Development of Microgrid Monitoring Interface

Ozan Aktürk, *EEE STAR, METU*

Abstract
This report explains the methods I used to develop a prototype microgrid monitoring interface on Python. It contains the pre-design phase, the libraries used, and the step-by-step explanation of the code.

***Index Terms*— application, button, checkbox, combobox, csv, data, datetime, export, interface, matplotlib, microgrid, monitoring, prototype, pyqt5, python, radiobutton**

I. Introduction

IN this report, I will describe a prototype interface for displaying the electrical information of a microgrid.

II. Pre-design Phase

A. Problem Description

This project aimed to design an interface that reads the necessary attributes of a microgrid from a file, and then shows the user via plotting or a table. It is intended to be interactive as it takes commands from the user.

It has to take the inputs regarding the starting and ending time of the power attributes the user wants to see. Then, it has to read and find the required part of the data from the input file. According to the user's choice, it needs to show that data via a plot or a table. Also, it has to be able to export the data into another file.

B. Before the Implementation Stage

I contacted Research Assistant Soheil Pouraltafi-Kheljan during the pre-design process frequently. He told me what we wanted our application to do. Then, I searched for tools or modules to utilize for the project in the design process. After my research, I decided on the "PyQt-5 module" for Python as it is a common and useful tool. Also, there are many documents about PyQt-5 in case of any problem -which it did very frequently- I would be able to find solutions quickly.

Next, I started to learn about the module and the functions it contains. From the easiest to more complicated ones, I developed many applications to get used to the module.

Along with PyQt-5, I learned "matplotlib", "csv", and "datetime" libraries to utilize for the project. I used the matplotlib library for plotting, csv library for reading from or writing to a ".csv" file, and datetime library for selecting a date. Other than that, I learned the concept of the "MplCanvas" class in Python to show figures -plots in our case- in a PyQt-5 application, which cost the most time.

III. Implementation Phase

A. Starting Ideas

When the program is executed, I wanted it to open a window containing the required tools such as plotting area, table, etc. Later, considering Soheil's request, I created two additional empty windows on the same interface as an example for switching between windows.

Before the implementation, I needed to know the format of the input. Soheil sent me an input example, which can be found on my GitHub page written at the bottom left corner of this page. I used the format used in that file throughout the project.

I needed to ask the user which period of time they wanted the program to show. I decided to create two "QDateTimeEdit" widgets on the window to get the starting and ending time inputs.

Then, I needed to read the information in that time interval from the input file. I thought I could use the csv library for reading & exporting purposes.

Next, I needed to ask the user whether they want the program to plot the data, show it on a table, or export the data to another .csv file. I thought that I could use simple buttons and checkboxes for that.

Since the program is a prototype, I also wanted the user to choose the legend placement as an example for "combobox".

Having these ideas, I started to implement them one by one using classes in Python.

B. Input Format

This program takes input from a text (.txt) file. The format

This report is planned to be submitted to Turnitin on November 14, 2020. The report and the file containing the codes are accessible on https://github.com/ozanakturk/METUEEESTAR/find/main at least until January 2021.

Ozan Aktürk is an undergraduate student at Electrical & Electronics Engineering Department, Middle East Technical University, Ankara, Turkey (e-mail: e2303923@.metu.edu.tr).

of the input the program takes is as follows:

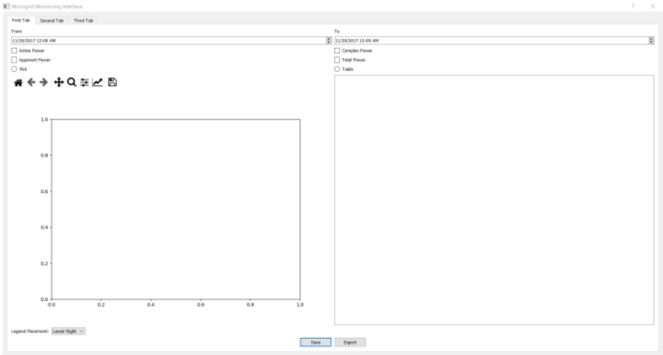
“mm/dd/yyyy hh:mm:ss aa.101 n n n n”

In the first item, m stands for months, d stands for days, and y stands for years. The multiple uses of these letters represent the expected digits it must have. Similarly, in the second item, h stands for hours and can be single or double-digit. M stands for minutes and aa stand for "AM" or "PM". ".101" has no meaning while n stands for a number, each one representing a power type. For example:

“11/46/2017 4:04:14 PM.101 8800.000 5490.000 7830.000 22110.000”

C. *Test Description and Results*

The program opens a window containing three sub-windows. The first window has all the tools, while the others are empty content-wise. The first window looks like in Fig. 1.



