# Supporting Documentation

# Method to Maintain Oxidative Capacity of Test Solutions for Oxidative Screening

## Tool Reference

RST Reference Number: RST26MC01.01

Date of Publication: 02/19/2026

Recommended Citation: U.S. Food and Drug Administration. (2026). *Method to Maintain Oxidative Capacity of Test Solutions for Oxidative Degradation Screening* (RST26MC01.01). https://cdrh-rst.fda.gov/method-maintain-oxidative-capacity-test-solutions-oxidative-degradation-screening

## For more information

Catalog of Regulatory Science Tools to Help Assess New Medical Devices

# Disclaimer

## About the Catalog of Regulatory Science Tools

The enclosed tool is part of the Catalog of Regulatory Science Tools, which provides a peer-reviewed resource for stakeholders to use where standards and qualified Medical Device Development Tools (MDDTs) do not yet exist. These tools do not replace FDA-recognized standards or MDDTs. This catalog collates a variety of regulatory science tools that the FDA's Center for Devices and Radiological Health's (CDRH) Office of Science and Engineering Labs (OSEL) developed. These tools use the most innovative science to support medical device development and patient access to safe and effective medical devices. If you are considering using a tool from this catalog in your marketing submissions, note that these tools have not been qualified as Medical Device Development Tools and the FDA has not evaluated the suitability of these tools within any specific context of use. You may request feedback or meetings for medical device submissions as part of the Q-Submission Program.

For more information about the Catalog of Regulatory Science Tools, email RST_CDRH@fda.hhs.gov.

# Supporting Documentation

# Method to Maintain Oxidative Capacity of Test Solutions for Oxidative Screening

## S1. Test Method Setup Guidelines

This appendix provides general guidelines for setting up the test method described in this RST, followed by a description of an experimental configuration for electrochemical monitoring and maintenance of hydrogen peroxide ($H_2O_2$) concentrations in buffered aqueous test solutions.

The general guidelines cover essential components such as the reaction vessel, heating and stirring apparatus, and electrochemical setup, offering specifications to help users select appropriate materials and equipment. These guidelines are intended to help users consistently maintain $H_2O_2$ concentration and oxidative conditions. Additionally, they discuss compatibility of components, accurate calibration, and proper sample handling, enabling users to achieve consistent and reproducible results across different test setups.

This test method includes a platinum working electrode, graphite counter/reference electrode, an open source potentiostat, IoT power relay, peristaltic pump, and Raspberry Pi microcomputer for control and data acquisition. Stepwise voltage cycling protocol, alternating between anodic potentials to induce $H_2O_2$ oxidation and cathodic potentials, is used to prevent electrode passivation. This cycling extends reliability in measuring $H_2O_2$ concentrations during prolonged experiments.

The control scripts for both the calibration and concentration maintenance procedures, along with the Python codes used for the respective experiments, are included in the Supporting Documentation. These resources are intended to help users implement the test system.

### 1.1. Apparatus

These guidelines are limited to components physically or functionally identical to those described below. Use of alternative components may necessitate evaluation of the system including but not limited to modifications to electrochemical parameters, calibration processes and overall system functionality. Statements made in this RST about the current system and method below may not be directly applied to alternative situations.

*1.1.1 Reaction Vessel:* Select a vessel for reactions that can accommodate the required test solution volume, with ports for the insertion of electrodes, test samples, and tubing for adding and removing solutions. It is important that the materials for the vessel be (i) stable under experimental conditions and not interfere with test substances or alter the test environment, (ii) be safely heatable to 100°C (±1°C) and (iii) robust to continuous stirring to maintain a homogeneous test solution. For long-duration experiments, appropriate sealing methods should be used to prevent evaporation and maintain test consistency. A 125 mL five-neck round-bottom glass flask was used in this test method.

*1.1.2 Heating and Stirring Apparatus:* The heating and stirring system needs to maintain the solution temperature within a range of ambient to 80°C, with an accuracy of ±1°C. Stirring ensures uniform exposure of the test specimens to oxidative agents. Various heating methods may be employed, provided they meet temperature control needs. An oil bath on a hotplate stirrer was used for stable heating in this test method.

