**Report Title:**

Differences between the original LabVIEW and new Python Wet Test Fixture (WTF) applications.

**Contributions:**

Report Preparation: Isaiah Baker

**Client:**

Profound Medical Inc, 2400 Skymark Ave #6, Mississauga, ON L4W 5K5, Canada

**Date:**

August 2nd, 2022

In February 2022 8fold Manufacturing was contracted by Profound Medical to create a new graphical user interface for their existing and future Wet Test Fixture (WTF) systems. The goal was to transition from a closed-source LabView interface to a modular, easy-to-maintain, and well documented Python application that can be improved and kept up to date by either 8fold or Profound in the future.

Several months after the project was contracted it was expanded to include the construction of an additional WTF system, using a combination of parts provided by Profound and parts purchased by 8Fold. The system includes a new motor controller assembly, featuring a Galil 4123 controller. This decision was made because the Parker VIX 250 IM Drive controllers are end of life and use obsolete software. In addition, testing with them showed that communication with them is slow and unreliable by comparison. The new motor controller assembly is fully backwards compatible with the existing systems. Pending complete validation of the new software and new motion controller system, existing WTF systems will be able to be seamlessly upgraded at Profound’s discretion, making them more futureproof, reliable, and easy to maintain.

Prior to this report’s preparation the software was tested on the newly finished WTF system in Dallas, and preliminary data comparing the results from the new hardware and software was collected. There were also tests done on a production Wet Test Fixture with both the new Python software and original LabVIEW software, to determine if there are any differences in how the software itself collects and analyzes data. The conclusion of the latter report was that there was no relevant systematic difference between the results from the new Python software and the old LabVIEW software, and the random variability observed was within normal limits.

The goal of this report is to explain the known differences in behavior and data analysis of the new Python software and the original LabVIEW software. It is believed that these differences are justified due to hardware compatibility, incomplete information about the original software’s design, or opportunities for improvement.

* Homing X now uses the negative limit instead of the home switch.

Justification: The wiring of the home switch differs between wet test fixtures, and the new motion controller box is not compatible with all of them (for instance WTF2). Homing on the negative limit works just as well, using a 2-step routine which first moves the motors until they reach the negative limit, and then reverses the direction slowly until it is no longer pressed.

* The results include additional pass/fail states, ‘no test’ and ‘DNF’, instead of defaulting to ‘pass’

Justification: If a test was terminated early or otherwise failed with the LabVIEW software, all elements that did not finish a test and often the UA itself would say ‘pass’ in the results summary, which is misleading and would cause issues if the data were to be analyzed programmatically. Rather than defaulting to ‘pass’, the Python software defaults to ‘no test’, until a measure element efficiency RFB step begins and then changes to ‘DNF’ until the step is complete. This is more descriptive.

* Many hardware or software errors no longer necessitate a full restart of the software.

Justification: In the original software many error states, for example a device failing to connect on application launch, either caused the application to terminate itself, or freeze and require the user to restart the application, sometimes having to kill the process in task manager. In most if not all of these cases, the python software handles the error more elegantly, providing an abort/retry/continue dialog with helpful info, or at least providing a descriptive error message in the console and wtf.log file.

* Results are saved by default when a script is aborted early.

Justification: This is intended to minimize the chance of data loss in the event of a hardware or software issue, or if a script is terminated early for the sake of time.

* Random Uncertainty and Total Uncertainty replaced with Standard Deviation in the RFB tab

Justification: These quantities in the original software were not defined and there was no unambiguous definition for them in this context. Instead, the Python software uses the standard deviation of the data as a means of detecting when uncertainty is too high or there has been a disturbance. For example, if a UA on interval shows a standard deviation of greater than .02 Watts, it will retry once and then prompt the user to abort, retry, or continue. This ensures that if an element passes its test the data is stable and reliable. There is a function used as a placeholder for these values, so if the definition of these quantities is provided in the future it will be easy to update these in the output file.

* Determination of acoustic power on and acoustic power off

In the original software, there is an algorithm which takes in the raw acoustic power trace (in which the acoustic power turns off and on a specified number of times, usually 3) and calculates one acoustic power measurement for each interval. The original software does this by using an algorithm to detect transitions in the signal, the details of which are unknown. It appears to detect when the signal crosses a certain threshold and define that as the "acoustic power on", and when it crosses a certain threshold towards the end of an interval it calls that the "acoustic power off” and average the two transitional powers to yield the acoustic power within the interval. It also extrapolates from the data in mysterious ways, plotting a data trace that loosely follows the raw data and changes throughout the test's duration. Users have noted that this extrapolated data can behave in odd ways and sometimes shows artefacts or transitional data which invalidates a test.

Rather than mimicking this extrapolation approach with incomplete information about how it works and why, the python software uses a more straightforward approach for calculating the acoustic power within each interval. For each UA on interval and UA off interval, the software excludes a fixed settling time from the beginning, and a fixed number of samples from the end, and averages the remaining data. This was set to 2 seconds for the testing so far, but is adjustable in the "local.yaml" config file. The goal of this is to exclude transitional data while the balance is still reaching equilibrium. To ensure that the balance reached stability and the test is free of disturbances, the standard deviation of the data is taken, and the test is rejected if it is greater than 0.02 Watts (or the value specified in the config).

The acoustic power measurement for each interval is calculated by subtracting the measurement from the UA off interval from the measurement for the UA on interval, effectively zeroing the power data.