Введение

Целью прохождения производственной практики в АО “РПКБ” (Раменское приборостроительное конструкторское бюро) является закрепление и формирование умения применять знания, которые были получены за время теоретического обучения, на основе практического участия в деятельности предприятия. АО “РПКБ” – российское проектно-конструкторское бюро, ведущий в России разработчик бортового радиоэлектронного оборудования для летательных аппаратов всех типов.

Основным видом деятельности предприятия являются научные исследования и разработки в области естественных и технических наук. В частности, организация занимается разработкой и производством авиаприборов, бортового оборудования, беспилотников, а также морских и наземных транспортных средств.

Отдел НИО-ПП занимается разработкой программного обеспечения для бортовых средства объективного контроля. Объективный контроль – комплекс мероприятий по сбору, обработке и анализу инструментально-регистрируемой информации о работоспособности авиационной техники и наземных средств обеспечения полетов. Использование бортовых средств объективного контроля (СОК) позволяет объективно оценить состояние авиадвигателей, бортового оборудования, действия в полёте лётного состава, полноту и качество выполнения полётного задания. Важное место в решении этих задач, а также при расследовании авиационных происшествий и инцидентов занимают бортовые системы регистрации полётных данных (БСРПД).

Ключевым объектом БСРПД является бортовой самописец – конечное устройство системы регистрации, используемое в авиации для записи основных параметров полёта, внутренних показателей функционирования систем летательного аппарата. Бортовые самописцы, используемые на предприятии, выполняет записи в файл, имеющий расширение .sok. В файле данные о параметрах в течении полета представлены в формате текста, что значительно замедляет их анализ.

Задачей эксплуатационной практики была разработка программного решения, которое необходимо для визуализации и упрощения просмотра файлов формата .sok.

Фреймворк Qt

Qt – это библиотека классов и набор инструментального программного обеспечения для создания кросс-платформенных приложений с графическим интерфейсом (GUI). ПО переназначено для разработки на на C, C++, JavaScript и QML. В рамках выполнения задачи эксплуатационной практики разработка приложения выполнялась с использованием библиотеки PyQt. Эта библиотека представляет из себя набор расширений графического фреймворка Qt на языке Python.

Реализация пользовательского интерфейса при помощи Qt Designer

Qt Designer – это инструмент для проектирования и создания графических пользовательских интерфейсов из компонентов Qt. Позволяет создавать и настраивать различные элементы интерфейса в режиме “что видишь, то и получишь”. Данный инструмент позволяет добавлять в графический интерфейс различные компоненты (например, кнопки, вкладки, чекбоксы и т.п.), а также редактировать их внешние свойства.

При помощи Qt Designer был разработан .ui файл, который содержит в себе описание графического интерфейса и представляет из себя заготовку, используя которую будет формироваться остальное содержимое окна приложения. Файл открытый в Qt Designer представлен на рисунке 1.

