User interface Stability Test

Author: Frances Hutchings

Date: 17/07/2020

Reviewer: Date:

**Scope:**

The aim of this report is to demonstrate the stable operation of the Graphical User Interface for the CANDO *in vitro* box.

# System Overview

A schematic overview of the user interface is given in Figure 1. The graphical user interface (GUI) is designed to allow users to set up closed or open loop experiments using the *in vitro* control box. It can be used with serial port interfaces or with test signals from a host machine.

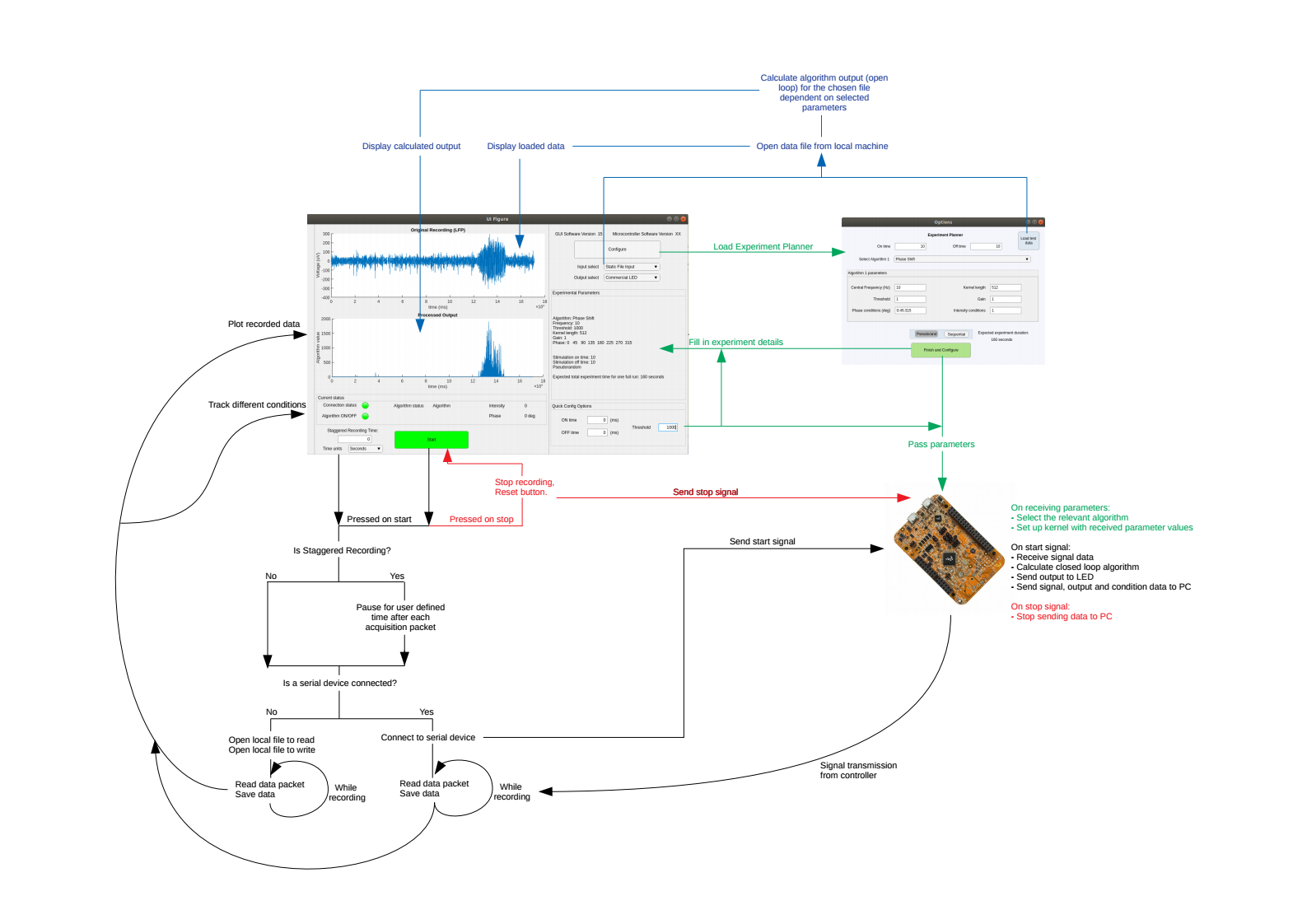


Figure *1:* System overview. The desktop control system requires a connection to external amplifier hardware for recording from and stimulation to brain tissue. The control system is able to record and process signals generated from the local machine, or fed into a static file in the local machine, for easier testing. It can also read a pre-recorded signal from the host machine and calculate the expected algorithm output given the currently selected parameters.

