**Report on plotter based on Qt**

**1. OVERVIEW**

The plotter is based on C++ for relatively faster execution while keeping in consideration practical aspects like community support and available open-source libraries. It uses Qt, a powerful C++ SDK that provides support for complicated GUI (and corresponding interactions) and has a wide community. For plotting, specifically, the QCustomPlot library was used.

**1.1 SCOPE AND FEATURES**

* The plotter supports plotting off a live datastream from a serial port, or a text file.
* 2 plotting modes are supported, with 1 plot per graph or 3 plots per graph.
* Multiple windows are automatically created as per the number of detected streams of data in the input source.
* When using serial port as an input, files with all recieved data are created and stored automatically, and can be directly plotted from later.
* Multiple files can be read from and plotted in one session, with the option of plotting live serial data parrallelly.
* Scrolling through graph is supported.
* Dynamically changing the relative scaling of Y-axis is supported, with space allocated(removed) to(from) the top or bottom of the graph independently.
* Dynamically changing the maximum number of data points visible in a graph at any time is supported.
* Dynamically setting the visibilty of a plot in the graph is supported.

**2. USAGE**

**2.1 INPUT FORMAT**

The format used to input data is used to determine the number of plots per graph.

Input through both serial and files must have one of the following formats:

* [4 lowercase letter graph name]:X[X-value]Y[Y-value]Z[Z-value]T[Tvalue]

For a graph with 3 plots, X, Y, and Z vs T

Example:

accl:X1Y2Z3T1

gyro:X4Y5Z6T1

* [4 lowercase letter graph name]:W[W-value]T[Tvalue]

For a graph with 1 plot, W vs T

Example:

baro:W1T1

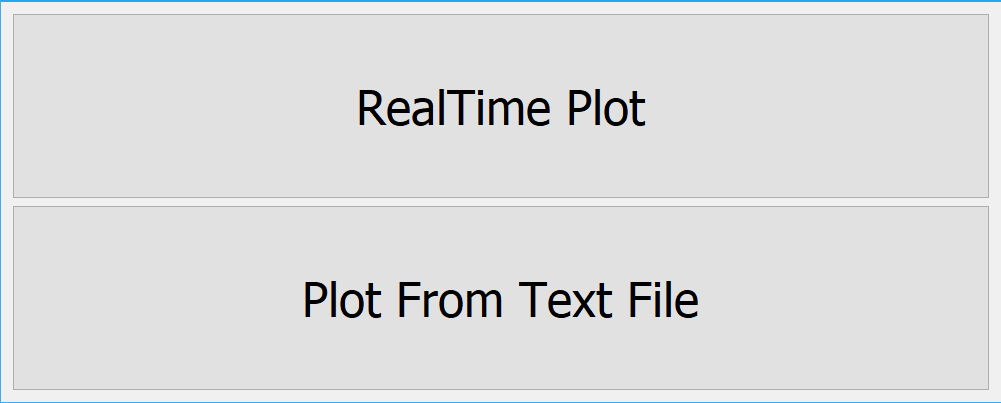
sine:W0T1

In both cases, each line of input is expected to end in a newline (\n) character.

The value of T is expected to indicate time elapsed, and is thus the key axis, the other axes being value axes. It must hence have a steadily increasing value for every subsequent input in the same plot.