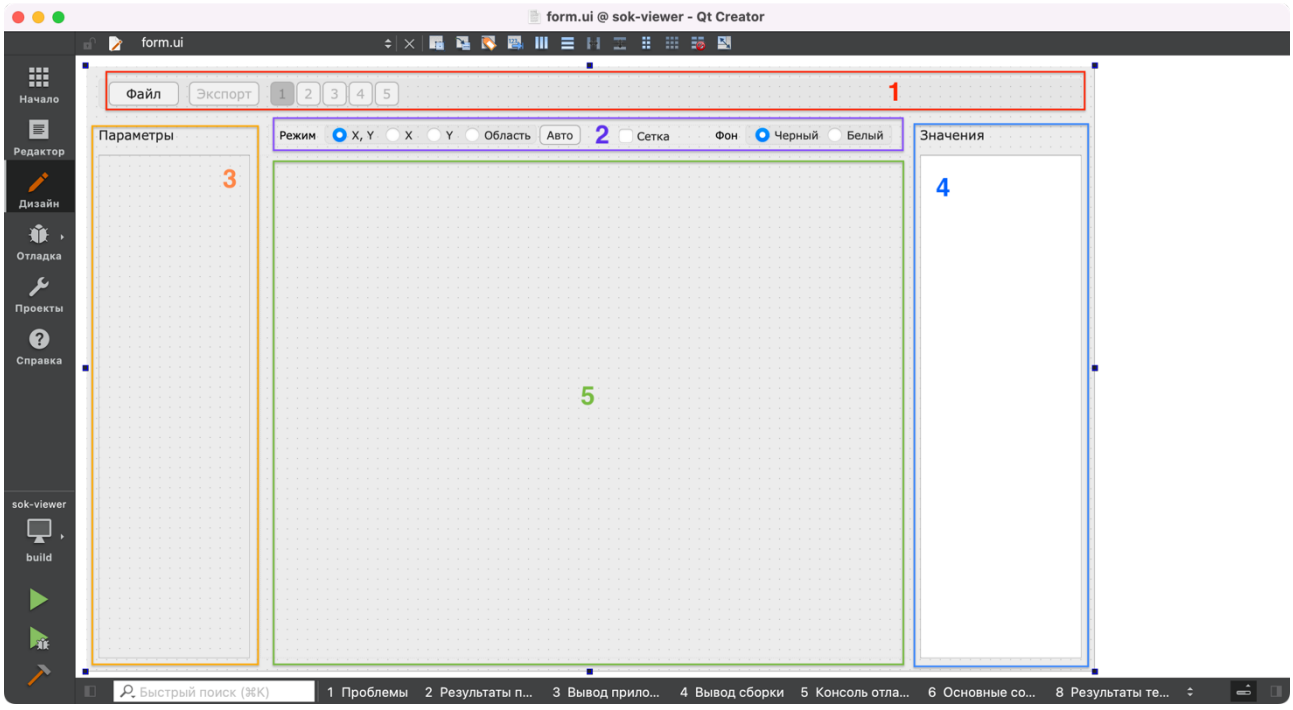

Рисунок 1 – Графический интерфейс программы, реализованный при помощи Qt Designer

На приведенном рисунке обозначены перечисленные ниже функциональные области.

1. Верхняя панель, которая содержит:
    - кнопку “файл”, выполняющую открытие и загрузку .sok файла;
    - кнопку “экспорт”, выполняющую сохранение отображаемых на выбранном слое графиков;
    - кнопки переключения между слоями.
2. Панель настроек работы с графиками, содержащую:
    - режим перемещения и масштабирования по осям: перемещение то осям X и Y, перемещение только по оси X, только по оси Y, или выполнение увеличения выбранного участка;
    - кнопка автомасштабирования, выполняющая установку области видимости с минимальными и максимальными значениями равными минимальному и максимальному значению по осям X и Y для выбранных параметров;

- чекбокс для включения/отключения сетки на координатной плоскости;
- выбор цвета координатной плоскости (черный или белый).

3. Список параметров с чекбоксами для включения/отключения отображения определенного параметра.
4. Список, содержащий в себе значения выбранных параметров для установленного на координатной плоскости указателя.
5. Основная рабочая область, на которой отображаются графики выбранных параметров.

При реализации основных элементов интерфейса использовались объекты следующих классов:

- QPushButton для кнопок;
- QGroupBox для объединения группы из нескольких связанных функциональностью кнопок (например, кнопок выбора цвета фона);
- QRadioButton для радиокнопок (элемент интерфейса, который позволяет пользователю выбрать одну опцию из предопределённого набора);
- QCheckBox для чекбоксов;
- QLabel для добавления текстовых подписей;
- QScrollArea для прокручиваемой области списка параметров;
- QListWidget для списка значений параметров в точке установленного указателя;
- QGridLayout для выравнивания всех помещенных в него элементов интерфейса по сетке (применение данного класса необходимо для сохранения расположения и масштабирования интерфейса при изменении размеров окна программы).

Также для большинства элементов интерфейса были созданы таблицы стилей, содержащие в себе описание внешнего вида каждого из элементов (толщина границ, скругления, цвета и т.д.).

Полученный файл имеет расширение .ui и в дальнейшем при сборке проекта будет автоматически преобразован Qt Creator в файл .py, содержащий в себе описание тех же самых элементов интерфейса, но уже на языке Python.

Реализация логики работы приложения.

Следующий после создания интерфейса этап разработки приложения – реализация логики его работы. Для этого в Qt используются слоты и сигналы.

Основное окно приложения представляет из себя объект класса Widget, содержащий внутри себя объект класса Ui_Widget, в котором, в свою очередь, описаны элементы интерфейса приложения. Виджеты имеют возможность посылать сигналы. Сигнал выполняется тогда, когда с элементом интерфейса происходит определенное событие (говоря иначе, сигналы срабатывают, когда пользователь взаимодействует с приложением). Слотом является функция, которая вызывается в ответ на определенный сигнал. Ниже представлена таблица соответствия сигналов и слотов для приложения.