# Experiments Conducted and Experimental Set-up

In order to test the stability of the user interface, the program was left running for 24 hours with a test signal input from the local machine. The test signal was a simple sine wave, generated by a local script which saves six bytes, the first of which is the current value of the sine wave, to a binary file on the local hard drive. This is so that the signal replicates the data packet sent by the microcontroller in the *in vitro* box system. Figure 2 shows a screenshot of the user interface displaying the test signal, a sine wave, and the zero values algorithm output.

When the binary file on the local machine reaches the number of bytes that would trigger a read from a serial input in the user interface (6000 by default, 1 seconds worth of recording in when the sampling rate is set to 1000Hz with a data packet of 6 bytes) then the signal is read by the user interface, and the local binary file is reset. The sine wave script includes a 0.1 second pause in each loop to prevent generating too large an amount of data too quickly, which would likely cause out of memory issues in a run of this length.

The binary output of the test signal is also saved in another local file, for comparison with the user interface save files at the conclusion of the long run. If the user interface is working as expected then the saved sinusoidal input signal should be identical in the local binary file and in the user interface saved results.

Stability tests of the user interface were conducted with both Windows and Linux machines. The first test used a Windows 10 Home desktop computer with an Intel Core i5 processor and 32GB RAM. The second test used a Dell Latitude laptop, Newcastle University owned, with 32GB RAM and an Intel core i7 CPU, running Ubuntu 18.04.4 LTS. Using multiple platforms allows us to confirm that stability is platform independent.

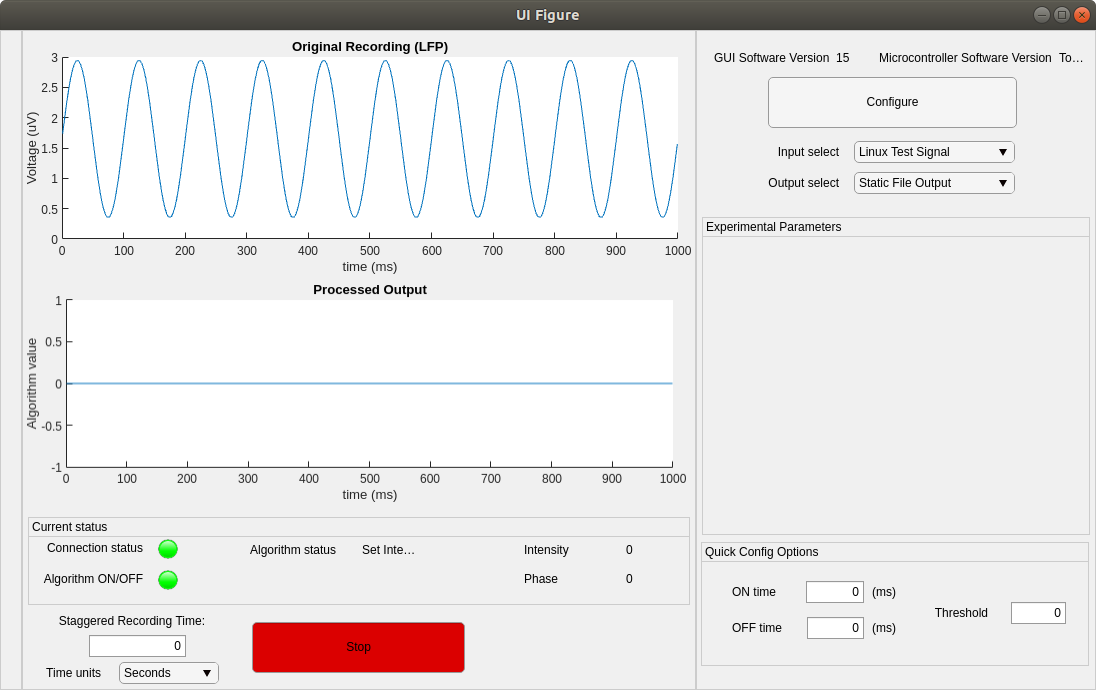


Figure 2: Screenshot of the user interface displaying the test signal and zero values algorithm signal during the 24 hour test