*1.1.3 Electrochemical Components:* The electrode assembly connected to a potentiostat and linked to a data acquisition system, such as a microcomputer, is expected to have both a working electrode (e.g., platinum) and a counter/reference electrode (e.g., graphite rod), ensuring proper immersion and positioning in the test solution to avoid variability in measurements. Consistency in electrode placement is important for both calibration and degradation screening. A 100 µm Pt working microelectrode and a 6.35 mm graphite rod counter electrode were used and connected to a Rodeostat potentiostat controlled by a Raspberry Pi microcomputer.

*1.1.4 Solution Handling:* The setup described herein allows for the controlled addition of concentrated $H_2O_2$ solution and the removal of excess solution to maintain a constant volume. This is helped by selecting tubing for solution handling compatible with the test solutions and designed for precise control of flow rates. Stock solutions with $H_2O_2$ concentrations exceeding ~2M are likely to introduce undesired variability in the stability range of $H_2O_2$. PFA tubing (1.59 mm inner diameter) connected to a 4-channel peristaltic pump were used for both adding $H_2O_2$ stock solution and removing excess test solution from the reaction flask.

*1.1.5 Sample Securing Mechanism:* Securely position test specimens within the reaction vessel, ensuring uniform exposure to the test solution without contact with vessel surfaces or stirring mechanisms. Buoyancy or movement could affect exposure to oxidative conditions. UHMWPE blocks (10 mm x 10 mm x 28 mm) were secured with polytetrafluoroethylene (PTFE) tape around a glass rod, ensuring no interference with electrode or solution dynamics.

*1.1.6 Data Recording and Control:* Choose a microcomputer or equivalent system that can execute control scripts for data acquisition, manage hardware components, and monitor experimental conditions. The data acquisition system should support long-term monitoring of electrochemical activity, with the ability to adjust parameters in real-time. A Raspberry Pi microcomputer was used to run Python scripts and to manage data acquisition and control the peristaltic pump for $H_2O_2$ adjustments in the reaction flask.

## 1.2. Reagents

1.2.1 Unless otherwise indicated, best results are expected if all reagents conform to the specifications of the Committee on Analytical Reagents of the American Chemical Society where such specifications are available. Other grades may be used, provided it is first ascertained that the reagent is of sufficiently high purity to permit its use without lessening the accuracy of the determination.

1.2.2 *Hydrogen peroxide ($H_2O_2$)* – Dilute a reagent grade hydrogen peroxide solution using the aqueous buffer solution to required concentrations for calibration and test procedures. Verify the starting concentration of $H_2O_2$ using any relevant analytical techniques. The concentration of the $H_2O_2$ stock solution will exceed the concentration of the test solution. Depending on the starting concentration of $H_2O_2$, the buffer solution will be diluted to different degrees. Account for this in test solution preparation to maintain the desired buffering capacity and salt concentrations.

1.2.3 *Buffer solutions* – Use standard buffer solutions (e.g. 1X phosphate buffered saline) to provide a consistent ionic strength and pH for the test solution. See ISO 13781 and ASTM F1635 for examples of buffers used in in vitro degradation studies.

1.2.4 *Purity of Water* – Unless otherwise indicated, references to water indicate reagent water as defined by Type IV of Specification ASTM D1193.

**1.3. Test Specimens**

1.3.1 Select test specimen size and shape so that it can be accommodated within the selected reaction flask without making physical contact with the electrodes or the stirrer. Consider the test specimen geometry, dimensions, location in the reaction flask, and placement relative to electrode positions to ensure fixed positioning. Consistent exposure and uniform interaction throughout the testing period are accomplished if the test sample is fully immersed in the test solution.

1.3.2 Select test specimen materials (and their possible contaminants) that are electrochemically inert and do not undergo unintended reactions, corrosion, or degradation that can interfere with the electrochemical environment and affect the accuracy of the measurements. For this method to work, electrochemical reactions will primarily involve the target species (e.g., $H_2O_2$) rather than the test specimen, preventing unintended contributions to the measured currents.

1.3.3 Carefully consider and keep consistent the surface area-to-volume ratio of the test specimen and the test solution for reproducible and comparable results.

1.3.4 Orient and secure the test specimen within the reaction flask to minimize spatial and temporal variations in exposure to oxidative agents. If multiple specimens are in the same vessel, position them in a manner that avoids shadowing by other specimens or flask components, ensuring even distribution of reactive species across the specimen surfaces.