Таблица 1 – Сигналы приложения и соответствующие им слоты

Отправитель	Сигнал	Слот (функция)
whiteRadioButton	toggled	backgroundChanged
blackRadioButton	toggled	backgroundChanged
radioButtonX	toggled	zoomModeChanged
radioButtonY	toggled	zoomModeChanged
radioButtonXY	toggled	zoomModeChanged
radioButtonArea	toggled	zoomModeChanged
layersWidget	tabBarClicked	layerChanged
gridCheckBox	stateChanged	gridClicked
dataButton	clicked	dataButtonClicked
exportButton	clicked	exportButtonClicked
autozoomButton	clicked	autozoomClicked

Основной виджет помимо объекта класса Ui_Widget содержит в себе следующие структуры данных:

- data – DataFrame, в котором хранятся считанные данные из файла;
- selectedParameters – список, в котором хранятся параметры, отображаемые на графике;
- time – список со значениями времени;
- fileAlreadyOpened – булевая переменная, показывающая был ли ранее открыт файл;
- plotPointerX, plotPointerY – строки, в которых хранятся координатами установленного указателя;
- plotBackgroundColor, plotZoomMode – строки, содержащие в себе значения параметров графика: цвет фона, режим работы (X, XY, область);
- plotGrid – булевая переменная, показывающая включена ли сетка координатной плоскости;
- legendScale – масштаб подписей для обозначений цветов графиков на экспортируемом изображении;
- layersNumber – количество слоев;
- currentLayer – текущий слой;
- layersParameters – список с параметрами всех имеющихся слоев.

Реализация функций-обработчиков сигналов.

Для обработки сигналов, посылаемых приложением, был реализован ряд функций, а также ряд вспомогательных функций, которые вызываются не напрямую сигналами, а внутри функций-обработчиков. Ниже описан принцип работы всех реализованных функций.

dataButtonClicked

Функция вызывается, когда пользователь нажимает на кнопку “файл” и предназначена для открытия .sok файла. Выполняет следующую последовательность действий:

- вызывает диалоговое окно выбора файла и получает имя выбранного файла;
- считываем значения из файла, удаляя строки с повторяющимися значениями в столбце со значениями времени;
- заполняем структуры данных, хранящиеся внутри виджета основного окна;
- если файл открыт впервые после запуска программы, помещаем в область 5, показанную на рисунке 1, объект класса PlotWidget из библиотеки pyqtgraph (это виджет, который предназначен для отображения графиков и взаимодействия с координатной осью);
- по названиям параметров, которые были считаны из файла, создаем список чекбоксов в окне "Параметры";
- если какое-то из описанных выше действий завершилось неудачно (обычно это происходит, когда файл имеет неправильный формат), выполняется очищение структур данных, хранящихся внутри виджета основного окна, а также отображение сообщения об ошибке чтения.

Также стоит отметить, что по нажатию каждого из чекбоксов в списке параметров вызывается функция checkBoxStateChanged, которая в свою очередь вызывает функции addParameter или deleteParameter:

- функция addParameter выполняется при включении чекбокса и выполняет вызов диалогового окна выбора цвета графика для параметра, а после добавляет его на координатную плоскость;
- функция deleteParameter выполняется при выключении чекбокса и выполняет удаление графика параметра с координатной плоскости.

Окно программы после удачной загрузки файла с включенным отображением одного из параметров представлено на рисунке 2.