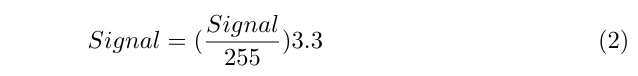
# Results

## Test signal comparison with user interface processed results

Figure 3 shows the test signal for the Windows and the Linux runs, to be compared with the results in Figures 4 and 5. The signals were generated using a Matlab script, with the function:

Where ω represents the frequency, set to 10 Hz, t is iterated from zero in steps of 0.001, and the round function rounds to the nearest integer. This produces a sin wave with integer values between 28 and 228, within the range of a single byte.

In the case of the Windows signal, each value was duplicated at each iteration of the signal generation. In the Linux signal the raw values were recorded. Within the user interface code signal from equation 1 is then transformed by the scaling factors in equation 2, to convert to the expected voltage range for a physiological signal.



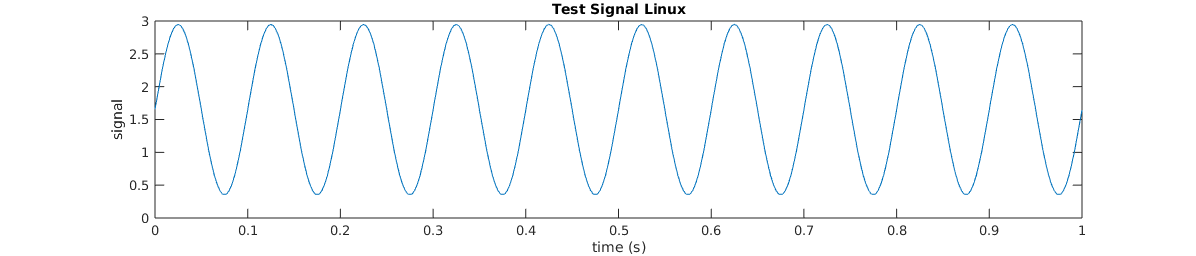
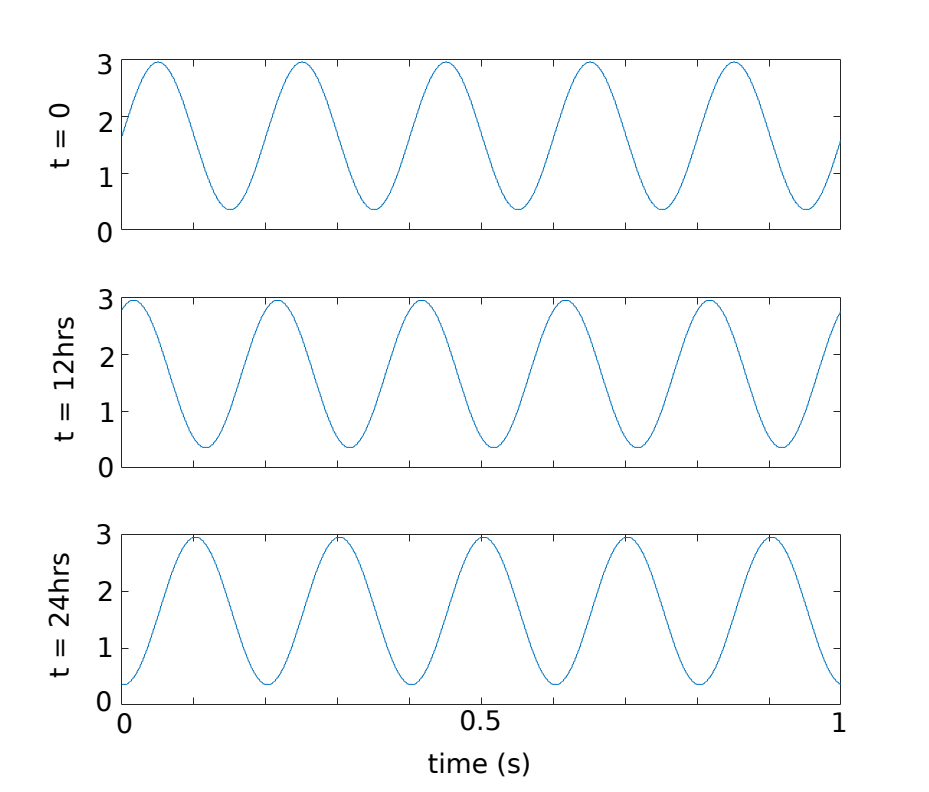
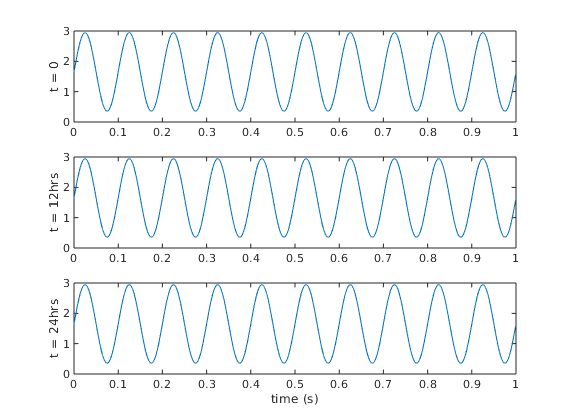
  