**1.4. Procedure**

*1.4.1 Test Method Assembly:*

*1.4.1.1 Computing and Potentiostat Configuration* - Install the Raspberry Pi OS (formerly Raspbian), including Python and the necessary libraries for the potentiostat's operation. Connect the Rodeostat potentiostat to the Raspberry Pi for control and data collection and calibrate it per manufacturer's recommendations. Configure the system to enable data acquisition, electrochemical measurements, and the operation of hardware components like the power relay and peristaltic pump.

1.4.1.2 *Power Relay and Fluid Management System Connection* – Connect the IoT external power relay to the Raspberry Pi using GPIO pins. The peristaltic pump is connected in the "Normally OFF" position on the power relay. Use perfluoroalkoxy (PFA) tubes (1.59 mm ID, 3.17 mm OD) for solution handling. Two channels deliver the concentrated $H_2O_2$ stock solution into the reaction flask, and two channels remove excess solution. Carefully position the tube just above the initial solution level to efficiently remove excess solution when needed. Set the rpm for the peristaltic pump to a low value to prevent excessive pumping of concentrated $H_2O_2$, which could broaden the $H_2O_2$ concentration range in the reaction flask. If there are differences in flow rates among the channels, prioritize the channels with higher flow rates for removing excess solution. This precaution prevents flooding of the reaction flask when all channels are activated simultaneously. Ensure that all components of the fluid management system are compatible with the reagents used in the experiment and can operate safely under the experiment's conditions. Select materials to resist corrosion, and secure connections to prevent leaks.

1.4.1.3 *Reaction Flask Preparation* – Securely clamp a clean five-neck 125 mL round-bottom flask in an oil bath on a hot plate, pre-set to the desired temperature. Add 120 mL of 1x PBS or similar buffer solution, ensuring that the test samples and electrodes are fully submerged. Allow 30 minutes for the solution and flask to reach thermal equilibrium at the set temperature before adding the required amount of $H_2O_2$ solution. This approach minimizes the decomposition of $H_2O_2$, ensuring the accuracy of the test solution's concentration at the start of the experiment.

1.4.1.4 *Electrode Installation* – Position the platinum (Pt) working electrode and the graphite rod, which serves as both the counter and reference electrode, in the opposite ports of the reaction flask. Ensure the immersion depth of the electrodes in the test solution is consistently maintained, as variations in the amount of electrode surface area in contact with the solution can affect measurement outcomes. Avoid electrode contact with each other or the vessel walls. Establish a secure connection between the potentiostat and the electrodes using alligator clamps.

1.4.1.5 *Sample Introduction and Securing* – Prior to initiating the experiment, carefully introduce the test samples into the reaction vessel, ensuring they are appropriately positioned for uniform exposure to the test solution. Given the variability in polymer densities, some samples may exhibit buoyancy issues. To mitigate this, securely anchor samples at the bottom of the vessel using suitable, non-reactive holders or weights that do not interfere with the electrochemical measurements or the solution's composition. Choose materials for anchoring that are electrochemically inert and compatible with the test solution. After introducing the polymer sample, seal the reaction flask (e.g. using PTFE tape or parafilm) to minimize water evaporation and maintain the solution's volume and concentration throughout the test.

1.4.1.6 *System Integration Check*- Ensure that all components – Raspberry Pi, potentiostat, IoT power relay, and peristaltic pump —are properly integrated and functioning. Verify that the RPi can execute the Python code for calibration (see Supporting Documentation, Section S1) and $H_2O_2$ concentration maintenance (see Supporting Documentation, Section S2). Control scripts are provided in the Supporting Documentation to guide users through the calibration (Section S3) and concentration maintenance procedures (Section S4). Ensure the system runs smoothly before initiating experiments. See Fig.1 for the test system schematic.

.