**2.2 UI**

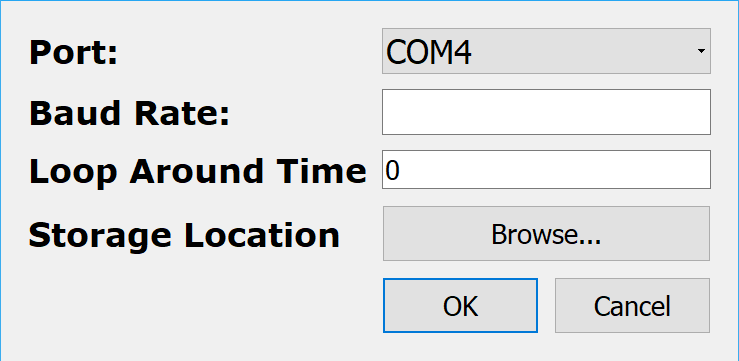
**2.2.1 Main window**

****

**Realtime plot:** Leads to Serial setup window. Used to plot via serial.

**Plot from text file:** Leads to file setup window. Used to plot off of files

**2.2.2 Serial setup**

****

**Port:** Used to select source for serial input. Available ports are populated into the ComboBox automatically.

**Baud Rate:** Enter the baud rate of the serial source here.

**Loop Around Time:** Enter max ‘T’ value of input after which subsequent points start from 0 time.

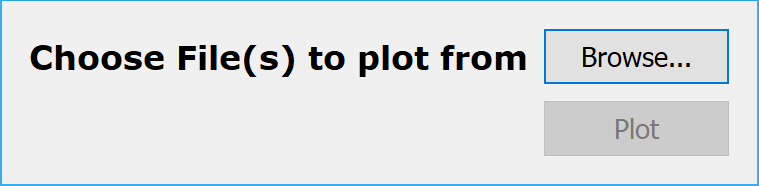
**Storage location:** Opens a dialog box used to select a folder. Inside, the selected folder, a new folder is created and named as the value of miliseconds elapsed since Epoch. Inside this folder files with recieved data are stored.

