**Automated RAA Standard Operating Procedure**

**Ver 1.0**

## I Disclaimer:

The current document is distributed as supplementary material for the following publication (not peer-reviewed):

"Automated Reactive Accelerated Aging for Rapid In Vitro Evaluation of Neural Implants Performance" by Matthew G. Street, Cristin G. Welle, and Pavel A. Takmakov

doi: <https://doi.org/10.1101/204099>; web: <https://www.biorxiv.org/content/early/2017/10/17/204099>

The views, opinions, and/or findings contained in this document are those of the authors and should not be interpreted as representing the official views or policies of Department of Health and Human Services or the U.S. Government. The mention of commercial products, their sources, or their use in connection with material reported herein is not to be construed as either actual or implied endorsement of such products by the Department of Health and Human Services.

The authors assume no responsibility whatsoever for use of the automated Reactive Accelerated Aging (aRAA) by other parties and make no guarantees, expressed or implied, about the quality, reliability, or any other characteristic of the system. Further, use of the aRAA in no way implies endorsement by the FDA or confers any advantage in regulatory decisions. Any use of the aRAA is to be cited as stipulated in the license agreement. In addition, any derivative work shall bear a notice that it is sourced from the aRAA, and any modified versions shall additionally bear the notice that they have been modified.

## II Materials and Equipment:

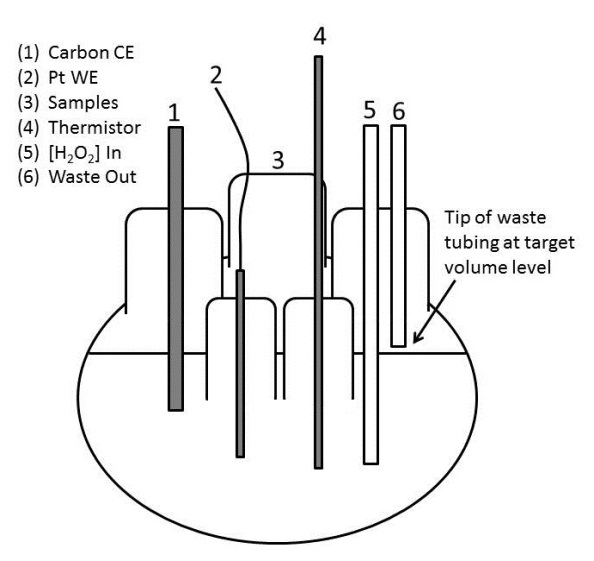
|  |  |  |
| --- | --- | --- |
| **Each reaction module:** | **Module with H2O2 control:** | **System:** |
| (1) Reaction vessel  (1) Temperature controller  (1) Heating mantle  (1) 2250 ohm thermistor  (1) Magnetic Stirrer | (1) Peristaltic pump  (1) IORodeo Potentiostat  (1) 1/8 x 1/16” PTFE tubing (~2 meters)  (1) 1/8 x 1/16” soft Viton tubing for pumps (~.25 meter) | (1) Raspberry Pi running Python script  (1) Flask containing concentrated hydrogen peroxide  (1) Empty flask for waste |

Preparing system:

**Tubing:**

If necessary, replace system tubing. Use new tubing and minimize the length of tubing used by keeping the pumps and flasks in close proximity.

**Concentrated Hydrogen Peroxide:**

Prepare 1500 mMol hydrogen peroxide in 1x PBS solution. Prepare sufficient volume of solution for the experiment to be conducted (typically 400 mL per reaction module). Make sure the PTFE tubing which draws out concentrated solution is positioned at the bottom of the flask and cut on an angle to avoid occlusion.

**Fluid Management:**

Delivering concentrated hydrogen peroxide and removing waste from solution is done simultaneously with a single peristaltic pump. This requires a pump with a head with multiple channels. Ideally, the pump head should have three channels: one for delivering concentrated hydrogen peroxide and two for removing waste. By removing twice the volume from the vessel as that of hydrogen peroxide delivery with the tip of the waste tubing at the desired level of vessel solution guarantees that the volume will stay the same. Verify the orientation of tubing in pump such that it draws from reaction vessel to waste while simultaneously drawing from concentrated hydrogen peroxide to the reaction vessel. There will be two PTFE tubes in the reaction vessel for fluid management (one inlet and one outlet) with placement shown in figure.

## III Operation:

**Temperature:**

Place a clean reaction vessel in each heating mantle with 1x PBS solution of target volume. Place the thermistors in the corresponding reaction vessels and power on the PID temperature control devices. Heating mantles should be plugged into output 2 of the PID controllers. Thermistors should be connected with red line in input 2 and black line in input 3 on front panel of PID controller. If the setpoint needs to be changed, hit the C:\Users\Matthew.Street\Documents\RAA SOP\upButtonPID.PNG button twice then push left or right button until you see an option for **PRoG** (Programming Mode). Select the **SP1** parameter and use the left and right buttons to set the desired value then hit the return button. With the setpoint stored, hit the C:\Users\Matthew.Street\Documents\RAA SOP\upButtonPID.PNGagain until the menu returns to the **PRoG** option. Using the left and right buttons select the **oPER** (Operate) option then select **Run**.

