|  |
| --- |
| Squidstat GUI  Programmers guide |
| 2017 |

Table of contents

[1 Introduction 3](#_Toc497860951)

[2 General description of the structure 4](#_Toc497860952)

[3 Serial communicator 5](#_Toc497860953)

[4 Instrument operator 6](#_Toc497860954)

[5 Instrument enumerator 7](#_Toc497860955)

[6 What is an experiment 8](#_Toc497860956)

[7 Workflow of a regular experiment 9](#_Toc497860957)

[8 What is a custom experiment 10](#_Toc497860958)

[9 Workflow of a custom experiment 11](#_Toc497860959)

[10 What is a manual experiment 12](#_Toc497860960)

[11 Workflow of a manual experiment 13](#_Toc497860961)

[12 Experiment plugin creation 14](#_Toc497860962)

[13 Builder element plugin creation 15](#_Toc497860963)

[14 How does the "Run an Experiment" tab work – [optional] 16](#_Toc497860964)

[15 How does the "Build an Experiment" tab work – [optional] 17](#_Toc497860965)

[16 How does the "Manual Control" tab work – [optional] 18](#_Toc497860966)

[17 How does the "View Data" tab work – [optional] 19](#_Toc497860967)

[18 How does the "Channel Status" tab work – [optional] 20](#_Toc497860968)

[19 How does the notification area work – [optional] 21](#_Toc497860969)

[20 How does the firmware updater work – [optional] 22](#_Toc497860970)

[21 How to extend the firmware updater – [optional] 23](#_Toc497860971)

[22 QSS tips and hints – [optional] 24](#_Toc497860972)

[1 Estimation summary: 25](#_Toc497860973)

# Introduction

This document contains a whole description of the SquidStat GUI software.

Developers can use this guide to extend the software functionallity.

# General description of the structure

The basis of the application is the "MainWindow" class (see fig. 1). It is the Controller of the application. Its duty is to create and coordinate all other classes.

First, the "MainWindow" creates the "MainWindowUI". The "MainWindowUI" is the View of the application. It handles all user activity and operates all UI components.

Next, the "MainWindow" loads prebuild experiment plugins (see p. 12), builder element plugins (see p. 13) and custom experiments (see p. 8).

Next, the "MainWindow" creates the "InstrumentEnumerator" (see p. 5). It continuously checks the set of connected instruments and changes in this set. All changes are reported to the "MainWindow".

Finally, the "MainWindow" creates connections (in terms of Qt) to handle corresponding happenings.

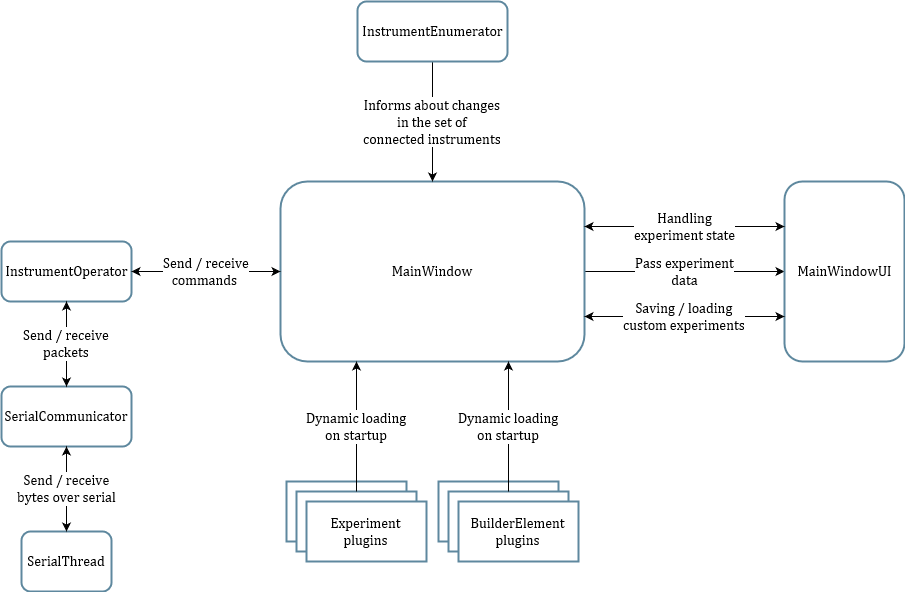


Figure 1 – General scheme of the application

Generally, the startup looks as the following:

1. Load fonts from the resources.
2. Create all UI elements.
3. Load prebuild experiment plugins.
4. Load builder element plugins.
5. Load custom experiments.
6. Create the "InstrumentEnumerator".
7. Apply stylesheets (loaded from the resources) to the whole application.