Figure 3: The 10Hz sine wave test signal used as an input to the user interface.

Figure 4 shows the results of the initial run on a Windows computer, plotted for different time points for both the user interface saved signal and the raw saved signal. These results show that the signal is captured by the user interface and saved from the beginning until the end of the experimental test. The signal appears to show a 5Hz oscillation as each data point of a 10Hz signal was duplicated in the data packet.

  
Figure 4: Windows tests with initial signal generating script, which had a byte loss with each data packet and a double value for each data point, resulting in a 5Hz signal. Snapshots of results over 24 hours.

The offset between the results is due to an issue with the code that produces the signal, as it was written to rest after the datafile is greater than the input buffer size of 6000 bytes (e.g. 1 second of the signal when the sampling rate is 1000Hz, as each time point produces a 6 byte packet). This meant that the data sent contained 6006 bytes, the final 6 of which were ignored by the user interface, and the next set of signal points would then be shifted by one time step. This issue has been corrected for future test runs, as shown in Figure 5 from the Linux test. However, including this byte loss version allows us to show that a signal that is effectively changing with each data packet sent has been effectively captured.

Figure 5 shows results from a second run with a corrected signal, where there is no phase shift evident. This shows that the stable signal is captured throughout the 24 hour run. There is no data loss over time, and the user interface continued to display and save the incoming signal accurately.

  
Figure 5: Results from the Linux run, with snapshots from different time points.

In order to quantify any data disparity, we saved the generated signal in binary form separately to the user interface, so that a direct comparison can be made. Figure 6 shows the differences over time for the full recorded time of the signal. As artificial delays were included in the signal generation this does not correspond directly to real time, so the time is unit-less in the figure.