Each PID controller should have a unique identification. To assign an id number:

1. Navigate in menu to **INIt>CoMM>USb>AddR**
2. Assign a unique number to the PID controller corresponding to the module to which it is connected e.g. 0001 for module 1
3. Repeat 1-2 for each PID controller used in the system

**IORodeo (Potentiostat):**

\*\*Note: As of 9/14/17 all software to operate IORodeo with RPi connected in WO62-G123 is installed.

\*\*Note: All Python scripts discussed herein are intended for Python 2.7.

The IORodeo comes pre-programmed with firmware and ready to use. Control of the device is accomplished using Python with libraries available online through bitbucket. The IORodeo is based on the Teensyduino (Arduino) device. In order to use the Teensy with a Raspberry Pi (Rpi) running Raspbian Jesse (Version of Debian), follow online directions found on PJRC.com or install Arduino IDE 1.8.x on RPi. Once communication can be established between RPi and Teensy, install IORodeo Python library as necessary (can be installed through PIP).

Refer to **stuff.iorodeo.com/docs/potentiostat** for detailed instruction for installation and use of potentiostat.

Each potentiostat should have a unique identification. To assign an id number:

1. Connect a single IORodeo potentiostat to the RPi
2. Open the file /pi/home/documents/RAA/Assign\_Potentiostat\_ID.py in the Python 2.7 IDE
3. Edit line 7 to correctly assign the desired ID e.g. for module 1 of RAA use **potentiostat\_ID = 1**
4. Repeat 1-3 for all potentiostats ensuring each is assigned a unique ID

Calibration of each potentiostat should be performed prior to each experiment as follows:

1. Connect a single IORodeo potentiostat to the RPi
2. Open the file /pi/home/documents/RAA/Calibrate\_Potentiostat.py in the Python 2.7 IDE
3. Edit line 7 to the previously assigned ID e.g. for module 1 of RAA use **potentiostat\_ID = 1**
4. Fill the clean reaction flask with 1x PBS
5. Place CE and WE in solution as detailed in figure above
6. The RE output of the potentiostat should be shorted to the CE
7. Connect the potentiostat to the electrodes via alligator clamps
8. Run the script by pressing **F5**. If prompted, save the script and run.
9. The potentiostat will begin continuous reading of the current using chronoamperometry.
10. Wait for at least two cycles of the current to be read and record the latest value.
11. Increase the H2O2 concentration by 5mMol
12. Repeat steps 10-11 until a sufficient curve can be regressed (typically 30mMol)
13. Perform linear regression analysis to convert current to concentration and record the equation in the form
14. Repeat 1-13 for all potentiostats to be used for RAA.

**Initializing Run:**

1. For each reaction module, place a clean flask filled with 1xPBS at target volume in heat mantle and begin heating with PID controller.
2. Connect all necessary components to reaction flask(s) except for test sample(s).
3. Double-check placement of waste tubing and orientation of soft tubing in peristaltic pump(s).
4. On the Rpi open file /pi/home/documents/RAA/RAA\_eChem\_RUN.py in the Python 2.7 IDE
5. Edit lines 20-30

\*\*Note: each list defined in lines 20-30 should have one element per reaction module. Each element for any given reaction module should have the same index in each list.

1. Run the script by pressing **F5**. If prompted, save the script and run.

\*\*Note: Initially, the potentiostat(s) and PID controller(s) do not need to be connected to the RPi

1. The script will begin printing current status and process values to the console.
2. Connect the potentiostat(s) and PID controller(s) to the RPi via USB.

\*\*Note: Given that the RPi only has four USB ports it may be necessary to use a powered USB expansion or to disconnect other peripherals (mouse/keyboard) in order to accommodate the devices.

## IV Troubleshooting:

**Temperature:**

If process value is reading wrong or not at all, first ensure the temperature sensor is in solution. If the error persists, ensure the correct temperature sensor is selected (refer to manual).

If process value is not reaching SV, first ensure the SV is set correctly (see above steps to change if necessary). If the problem persists, check that the correct input is stored in the PID controller by navigating to **INIT>INPt>tHRM** and select **2.25k**.

If heating problem does not resolve, run auto tune on the PID controller parameters (refer to manual).

**Python Script:**

The most common problem in Python is the USB communication. If the problem appears to be this type of issue, run file /pi/home/documents/RAA/USB\_Comm.py in the Python 2.7 IDE. This will list all available USB devices found and print the corresponding port name.

If serial communication problem persists with PID controller, ensure the correct communication method is being used by doing the following:

1. Navigate in menu to **INIt>CoMM>USb>PRot>oMEG>ModE**
2. Select **CMd**
3. Navigate in menu to **INIt>CoMM>USb>PRot>oMEG>dAt.F**
   1. **StAt>No**
   2. **RdNG>yES**
   3. **PEAK>No**
   4. **VALy>No**
   5. **UNIt>No**
4. Navigate in menu to **INIt>CoMM>USb>PRot>oMEG>\_LF\_**
5. Select **No**
6. Navigate in menu to **INIt>CoMM>USb>PRot>oMEG>ECHo**
7. Select **No**
8. Navigate in menu to **INIt>CoMM>USb>PRot>oMEG>SEPR**
9. Select **\_CR\_**

If serial communication problems persist with potentiostat, refer to documentation page:

http://stuff.iorodeo.com/docs/potentiostat/examples.html