Developer notes

# Overview

This document will serve as a living description of the application and its various components. This document is not intended to be “polished” or outward facing. For anyone who wants to further extend this application, this will clarify what application decisions were made and why. For full disclosure, this project is as much a learning opportunity as it is a passion project. A lot of decisions were made out of ignorance. Some decisions were made just to get the thing done. A detailed list of outstanding issues and bugs will be noted on github.

# Application structure

MotorMeter is a desktop application written for Windows 10 with c++. The initial version of this software was cobbled together with excel macros and a complex python backend. The software used a third party, open-source analysis tool and all together it kind of worked. As time went on, it proved difficult to get working on other machines and windows updates broke some key features. I then decided to port all this code over to a c++ application to avoid the installation headaches and sensitivity to windows changes. Since my only c++ experience has been in embedded development, this also serves as a learning opportunity.

The MotorMeter application consists of a few key features. There is the data collection through a serial interface layer, the data analysis through the zfit engine, and the integration of both things with the UI. Each of these sections are described in more detail later, but I want to highlight the interaction now. The UI serves to give the user a simple way to configure their supported test device, as well as conduct and review their test results. The UI also binds in any impedance analysis to a single model to keep all the data together. While the data collection and impedance analysis are the meat of the functionality, the UI is the core of the application. The UI should allow someone with no background in electrical engineering a clean way to collect characteristic performance data to quantify a motor’s underlying behavior and predict certain performance metrics before needed any in depth mechanical testing.

## Typical use case

The typical flow is to open the application to collect some data. The device under test is hooked up to the measurement device and the desired test parameters and settings are selected. The test is then executed by the user. This starts the serial interaction from the application to the test device. The application sets all the necessary parameters and then executes the test. The application is sanitizing the results and saving them to the current open “file” location. Upon completion of the test, the raw results are presented to the user in the form of two frequency-based plots. One plot is the primary measurement made, while the other is the secondary measurement made. The user can run more tests or delete some test series. If desired, the user could also export the test data to a CSV or excel format for further manipulation. If the user collected data on impedance and signal phase, the data can be further analyzed with the Zfit engine.

The Zfit engine can be used to compare collected test data to an equivalent circuit model of the system and back out values for the RLC circuit components. The equivalent circuit model can be generated by the user as external c++ functions. A simulation of the modeled circuit is then run iteratively and compared to the numerical data collected. The results of the simulation and the numerical results are plotted together to show how well they agree. The values of the constituent RLC components used in the “simulation” are displayed to the user, and can be used to gain insight about the actuator that would otherwise be difficult or impossible to attain. In the cases of a good approximation of the actuator being hard to attain, a simulation tuning system can be used to weight certain sections of the test data more than others, and to smooth out some sections of data. Upon satisfactory component information is determined, the user can once again save the data and export it to a CSV or excel file type.