From: 11/20/2017 12:00 AM

To: 11/20/2017 12:05 AM

☐ Active Power

☐ Apparent Power

○ Plot

☐ Complex Power

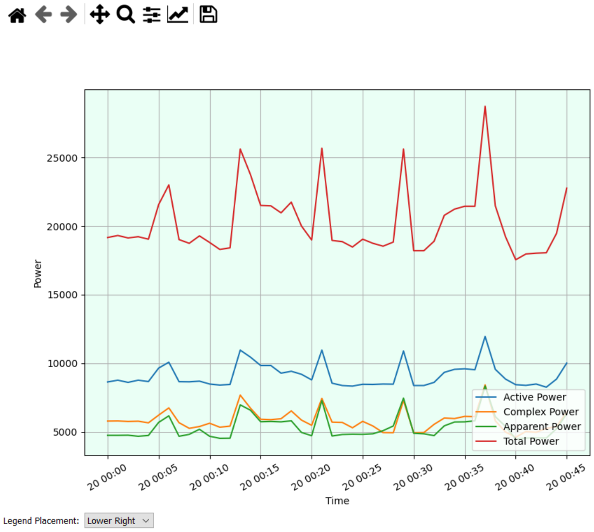
☐ Total Power

○ Table

Legend Placement: Lower Right

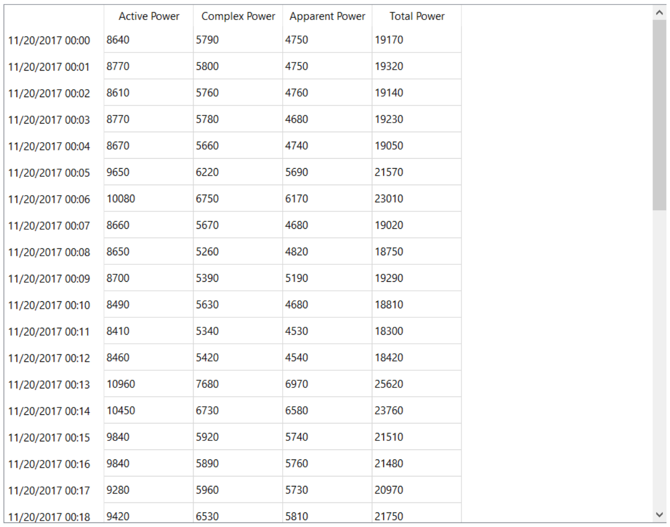
Fig. 1. The starting line-up of the program.

After selecting the dates and the power types, plotting and table widgets (not the whole window) look like Fig. 2 and Fig. 3, respectively. Note that the user should choose between plot or table "radiobutton" and then click on the "Save" button, one at a time. Showing both plot and table is also possible if chosen one at a time as usual.



Legend Placement: Lower Right

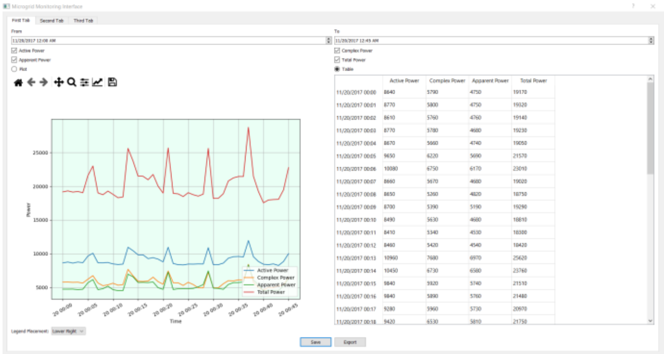
Fig. 2. The plotting widget’s normal operation.