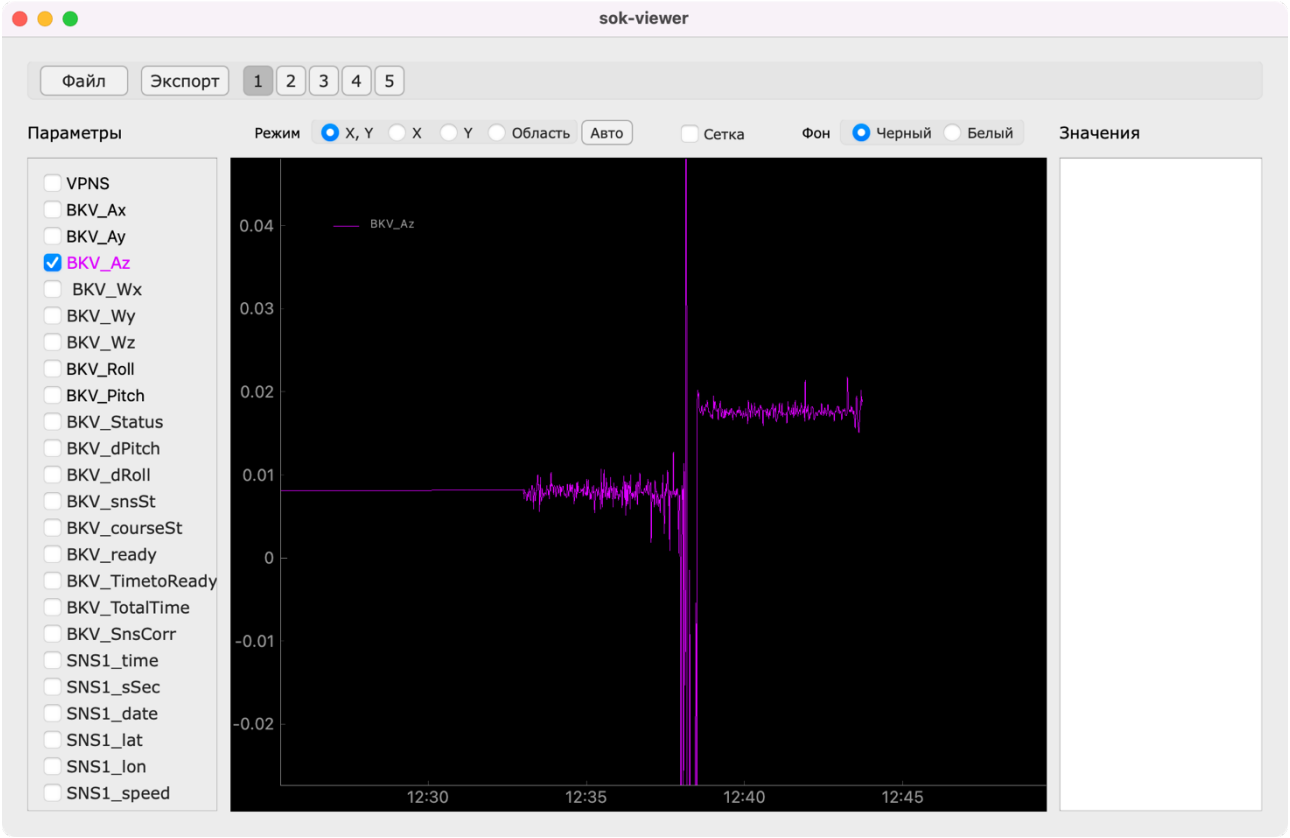

Рисунок 2 – Окно программы после удачной загрузки файла

backgroundChanged, zoomModeChanged, autozoomClicked, gridClicked

Данные функции вызываются при взаимодействии с элементами интерфейса настройки параметров работы с графиками.

- Функция backgroundChanged обрабатывает событие смены цвета графика, и в зависимости от выбранного значения устанавливает параметр plotBackgroundColor (“Black” или “White”);
- Функция gridClicked вызывается при нажатии на чекбокс сетки и выполняет включение/отключение отображения сетки на координатной плоскости.
- Функция zoomModeChanged обрабатывает событие смены режима зума и перемещения по графику, и в зависимости от выбранного значения устанавливает параметр plotZoomMode. Режимы “X”, “Y”, “XY”, позволяют выполнять масштабирования и перемещения при помощи мыши только по соответствующий названию осям, а режим “Area”

позволяет выполнять масштабирование по обоим осям и приближать выбранный при помощи прямоугольного выделения участок.

- Функция autozoomClicked вызывается при нажатии на кнопку автомасштабирования и устанавливает область видимости координатной плоскости по минимальным и максимальным значениям отображаемых графиков.

plotClicked

Функция вызывается при нажатии на график и выполняет установку указателя в выбранной точке, а также отображение в окне “Значения” точных значений включенных параметров для соответствующего точке значения времени. Выполняет следующую последовательность действий:

- получает координаты точки, в которой было выполнено нажатие мыши;
- отображает в выбранной точке указатель, а также отображает рядом с ним соответствующее выбранной точке значение времени;
- при помощи функции createValuesList формирует список включенных параметров и соответствующие им в точке указателя значения;
- при нажатии правой кнопки мыши на координатной плоскости удаляет указатель и очищает значения в окне “Значения”.