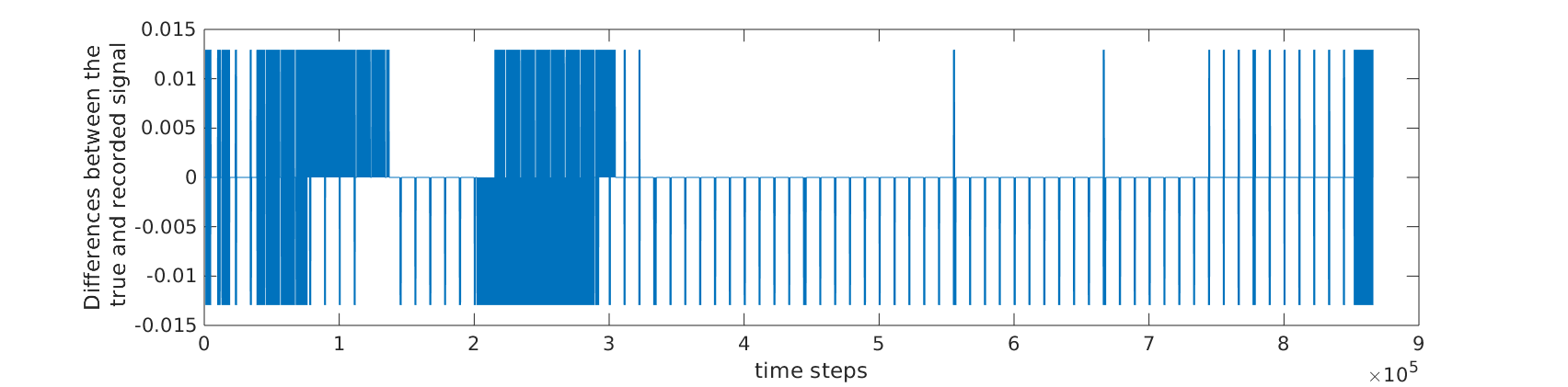
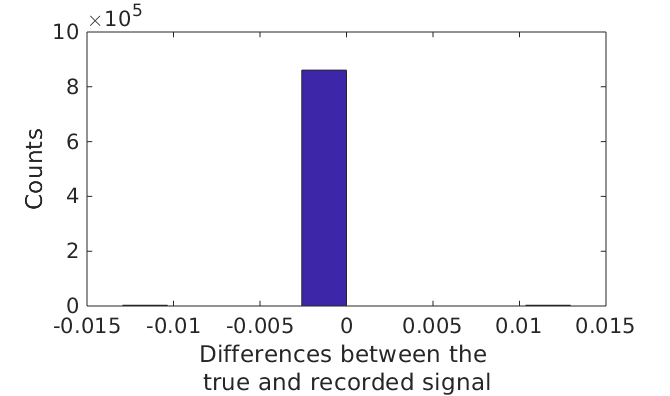


Figure 6: Disparities between the user interface saved signal and the raw input signal over time.

There are instances across the whole time series where there are minor disparities between the ‘true’ signal, as saved directly when generated, and the user interface recorded version of the signal. Figure 7 shows a histogram of the differences, it is apparent that the differences are very marginal, a +/-0.013 in a signal with a range from 0 to 3.3. The vast majority of values show differences around zero. These minor differences are not visible in plots of the raw data, as shown in figure 5.

  
Figure 7: Histogram of disparities between the true and UI recorded signal

## Memory Capacity

The tests carried out in this document did not result in any memory issues, as pauses added to the signal generator script on the host machine meant that there was a 1KB datafile produced roughly each minute, resulting in 1440KB of data, a very manageable amount.

However, in a less artificial situation we might expect to be producing more data at a much faster rate. Machines in the neuroscience laboratories have limited memory capacity, especially if Intan recordings are likely to be made at the same time, storing multichannel data rather than the single channel information that the user interface processes. In long term experiments it would be worth being aware of memory availability and a short test run can confirm the expected size and production rate of saved files that the user interface will produce.

## Automatic time outs

Whilst our results do show that the user interface appears able to perform for long periods of time without issue, there were some initial problems encountered with experiments due to settings on the computer triggering the host machine to go into a sleep state when left undisturbed for a given period of time. This is a common setting on many computers, and can be easily prevented by first checking the power saving settings.

In Windows 10 this setting can be found in the settings menu under Power & Sleep, where the sleep setting needs to be set to ‘never’. Similar settings will be available on other operating systems. The power saving settings for the screen turning off had no impact on the user interface, with recording continuing unperturbed. Similarly this can be set to never happen. On some university PCs there may be issues with automated log out times which would also need to be accounted for, as this would likely have an adverse impact on the user interface.

Further long term testing on the neuroscience laboratory computers would be useful to ensure proper performance in the setting where the user interface will be used for experiments. In current circumstances that has not been possible for this report (at the time of recording access to university laboratories is restricted due to the ongoing coronavirus pandemic).

# Conclusion

The user interface is able to perform without issue for long run times (up to 24 hours tested), and can accurately process and save an incoming signal. There were no issues noted during this test, although it is recommended that automatic updates and sleep settings are turned off on the host computer, and that sufficient memory space is available for any long term experiments to be carried out with the final *in vitro* box.