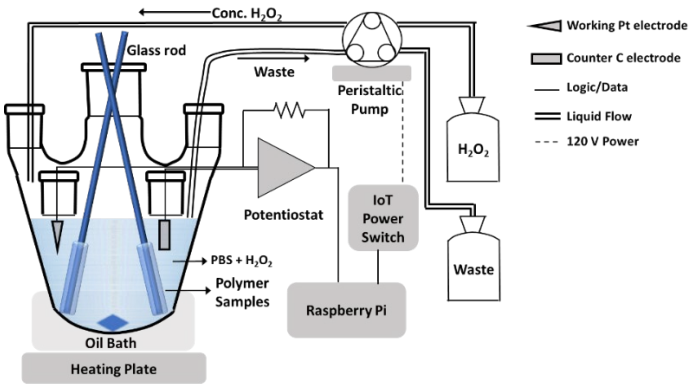


**Figure 1:** Schematic of the test system for monitoring and maintaining $H_2O_2$ concentration in test solutions.

*1.4.2 Calibration between measured current and $H_2O_2$ concentration*

*1.4.2.1 Calibration:* Configure the test system following the guidelines provided in Section 4.1. Execute the Python script for the calibration procedure (Supporting Documentation, Section S1) through either a command line interface or any available Python integrated development environment (IDE) on the Raspberry Pi. Before starting the calibration, ensure that all experimental details are entered, including electrochemical settings for detecting and monitoring $H_2O_2$ and the data file names for current measurements. Execute the code and wait for stabilization of current values. Allow recording of current values at the selected $H_2O_2$ concentration for 2 – 5 minutes or until a statistically significant dataset is acquired to facilitate accurate average value calculation. Note that the chronoamperometry experimental parameters in Lines 17-23 in Supporting Documentation, Section S1 are optimized for the example two-electrode setup. Adjustments may be necessary if alternative electrochemical components are employed.

*1.4.2.2 Solution Handling:* Gradually increase the $H_2O_2$ concentration in predefined increments (e.g., 5-50 mM) in the reaction vessel. After each increment, repeat the calibration process described in Section 4.2.1, recording the current response at each stage. Continue until you have collected enough data for a comprehensive calibration curve. Use linear regression to analyze the data and establish the relationship between current intensity and $H_2O_2$ concentration. Record the regression equation for use in future concentration measurements.

*1.4.3 Maintaining and Monitoring $H_2O_2$ concentration*

*1.4.3.1 Concentration Maintenance* - Configure the test system following the guidelines outlined in Section 4.1. Open the designated monitoring script (Supplemental Information, Section S2) on the Raspberry Pi microcomputer, using a command line interface or a Python IDE. Ensure that the script is pre-configured with the experiment's name (Line 11), target $H_2O_2$ concentration range (Line 12), GPIO control pin that's controlling the external power relay (Line 38), and the linear regression equation from the calibration phase (Line 18). Ensure the electrochemical parameters match those used in the calibration curve experiment. Execute the Python code, which continuously monitors current, converts the data into $H_2O_2$ concentration, and maintains the concentration by controlling the peristaltic pump via the power relay when necessary.

*1.4.3.2 Concentration Verification* - In addition to electrochemical monitoring, periodically verify $H_2O_2$ concentration using alternative methods such as spectrophotometry, titration, or enzymatic assays. One method is to collect 1-5 mL aliquots from the test solution and analyze using UV-Vis spectroscopy to measure absorbance at 250 nm, which is characteristic of $H_2O_2$. For titration, standards such as ASTM D2180 or USP 29 describe methods using potassium permanganate. Regular verification ensures the accuracy of the electrochemical data and confirms that concentration adjustments are based on reliable information. When selecting the verification method, prioritize sensitivity, specificity to $H_2O_2$, and compatibility with the test solution. Regular verification ensures the accuracy of the electrochemical data and confirms that concentration adjustments are based on reliable information.

**1.5. Report**

Report the optimization and calibration process, including documentation of the repeatability of the calibration through periodic calibration tests throughout the test duration to ensure consistent electrode performance and stability of the $H_2O_2$ concentration measurements. Additionally, document any deviations observed and adjustments made during prolonged use, such as modifications to electrochemical parameters or preconditioning processes to restore and maintain electrode activity since electrode performance may change over time. Include calibration curves, accuracy of $H_2O_2$ concentration maintenance, and any corrective actions taken to address performance issues.

*1.5.1 Assembly of the Test System*- In the written ,report, include a detailed description of how the test system was assembled, including the following components:

*1.5.1.1 Electrode Arrangement* – Include the manufacturer and specifications of the Pt working microelectrode and of the graphite rod counter electrode, the condition of their surface, any preconditioning steps, and their positions in the reaction flask.