Окно программы после установки указателя, а также с измененными параметрами отображения графика представлено на рисунке 3.

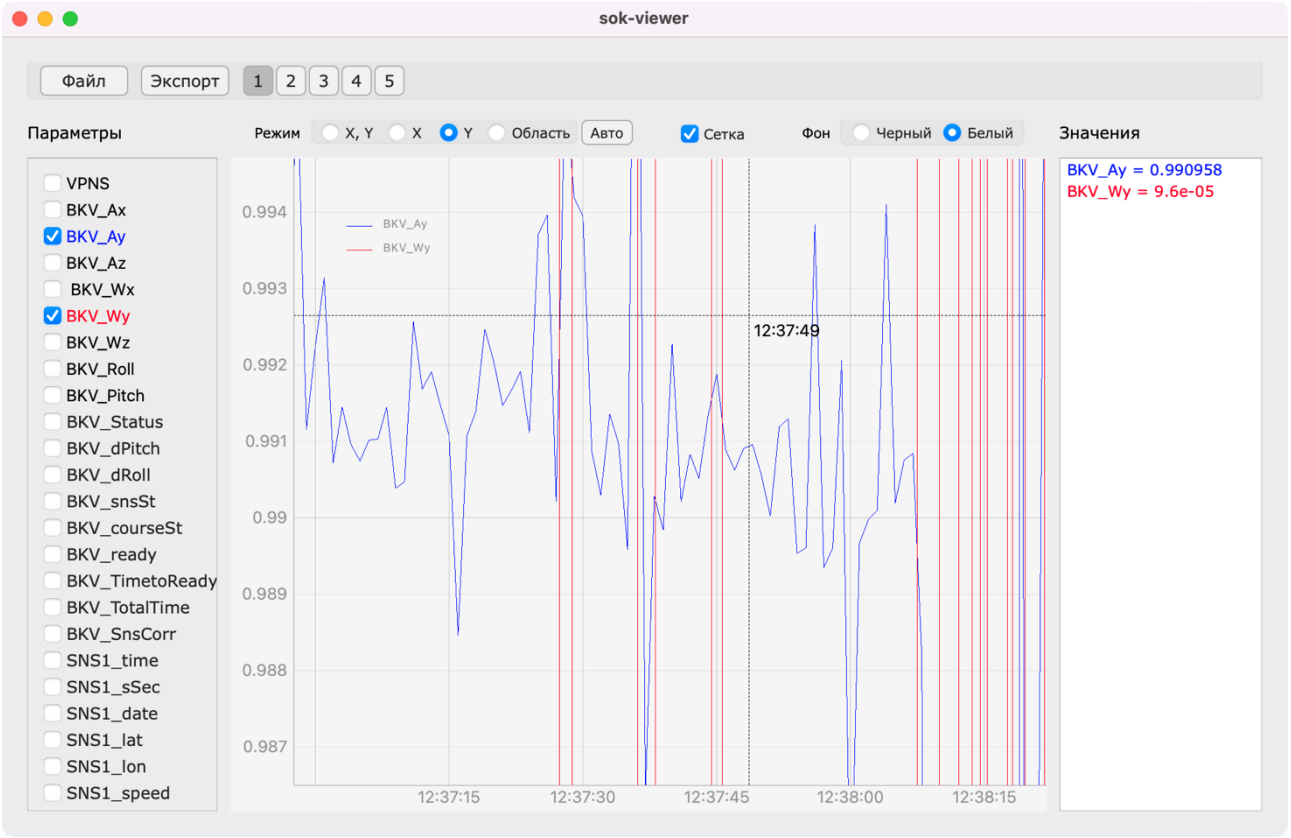

Рисунок 3 – Окно программы после установки указателя

layerChanged

Функция выполняет переключение текущего слоя. Каждому слою соответствует свой набор активных параметров, а также своя область видимости и установленный указатель на координатной плоскости. Для работы со слоями используются следующие структуры данных внутри класса виджета основного окна:

- selectedParameters – список активных параметров текущего слоя;
- plotPointerX и plotPointerY – координаты указателя текущего слоя;
- currentLayer – номер текущего слоя;
- layersParameters – список словарей для всех слоев, в каждом из которых хранится: номер слоя, выбранные параметры, список объектов QCheckBox, координаты X и Y для указателя.