	Active Power	Complex Power	Apparent Power	Total Power
11/20/2017 00:00	8640	5790	4750	19170
11/20/2017 00:01	8770	5800	4750	19320
11/20/2017 00:02	8610	5760	4760	19140
11/20/2017 00:03	8770	5780	4680	19230
11/20/2017 00:04	8670	5660	4740	19050
11/20/2017 00:05	9650	6220	5690	21570
11/20/2017 00:06	10080	6750	6170	23010
11/20/2017 00:07	8660	5670	4680	19020
11/20/2017 00:08	8650	5260	4820	18750
11/20/2017 00:09	8700	5390	5190	19290
11/20/2017 00:10	8490	5630	4680	18810
11/20/2017 00:11	8410	5340	4530	18300
11/20/2017 00:12	8460	5420	4540	18420
11/20/2017 00:13	10960	7680	6970	25620
11/20/2017 00:14	10450	6730	6580	23760
11/20/2017 00:15	9840	5920	5740	21510
11/20/2017 00:16	9840	5890	5760	21480
11/20/2017 00:17	9280	5960	5730	20970
11/20/2017 00:18	9420	6530	5810	21750

Fig. 3. The table widget’s normal operation.

The program looks like in Fig. 4 in normal operation.



From: 11/20/2017 12:00 AM

To: 11/20/2017 12:45 AM

☑ Active Power

☑ Apparent Power

○ Plot

☑ Complex Power

☑ Total Power

● Table

Legend Placement: Lower Right

	Active Power	Complex Power	Apparent Power	Total Power
11/20/2017 00:00	8640	5790	4750	19170
11/20/2017 00:01	8770	5800	4750	19320
11/20/2017 00:02	8610	5760	4760	19140
11/20/2017 00:03	8770	5780	4680	19230
11/20/2017 00:04	8670	5660	4740	19050
11/20/2017 00:05	9650	6220	5690	21570
11/20/2017 00:06	10080	6750	6170	23010
11/20/2017 00:07	8660	5670	4680	19020
11/20/2017 00:08	8650	5260	4820	18750
11/20/2017 00:09	8700	5390	5190	19290
11/20/2017 00:10	8490	5630	4680	18810
11/20/2017 00:11	8410	5340	4530	18300
11/20/2017 00:12	8460	5420	4540	18420
11/20/2017 00:13	10960	7680	6970	25620
11/20/2017 00:14	10450	6730	6580	23760
11/20/2017 00:15	9840	5920	5740	21510
11/20/2017 00:16	9840	5890	5760	21480
11/20/2017 00:17	9280	5960	5730	20970
11/20/2017 00:18	9420	6530	5810	21750

Fig. 4. The normal operation of the program (after plotting and creating a table).

Clicking on the "Export" button does not change the look of the program. Instead, it creates a .csv file at the folder containing the program's executable file. Then, it fills the .csv file with the data shown on the plot or table widget. The first line of the exported file is as follows:

“11/20/2017 00:00, Active Power: 8640, Complex Power: 5790, Apparent Power: 4750, Total Power: 19170”

D. *Implementation Issues*

I spent most of the time formatting the plotting widget. When I created a space for the plotting widget, I was not able to change it from empty. As a solution, I found a method suggesting creating a class called "MplCanvas", which I will not explain in this report.

Creating the columns and rows for the table also was not easy. I had to learn and test out a lot of methods of "QTableWidget".

E. *Lessons Learned*

When creating the plotting widget, I first opened space for it by creating an "MplCanvas" object, and then recreating it in the "plot" function. Next time, I may want to find another solution that does not create a plotting widget twice. Also, some searching algorithms might be optimized in case of need. Lastly, the exporting format might be shorter.

IV. Conclusion

Most of the time, the code writing process was both learning and implementing at the same time. The last version of the program can successfully do all the tasks it is designed to do initially. As a prototype, it includes the necessary tools for a monitoring application, but it does not connect to a database.

When needed, the connection of this program with other programs or databases can be constructed. Also, some other sub-windows can be added and designed.

References

[1] M. Fitzpatrick, "Matplotlib plots in PyQt5, embedding charts in your GUI applications," 16-Nov-2020. [Online]. Available: https://www.learnpyqt.com/tutorials/plotting-matplotlib/. [Accessed: 16-Nov-2020].

[2] Programiz, “Reading CSV files in Python,” *Programiz*, 2020. [Online]. Available: https://www.programiz.com/python-programming/reading-csv-files. [Accessed: 16-Nov-2020].

[3] “Python Datetime,” *Python Dates*, 2020. [Online]. Available: https://www.w3schools.com/python/python_datetime.asp. [Accessed: 16-Nov-2020].

[4] Q. Company, “Qt Documentation,” *QTableWidget - Qt for Python*, 2020. [Online]. Available: https://doc.qt.io/qtforpython/PySide2/QtWidgets/QTableWidget.html . [Accessed: 16-Nov-2020].