*1.5.1.2 Peristaltic Pump Connection* – Outline the setup of the peristaltic pump and describe the positioning of tubing for both the delivery and removal of $H_2O_2$ stock solution and excess solution.

*1.5.1.3 Microcomputer and Potentiostat Incorporation* – Describe how the microcomputer is integrated with the system for control and data acquisition. Specify the connection and configuration of the potentiostat with the microcomputer.

*1.5.2 Calibration Experiment Details*- Summarize the steps undertaken in the calibration experiment to establish the relationship between measured current and $H_2O_2$ concentration. Specify the electrochemical parameters used during the calibration experiment, including cathodic and anodic voltages, step times, and any other relevant settings. Report the experimental conditions, such as temperature, pH, and solution composition, during the calibration to ensure reproducibility.

*1.5.3 Monitoring and Maintaining $H_2O_2$ concentration Experiment Details*- Outline the main experiment procedures for continuous $H_2O_2$ concentration monitoring, including the voltage protocol for generation of current at the electrodes and the experimental conditions, such as temperature, pH, solution composition, and test duration. Include details on any user-defined parameters, such as the target concentration, experiment name, and linear regression equation obtained from the calibration curve experiment.

*1.5.4 Electrode Maintenance and Calibration*- Describe the procedures for electrode maintenance, including cleaning and calibration steps, to ensure the reliability of measurements. Include information on the frequency of electrode calibration and any criteria used to determine the need for recalibration.

*1.5.5 Solution Preparation and Handling*- Include details on how the $H_2O_2$ solutions for calibration and the $H_2O_2$ stock solution for concentration maintenance procedures were prepared, including concentrations and volumes.

*1.5.6 Data Analysis and Interpretation*- Present the methods used for data analysis, including any software tools or algorithms employed to convert recorded current values into $H_2O_2$ concentrations. Discuss the interpretation of results, addressing any observed trends, fluctuations, or anomalies in the continuous monitoring data.

*1.5.7 System Validation*- Provide evidence of system validation, demonstrating the reliability and accuracy of the test method. This may include comparisons with known concentrations, replicability studies, or validation against alternative analytical methods such as UV-Vis Spectroscopy or titration against potassium permanganate.

## S2. Python Code for Calibration Between $H_2O_2$ Concentration and Current Generated at the Electrodes

S2.1 This section presents the Python code to facilitate the calibration between $H_2O_2$ concentration and the current generated at the electrodes in response to the selected voltage protocol. It outlines the process of initializing the electrochemical system, setting calibration parameters, applying voltage protocols, and processing the resulting data to establish a calibration curve.