For every connected instrument the "MainWindow" has the instance of the "InstrumentOperator" (see p. 4) that provides API for sending and receiving all supported commands and responses.

The application is mostly single-threaded. Separated threads are used for raw data exchange over serial and for the instrument enumeration. So, the total amount of threads in application is equal to .

# Serial communicator

For the PC every instrument is the COM-port. Every instrument supports the specific protocol[[1]](#footnote-1) over serial interface.

To handle the communication over the serial the application use two classes: the "SerialThread" and the "SerialCommunicator".

The "SerialThread" is inherited from the "QThread". It owns the corresponding "QSerial" and perform read/write operations over the serial. All Qt-connections outside "SerialThread" have to be of the "Qt::QueuedConnection" type[[2]](#footnote-2). So, the "SerialThread" reads all data from the serial even on high data rates and queue the data for the further handling to the main thread.

The "SerialCommunicator" is inherited from the "QObject" and owns the "SerialThread". The "SerialCommunicator" handles the instrument protocol over the raw serial data. It allows to send commands and emits signal "SerialCommunicator::ResponseReceived" on every valid packet in the data flow that was recognized.

# Instrument operator

The "InstrumentOperator" class is the abstraction from the instrument protocol for the application. It has API that duplicates every command and every response to guarantee the correctness of the input parameter set and parameter types.

The "InstrumentOperator" owns the "SerialCommunicator".

# Instrument enumerator

The "InstrumentEnumerator" is the thread that continuously checks the set of the connected instruments and emits signal on every change (arrival or removal). It is inherited from the "QThread".

Every second (hardcoded value) the "InstrumentEnumerator" performs the following:

1. Get the list of COM-ports (available ports).
2. Check if "already connected ports" are still among "available ports".
3. Every "already connected port" that is not in the "available ports" list move to the "instruments to delete" list.
4. Emit the "InstrumentEnumerator::RemoveDisconnectedInstruments" signal for every port from the "instruments to delete" list.
5. For every port from the "available ports" list and not in the "already connected ports" list try to request the handshake via instrument protocol. If success – add this port into the "instruments to add" list.
6. Emit the "InstrumentEnumerator::AddNewInstruments" signal with the "instruments to add" list as a parameter.

Ports are considered to be the same if they have the same name and serial number.

# What is an experiment

0.5 h

# Workflow of a regular experiment

Description of the whole call-chain between "Start Experiment" button pressing and data plotting.

1.5 h

# What is a custom experiment

0.5 h

# Workflow of a custom experiment

0.5 h

# What is a manual experiment

0.5 h

# Workflow of a manual experiment

0.5 h

# Experiment plugin creation

Updating existing document.

0.5 h

# Builder element plugin creation

Describing the process of the builder element plugin creation.

1.5 h

# How does the "Run an Experiment" tab work – [optional]

Widget structure, widget interaction logic.

1.5 h

# How does the "Build an Experiment" tab work – [optional]

Widget structure, widget interaction logic.

1.5 h

# How does the "Manual Control" tab work – [optional]

Widget structure, widget interaction logic.

1 h

# How does the "View Data" tab work – [optional]

Widget structure, widget interaction logic.

2 h

# How does the "Channel Status" tab work – [optional]

Widget structure, widget interaction logic.

0.5 h

# How does the notification area work – [optional]

Widget structure, widget interaction logic.

0.5 h

# How does the firmware updater work – [optional]

Widget structure, widget interaction logic, protocol description.

1 h

# How to extend the firmware updater – [optional]

For example, what to do to add teensy utility.

0.5 h

# QSS tips and hints – [optional]

Some undocumented and unobvious tricks for Qt Style Sheets.

0.5 h

# Estimation summary:

Minimum estimation – 10.5 hours.

Optional paragraphs – 9 hours.

Total (minimum + optional) – 19.5 hours.

1. This document does not contain the description of the instrument protocol [↑](#footnote-ref-1)
2. Connection type is the fifth parameter of the "QObject::connect" method [↑](#footnote-ref-2)