371/full/ncomms4371.html.

[6] Röcker C, Pötzl M, Zhang F, Parak WJ, Nienhaus GU. A quantitative fluorescence study of protein monolayer formation on colloidal nanoparticles. Nature Nanotechnol 2009;4(9):577–80. http://dx.doi.org/10.1038/nnano.2009.195, URL http://www.nature.com/nnano/journal/v4/n9/full/nnano.2009.195.html.

[7] Grotjohann T, Testa I, Leutenegger M, Bock H, Urban NT, Lavoie-Cardinal F, et al. Diffraction-unlimited all-optical imaging and writing with a photochromic GFP. Nature 2011;478(7368):204–8. http://dx.doi.org/10.1038/nature10497, URL http://www.nature.com/nature/journal/v478/n7368/full/nature10497.html.

[8] Göttfert F, Wurm CA, Mueller V, Berning S, Cordes VC, Honigmann A, et al. Coaligned dual-channel STED nanoscopy and molecular diffusion analysis at 20 nm resolution. Biophys J 2013;105(1):L01–3. http://dx.doi.org/10.1016/j.bpj.2013.05.029, URL http://www.sciencedirect.com/science/article/pii/S0006349513006127.

[9] Harke B, Keller J, Ullal CK, Westphal V, Schönle A, Hell SW. Resolution scaling in STED microscopy. Opt Expres 2008;16(6):4154–62. http://dx.doi.org/10.1364/OE.16.004154, URL http://www.osapublishing.org/abstract.cfm?uri=oe-16-6-4154.

[10] Hell SW, Schmidt R, Egner A. Diffraction-unlimited three-dimensional optical nanoscopy with opposing lenses. Nature Photonics 2009;3(7):381–7. http://dx.doi.org/10.1038/nphoton.2009.112, URL http://www.nature.com/nphoton/journal/v3/n7/full/nphoton.2009.112.html.

[11] Jantzen U, Kurz AB, Rudnicki DS, Schäfermeier C, Jahnke KD, Andersen UL. et al. Nanodiamonds carrying quantum emitters with almost lifetime-limited linewidths, arXiv:1602.03391 [cond-mat, physics:physics, physics:quant-ph] URL http://arxiv.org/abs/1602.03391.

[12] Perkel JM. Programming: Pick up Python. Nature 2015;518(7537):125–6. http://dx.doi.org/10.1038/518125a, URL http://www.nature.com/doifinder/10.1038/518125a.

[13] Cass S. The 2015 Top Ten Programming Languages (Jul. 2015) URL http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages.

[14] Singh Chawla D. The unsung heroes of scientific software. Nature 2016; 529(7584):115–6. http://dx.doi.org/10.1038/529115a, URL http://www.nature.com/doifinder/10.1038/529115a.

[15] Ousterhout JK. Scripting: higher level programming for the 21st century. Computer 1998;31(3):23–30. http://dx.doi.org/10.1109/2.660187.

[16] Campagnola L, Kratz MB, Manis PB. ACQ4: an open-source software platform for data acquisition and analysis in neurophysiology research. Front Neuroinform 2014;8(3): http://dx.doi.org/10.3389/fninf.2014.00003, URL http://www.frontiersin.org/neuroinformatics/10.3389/fninf.2014.00003/abstract.

[17] pi3diamond source code, URL https://github.com/HelmutFedder/pi3diamond.
[18] Dolde F, Fedder H, Doherty MW, Nöbauer T, Rempp F, Balasubramanian G, et al. Electric-field sensing using single diamond spins. Nature Phys 2011;7(6):459–63. http://dx.doi.org/10.1038/nphys1969, URL http://www.nature.com/nphys/journal/v7/n6/abs/nphys1969.html.
[19] Michl J, Teraji T, Zaiser S, Jakobi I, Waldherr G, Dolde F, et al. Perfect alignment and preferential orientation of nitrogen-vacancy centers during chemical vapor deposition diamond growth on (111) surfaces. Appl Phys Lett 2014;104(10):102407. http://dx.doi.org/10.1063/1.4868128, URL http://scitation.aip.org/content/aip/journal/apl/104/10/10.1063/1.4868128.
[20] Qt—Cross-platform application development for desktop & embedded, https://www.qt.io/. [Accessed 7 August 2016].
[21] Riverbank—Software—PyQt—What is PyQt? https://riverbankcomputing.com/software/pyqt/intro. [Accessed 7 August 2016].
[22] PySide, https://wiki.qt.io/PySide.
[23] Signals & Slots in Qt 5, URL http://doc.qt.io/qt-5/signalsandslots.html.
[24] PyQtGraph—Scientific Graphics and GUI Library for Python, http://www.pyqtgraph.org/. [Accessed 7 August 2016].
[25] KDE Human Interface Guidelines, https://community.kde.org/index.php?title=KDE_Visual_Design_Group/HIG&oldid=72475. [Accessed 11 October 2016].
[26] Gruber A, Dräbenstedt A, Tietz C, Fleury L, Wrachtrup J, Borczyskowski Cv. Scanning confocal optical microscopy and magnetic resonance on Single Defect Centers. Science 1997;276(5321):2012–4. 00662, http://dx.doi.org/10.1126/science.276.5321.2012, URL http://www.sciencemag.org/content/276/5321/2012.
[27] Jelezko F, Wrachtrup J. Read-out of single spins by optical spectroscopy. J Phys: Condens Matter 2004;16(30):R1089. 00084, http://dx.doi.org/10.1088/0953-8984/16/30/R03, URL http://iopscience.iop.org/0953-8984/16/30/R03.
[28] Pawley J, Masters BR. Handbook of biological confocal microscopy. Opt Eng 1996;35(9):2765–6.
[29] Beveratos A, Brouri R, Gacoin T, Poizat J-P, Grangier P. Nonclassical radiation from diamond nanocrystals. Phys Rev A 2001;64(6):061802.
[30] Sipahigil A, Jahnke KD, Rogers LJ, Teraji T, Isoya J, Zibrov AS, Jelezko F, Lukin MD. Indistinguishable photons from separated silicon-vacancy centers in diamond. Phys Rev Lett 2014;113(11):113602.
[31] Häussler AJ, Heller P, McGuinness LP, Naydenov B, Jelezko F. Optical depth localization of nitrogen-vacancy centers in diamond with nanometer accuracy. Opt Exp 2014;22(24):29986. http://dx.doi.org/10.1364/OE.22.029986, URL https://www.osapublishing.org/oe/abstract.cfm?uri=oe-22-24-29986.
[32] Newville M, Nelson A, Ingargiola A, Stensitzki T, Allan D, Micha. et al. lmfit-py 0.9.5 (Jul. 2016). http://dx.doi.org/10.5281/zenodo.58759.
[33] Nizovtsev A, Kilin SY, Neumann P, Jelezko F, Wrachtrup J. Quantum registers based on single NV+ n 13C centers in diamond: II. Spin characteristics of registers and spectra of optically detected magnetic resonance. Opt Spectrosc 2010;108(2):239–46.
[34] London P, Scheuer J, Cai J-M, Schwarz I, Retzker A, Plenio MB, et al. Detecting and polarizing nuclear spins with double resonance on a single electron spin. Phys Rev Lett 2013;111(6):067601. 00011, http://dx.doi.org/10.1103/PhysRevLett.111.067601, URL http://link.aps.org/doi/10.1103/PhysRevLett.111.067601.
[35] Dolde F, Jakobi I, Naydenov B, Zhao N, Pezzagna S, Trautmann C, et al. Room-temperature entanglement between single defect spins in diamond. Nature Phys 2013;9(3):139–43.