S2.2 Note that this code is written in Python 2.7. Given the discontinuation of support for Python 2.7 and potential compatibility issues with newer Python versions, users are encouraged to adapt the code to be compatible with their operational environment, preferably using a more recent version of Python, such as Python 3.x. This adaptation may involve syntax changes and updates to library calls to ensure the script's functionality in maintaining $H_2O_2$ concentration is preserved across different experimental setups and software configurations.

```
1 | # Import necessary modules
2 | from potentiostat import Potentiostat
3 | import time
4 | import serial
5 | import serial.tools.list_ports
6 | import traceback
7 | potentiostat_ID = 0

8 | # Electrochemical Run Parameters:
9 | # These parameters define the settings for the potentiostatic experiment (electrochemical run) for each reaction module.
10 | # Each module has a dictionary specifying the current range, voltage range, voltage limits, and corresponding time durations.
11 | # - "curr_range": Current range for the potentiostatic experiment (e.g., '1000uA' for 1000 microamperes).
```

```
# - "volt_range": Voltage range for the potentiostatic experiment (e.g., '1V' for 1 volt).
# - "low_volt": Lower voltage limit during the experiment in volts.
# - "high_volt": Higher voltage limit during the experiment in volts.
# - "low_volt_time": Duration of the lower voltage limit in seconds.
# - "high_volt_time": Duration of the higher voltage limit in seconds.

curr_range = '1000uA'
volt_range = '1V'
low_voltage = -0.3
low_time = 0.5
high_voltage = 0.7
high_time = 2
sample_period = 0.1

# Define output file for results
filename = 'Add File Name here.txt'  # Adjusted filename

try:
    # Initialize potentiostat device
    dev = None
    ports = list(serial.tools.list_ports.comports())
    for p in ports:
        print 'Checking port %s' % (str(p))
        try:
            dev_temp = Potentiostat(p[0])
            cT = dev_temp.get_device_id()
            if cT == potentiostat_ID:
                dev = dev_temp
            print 'Connected...'
        except:
            print 'Skipping...'
            pass

    if dev:
        # Configure potentiostat for chronoamperometry
        print 'Running chronoamperometry on potentiostat with device ID: %s' %
(str(dev.get_device_id()))
        dev.set_all_elect_connected(True)
        dev.set_volt_range(volt_range)
        dev.set_curr_range(curr_range)

        # Define variables for data collection
        list_length = 5
        curr_list = None

        while True:
            stTime = time.time()
            run_pot = True
```

```
53 |          hold_period = False
54 |          sample_time = time.time()
55 |          num_samples = 0
56 |          total_curr = 0

57 |          # Cycle for low_time + high_time
58 |          dev.set_volt(low_voltage)
59 |          while run_pot:
60 |             if time.time() - stTime > low_time:
61 |                # Start holding high voltage
62 |                if not hold_period:
63 |                   dev.set_volt(high_voltage)
64 |                   hold_period = True

65 |                # Sample every sample_period seconds
66 |                if time.time() - stTime > low_time + (high_time/2):
67 |                   if time.time() - sample_time > sample_period:
68 |                      total_curr += dev.get_curr()
69 |                      num_samples += 1
70 |                      sample_time = time.time()

71 |             # Exit cycle
72 |             if time.time() - stTime > low_time + high_time:
73 |                run_pot = False

74 |          # Calculate moving average of sampled currents
75 |          if curr_list is None:
76 |             curr_list = [total_curr/num_samples]
77 |          elif len(curr_list) < list_length:
78 |             curr_list = curr_list + [total_curr/num_samples]
79 |          else:
80 |             curr_list = curr_list[1:len(curr_list)] + [total_curr/num_samples]
81 |          curr = sum(curr_list)/len(curr_list)

82 |          # Write current value to the file
83 |          with open(filename, 'a') as fp:
84 |             fp.write("%f\n" % curr)
85 |          print "%f" % (curr)
86 |    else:
87 |       print 'Could not connect potentiostat with device ID: %s' % (str(potentiostat_ID))
88 | except Exception as e:
89 |    # Print traceback if an exception occurs
90 |    traceback.print_exc()
91 | finally:
92 |    # Ensure that the test is stopped on all available ports
93 |    ports = list(serial.tools.list_ports.comports())
94 |    for p in ports:
95 |       try:
```

```
96 |         dev = Potentiostat(p[0])
97 |         dev.stop_test()
98 |       except:
99 |         pass
```

### S3. Python Code for Monitoring and Maintaining $H_2O_2$ Concentration

S3.1 This section presents the Python code used for maintaining $H_2O_2$ concentration within specified electrochemical experiments. The code exemplifies the integration of software control with the experimental setup to regulate $H_2O_2$ levels dynamically. It encompasses initializing the system, setting experimental parameters based on prior calibration, conducting continuous electrochemical monitoring, and activating adjustment mechanisms to maintain the target concentration.