**Press OK to proceed to start reading the serial port and plotting Data. Also closes Serial Setup window.**

**Pressing OK without all fields configured will lead to messageBox stating “Invalid serial config”**

**Press Cancel to close Serial Setup window without starting to read serial port.**

**2.2.3 File Setup**

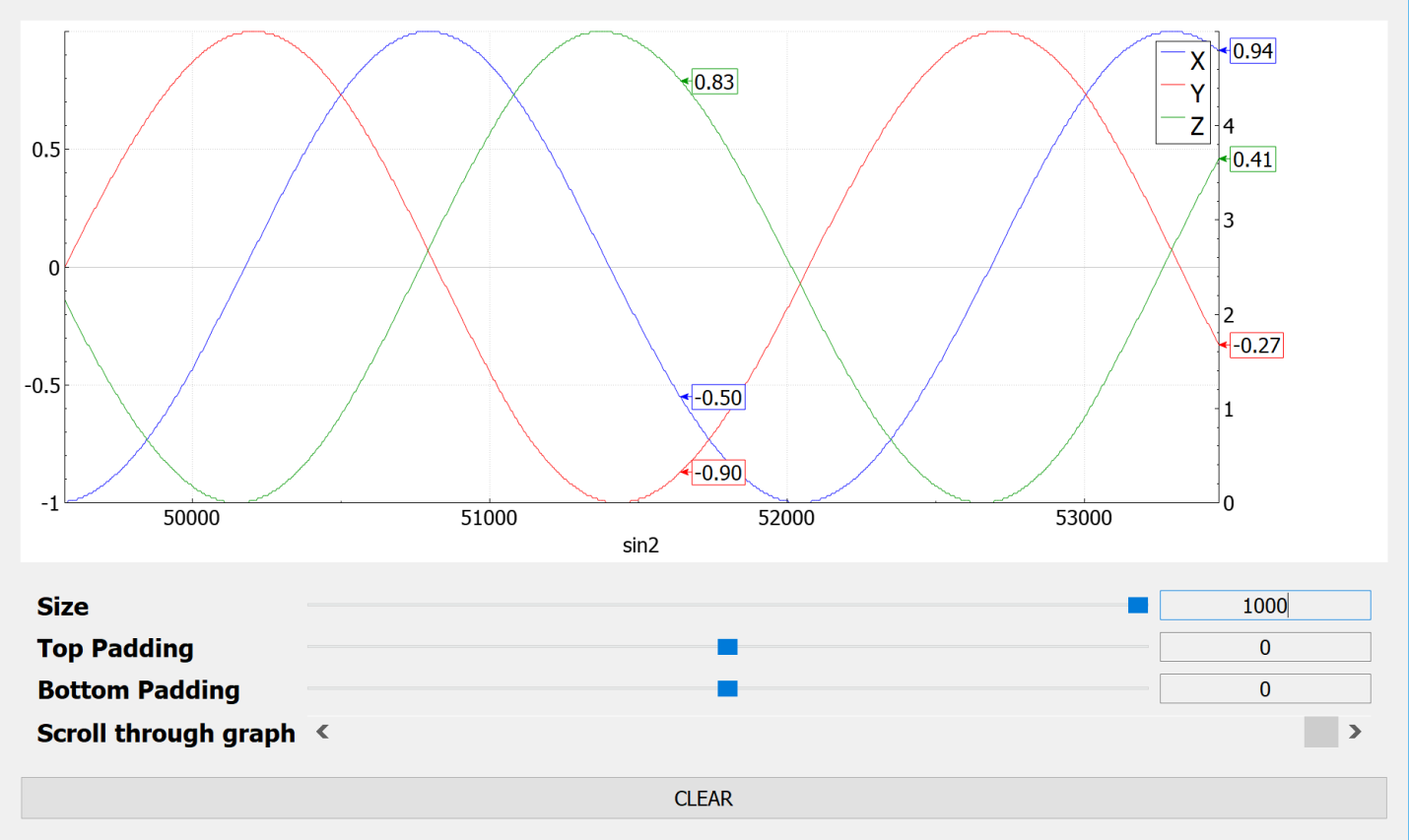
****

**Choose file(s) to plot from:** Opens a dialog box used to select multiple files. Plot button becomes active if files are actually selected.

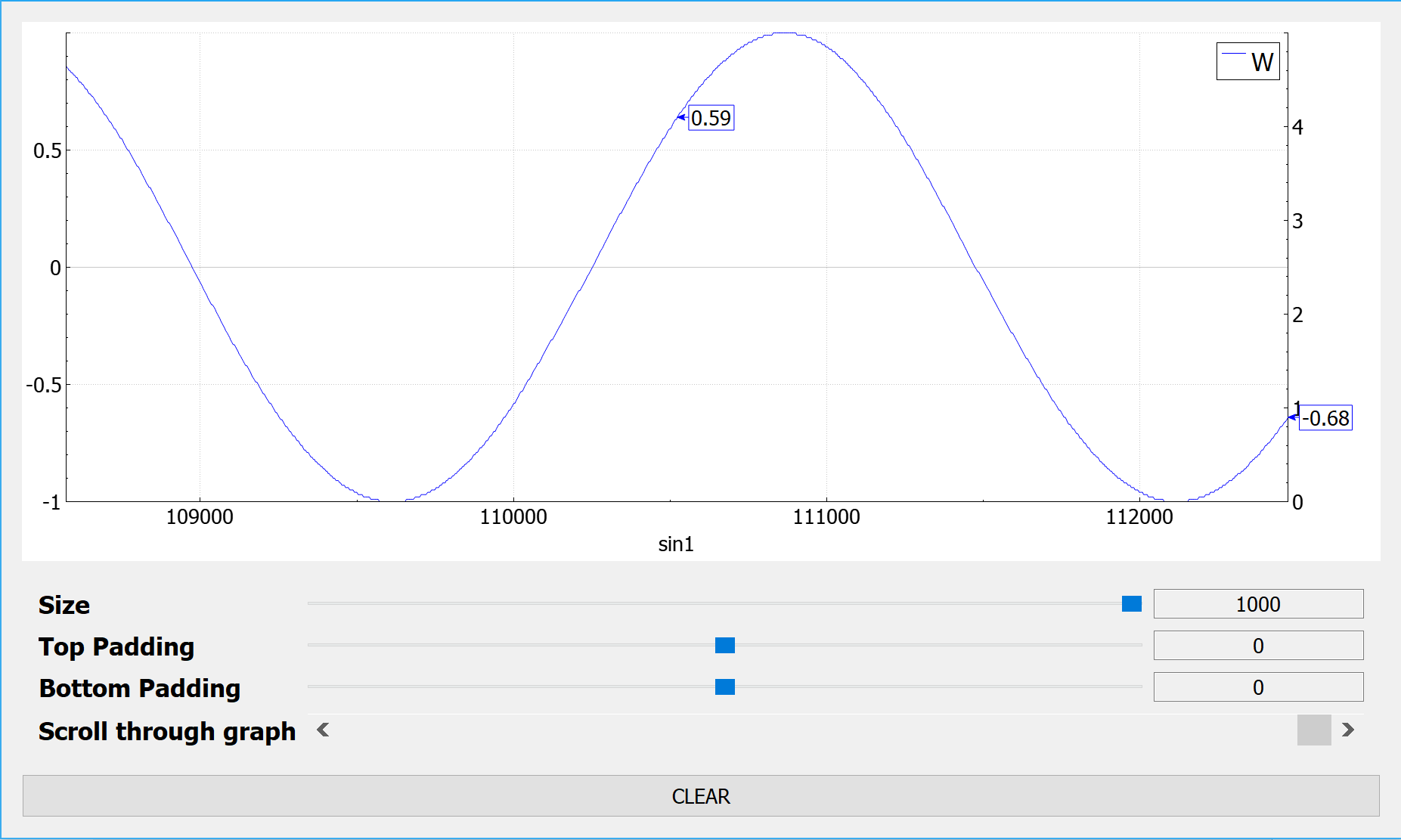
**Press PLOT to read the files and populate the graph with corresponding data. Also closes File Setup window.**