При переключении слоя функция выполняет следующие действия:

- получает номер выбранного слоя;
- копирует все данные о текущем слое в список layersParameters;

- получает из списка layersParameters все данные, соответствующие новому слою, и записывает их в переменные, хранящие данные о текущем слое (selectedParameters, plotPointerX, plotPointerY, currentLayer);
- формирует соответствующий текущему слою список чекбоксов в окне “параметры”.

exportButtonClicked

Функция вызывается при нажатии кнопки экспорта и выполняет экспорт в .png файл координатной плоскости для текущего слоя. Выполняет следующую последовательность действий:

- вызывает диалоговое окно сохранения файла;
- получает выбранное имя файла, а также путь для его сохранения;
- выполняет экспорт графиков в файл формата .png.

Пример полученного в результате экспорта изображения представлен на рисунке 4.

Рисунок 4 – Экспортированное изображение

drawPlot

Функция выполняет перерисовку графика и вызывается каждый раз, когда с графиком происходят какие-либо изменения (добавляется новый параметр или удаляется уже активный, меняется цвет или сетка графика, устанавливается или убирается указатель).

Получение исполняемого файла приложения.

Финальный этап разработки приложения – представление его в форме исполняемого файла Windows. Исполняемый файл – это файл, который может быть установлен или запущен на компьютере без использования дополнительного программного обеспечения или библиотек. Он имеет расширение .exe для ОС Windows. Конвертируя свой скрипт на языке Python в исполняемый файл, можно защитить свой код от изменения или кражи, а также облегчить другим людям использование программы без установки интерпретатора и дополнительных библиотек.

На предприятии установлены компьютеры с операционной системой Windows 7 разрядности 32 и 64 бита, а также Windows 10 разрядностью 64 бита, поэтому при получении исполняемого файла необходимо ориентироваться на его работоспособность на системах с указанными параметрами.

Приоритетной задачей является поддержка исполняемого файла самой старой ОС с самой младшей разрядностью – Windows 7 разрядностью 32 бита, потому что Windows 10 поддерживает запуск приложений из предыдущих версий ОС, а также 64-битные версии ОС поддерживают приложения для 32-битных, но не наоборот.

Для сборки исполняемого файла была использована библиотека PyInstaller. PyInstaller – это библиотека Python, которая может анализировать код и компоновать его с необходимыми модулями и библиотеками в один исполняемый файл. Она поддерживает множество платформ, включая Windows, Linux и Mac OS X. PyInstaller также может обрабатывать сложные

случаи, такие как импорт файлов данных, скрытый импорт, приложения с графическим интерфейсом и т.д.

Для сборки приложения в исполняемый файл была использована команда: pyinstaller --noconsole --onefile hello.py. Ключ noconsole отключает консоль при запуске исполняемого файла, а ключ onefile выполняет сборку приложения в единый .exe файл, без дополнительных программных файлов.

Также стоит отметить, что внутри исполняемого файла находится копия интерпретатора Python, который был установлен на компьютере, на котором выполнялась сборка. Поэтому необходимо было использовать интерпретатор, поддерживаемый 32-битной Windows 7 – 32-битную версию интерпретатора Python версии 3.8.6.

Заключение

В процессе прохождения практики было получен опыт разработки графического приложения для предприятия АО “РПКБ”.

За время практики было выполнено ознакомление с фреймворком Qt, а также получен опыт работы с ним. Было реализовано графическое приложение для визуализации файлов формата .sok. Разработанное приложение было представлено в виде исполняемого файла Windows.

Использованные источники

1. Qt for Python // Qt Documentation | Home URL: https://doc.qt.io/qtforpython-6/ (дата обращения: 05.07.2023).
2. PyQtGraph - Scientific Graphics and GUI Library for Python // PyQtGraph - Scientific Graphics and GUI Library for Python URL: https://www.pyqtgraph.org/ (дата обращения: 10.07.2023).
3. Using PyInstaller — PyInstaller 6.1.0 documentation // PyInstaller Manual — PyInstaller 6.1.0 documentation URL: https://pyinstaller.org/en/stable/usage.html (дата обращения: 22.07.2023).