S3.2 Note that this code is written in Python 2.7. Given the discontinuation of support for Python 2.7 and potential compatibility issues with newer Python versions, users are encouraged to adapt the code to be compatible with their operational environment, preferably using a more recent version of Python, such as Python 3.x. This adaptation may involve syntax changes and updates to library calls to ensure the script's functionality in maintaining $H_2O_2$ concentration is preserved across different experimental setups and software configurations.

```
1 | # Import necessary modules
2 | import RPi.GPIO as GPIO
3 | import time
4 | from datetime import datetime
5 | import smbus
6 | import serial
7 | import serial.tools.list_ports
8 | import traceback
9 | from potentiostat import Potentiostat

10 | # Vessel Parameters
11 | experimentName = ['Add file name here']
12 | experimentRunParameters = [[145]]  # Set target H2O2 concentration here
13 | average_value = 20 # This calculates the running average of measured H2O2 concentration

14 | # Calibration parameters for converting current to H2O2 concentration
15 | # The values below (12.292 and 0.1999) are examples and should be edited based on the user's calibration experiment.
16 | # These values represent the y-intercept and slope of the linear regression between current and H2O2 concentration.
17 | # Calibration Experiment: y = mx + b, where y is H2O2 concentration, x is current.
18 | experimentCurrToConcFunctions = [lambda x: (x - 12.292) / 0.1999]

19 | # Electrochemical Run Parameters:
20 | # These parameters define the settings for the potentiostatic experiment (electrochemical run) for each reaction module.
21 | # Each module has a dictionary specifying the current range, voltage range, voltage limits, and corresponding time durations.
```

```
# - "curr_range": Current range for the potentiostatic experiment (e.g., '1000uA' for 1000 microamperes).
# - "volt_range": Voltage range for the potentiostatic experiment (e.g., '1V' for 1 volt).
# - "low_volt": Lower voltage limit during the experiment in volts.
# - "high_volt": Higher voltage limit during the experiment in volts.
# - "low_volt_time": Duration of the lower voltage limit in seconds.
# - "high_volt_time": Duration of the higher voltage limit in seconds.
echemRunParameters = [{"curr_range": '1000uA',
                       "volt_range": '1V',
                       "low_volt": -0.3,
                       "high_volt": 0.7,
                       "low_volt_time": .5,
                       "high_volt_time": 2}]

# GPIO pins corresponding to peristaltic pump control
# Pin 17 is an example and should be edited based on the user's setup.
# This pin has positive voltage and is connected to an external power relay
# that controls the peristaltic pump.
pumpPins = [17]

defaultPath = '/home/pi/potentiostat-master/software/python/potentiostat/RAA'

# Sampling rate for collecting data from the potentiostat. This is the rate
# at which the system reads potential and current data from the potentiostat.
# For example, the code below specifies the sampling rate (sampleRate) as float(2.5**-1),
# which is equivalent to 1/2.5 or approximately 0.4 Hz.
# This means that the system collects data from the potentiostat every 1/0.4 seconds,
# or approximately every 2.5 seconds.
sampleRate = float(2.5**-1)

# Data write rate for storing collected data to a file. This is the rate
# at which the system writes the collected data to a file.
# For example, the code below specifies the sampling rate (dataWriteRate) as float(15**-1),
# which is equivalent to 1/15 or approximately 0.067 Hz.
# This means that the system writes data to the file every 1/0.067 seconds,
# or approximately every 15 seconds.
dataWriteRate = float(15**-1)

# I2C bus setup
bus = smbus.SMBus(1)
numberOfRAAs = len(experimentName)
allH2O2Conc = [[] for _ in range(numberOfRAAs)]
all_current = [float('nan') for _ in range(numberOfRAAs)]
echem_running_status = [False for _ in range(numberOfRAAs)]
echem_timing = [[] for _ in range(numberOfRAAs)]
new_current_available = [False for _ in range(numberOfRAAs)]
estimated_echem_end_time = [[] for _ in range(numberOfRAAs)]
```

```
# File names for data logging
dataLogFileName = [defaultPath + '/' + exp_name + '.txt' for exp_name in experimentName]
GPIO.setmode(GPIO.BCM)

# Set up GPIO pins for pumps
for k in range(len(pumpPins)):
    GPIO.setup(pumpPins[k], GPIO.OUT, initial=GPIO.LOW)

# Function to find Serial ports based on PID
def find_PID_serial_ports(num_RAAs):
    ser_ports = list(serial.tools.list_ports.comports())
    serialPID = [[] for _ in range(num_RAAs)]
    for k in range(len(ser_ports)):
        for k1 in range(num_RAAs):
            ser = serial.Serial(ser_ports[k][0], 9600, timeout=0)
            ser.flushInput()
            ser.flushOutput()
            cT = ser.readlines()