**2.2.4 Graph**

**For 3 plots /graph**

****

**For 1 plot /graph**

****

**UI ELEMENTS**

**Size:** Number of points on the graph at a time. Slider has a range of 50 to 1000, while the Textbox can be set to a custom value, Higher, lower or within the range of the slider.

**Top Padding:** Used to set padding above the graph. Slider has a range of -1000 to 1000, while the Textbox can be set to a custom value, Higher, lower or within the range of the slider, Including floating point values.

**Bottom Padding:** Used to set padding below the graph. Slider has a range of -1000 to 1000, while the Textbox can be set to a custom value, Higher, lower or within the range of the slider, Including floating point values.

**Scroll through graph:** Used to scroll through the graph if the entirety of it cannot be displayed at once. If plotting from serial, setting this to maximum will result in plotting the latest set of values, while by decreasing it enables viewing previous values from the current plotting session.

**Clear:** Used to clear all data in the graph’s data structures. Only new data will be visible in the graph, but the file saving the data will retain previous data.

**INTERACTIVE ELEMENTS**

**Histogram:** Clicking on a specific axis on the graph’s histogram toglles the axis’ visibility.

**The graph:**  Hovering the mouse over any part of the plot will display the Y-axis values for the corresponding X-axis value of the point over which the mouse is hovering.

**PASSIVE FEATURES**

**Graph name** can be seen just below the X-axis, centered W.R.T. the screen.

**Auto scaling** of the axes is enabled and takes into account the vanilla data range and the padding defined using the UI elements.

**3. Class Documentation**

This section focuses on abstractly explaining used in the application. Please refer to the source code for in-depth explanation of the code through comments.

**3.1 CLASS OVERVIEWS**

**Main**

Entry pint into the program. Sole purpose is to instantiate the MainWindow class and pass flow control to it.

**MainWindow**

The base class in Qt that executes and sets up the initial UI main menu. Further used to create new windows and pass data between input sources and the string processing class. Creates and reads data from file or creates serial objects for serial input depending on UI input.

**SerialInitial**

Creates a UI that contains fields to specify parameters to open a serial connection to read data from. In the event of a successful submit operation, it passes this data back to MainWindow where a serial object is created and read from.

**FileInitial**

Creates a UI that contains fields to specify parameters to initialize potentially multiple files. . In the event of a successful submit operation, it passes this data back to MainWindow where file object(s) is(are) created and read from.

**Serial**

Creates a QSerialPort using specified parameters and provides a slot to read all data from the port and put it into an instance of DataStorage class, which only allows synchronized access to itself; Also, provides a signal that is emitted when a successful write to the DaraStorage instance is made.

**StringParser**

Detects the type of graph to plot depending on the standard string format used in the input. Contains checks and validations to detect corrupt input. After checks are completed it extracts data points from the string and sorts them into the data structure corresponding to their graph. If a data point is detected for a graph that does not exist, it requests MainWindow to create a new graph and give it the reference. At the end of each cycle, transfers data points to Objects of the ParsedDataStorage class and emits a slot telling the graphs that data has been updated. Each graph has its own object of parsedDataStorage and points are only sent to the correct instance depending on which graph they belong to. Also, all of the recieved data is dumped into a file that can be directly read later to get the exact same graph.

**DataStorage**

Stores raw string data read , before it is read. Data is removed after read. Provides read and write functions for synchronized access that return a boolean that indicates the success of read or write operation. Also contains URL to file data is being saved to.

**ParseDataStorage**

Contains Vectors that contain values for data points that have been parsed and must be passed to a graph. Provides read and write functions that provide sequential access and return a boolean that indicates the success of read or write operation. Data is removed from the object after being read.

**PlotAndDump**

Creates graphs from passed data points, contains UI elements that implement features on the graph.

**AxisTag**

Objects act as tags to label data points on a graph. They use graph coordinates to position themselves and to control their readouts.