            string = f'*00{k1 + 1}G110 \r\r'
            ser.flushInput()
            ser.flushOutput()
            ser.write(string)
            ser.flushInput()
            ser.flushOutput()
            time.sleep(.1)
            cT = ser.readlines()
            ser.flushInput()
            ser.flushOutput()
            ser.close()
            if cT:
                serialPID[k1] = ser_ports[k][0]
    return serialPID

# Function to find Arduino Serial ports
def find_arduino_serial_ports(num_RAAs):
    Ard_Address = [[] for _ in range(num_RAAs)]
    ports = list(serial.tools.list_ports.comports())
    print('Looking for ARD port...')
    for p in ports:
        try:
            dev = Potentiostat(p[0])
            cT = dev.get_device_id()
            if float(cT) == 0:
                print(f'Connecting RAA 1 to Arduino (ID: {cT}) via port {p[0]}.')
                Ard_Address[0] = dev
        except:
            pass
    for i in range(num_RAAs):
```

```
        if not Ard_Address[i]:
            print(f'Could not connect RAA {i + 1} to potentiostat')
    return Ard_Address

# Function to write data to RAA file
def Write_To_RAA_File(log_file_name, new_line):
    current_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S.%f')[:-3]
    new_line = [current_time] + [str(val) for val in new_line]
    new_line = '\t'.join(new_line) + '\n'
    with open(log_file_name, 'a') as RAAFile:
        RAAFile.write(new_line)

# Function to add value to data list
def Add_Value_To_Data_List(old_vals, new_val, len_limit):
    if len(old_vals) >= len_limit:
        old_vals = old_vals[1:]
    old_vals.append(new_val)
    return old_vals

# Function to run experiment in synchronized mode
def run_rodeo_synchronized(device_ID, echem_timer, echem_parameters):
    # ... (skipped for brevity)

# Function to initialize the experiment
def initialize_experiment(device_ID, pump_pin, experiment_parameters):
    # ... (skipped for brevity)

# Find Serial ports for PID and Arduino
serialPID = find_PID_serial_ports(1)  # Changed to 1 RAA
Ard_Address = find_arduino_serial_ports(1)  # Changed to 1 RAA

# Check if connections are successful
if not serialPID[0]:
    print('Could not connect to RAA 1')
    GPIO.cleanup()
    quit()

# Initialize experiment for the single RAA
pumpPin = pumpPins[0]
deviceID = Ard_Address[0]
experimentRunParameter = experimentRunParameters[0]
experimentCurrToConcFunction = experimentCurrToConcFunctions[0]
echemRunParameter = echemRunParameters[0]
experimentName_k = experimentName[0]
dataLogFileName_k = dataLogFileName[0]

experiment_parameters = {
    'curr_to_conc_function': experimentCurrToConcFunction,
```

```
148 |    'echem_parameters': echemRunParameter,
149 |    'experiment_run_parameters': experimentRunParameter,
150 |    'experiment_name': experimentName_k,
151 |    'data_log_file_name': dataLogFileName_k,
152 |    'sample_rate': sampleRate,
153 |    'data_write_rate': dataWriteRate
154 | }

155 | try:
156 |    device_ID = Potentiostat(serialPID[0])
157 |    initialize_experiment(device_ID, pumpPin, experiment_parameters)
158 | except:
159 |    print(f"Error in RAA 1: {traceback.format_exc()}")
160 |    pass

161 | finally:
162 |    GPIO.cleanup()
163 |    print('Exiting script...')
```

## S4. Process Flow for the Control Script for Calibration Between $H_2O_2$ Concentration and Current Generated at the Electrodes

S4.1 This section outlines the process flow for the control script used in the electrochemical calibration of hydrogen peroxide ($H_2O_2$) concentration. The flowchart provided herein details the sequential steps executed by the script, from initializing the electrochemical system to the final data analysis and verification stages. It serves as a guide for replicating the calibration procedure within the specified test method and is intended to assist users in understanding the software-driven operations that underpin the electrochemical measurements. Each step in the flowchart corresponds to a critical function of the control script, ensuring that the system operates accurately and efficiently to maintain the integrity of the calibration process. The script's primary functions include:

- *S4.1.1 Initialization and Connection*: Initiates communication with and connects to the electrochemical device, identifying the device using its unique identifier.
- *S4.1.2 Parameter Configuration:* Defines the parameters for the electrochemical experiment, including voltage protocol and timing for data collection.
- *S4.1.3 Voltage Application and Data Collection:* Applies the predetermined voltage protocol, while simultaneously recording the current response.
- *S4.1.4 Data Processing and Analysis:* Processes the recorded data to calculate current.
- *S4.1.5 Decision Point:* The script reaches a critical decision point where it assesses whether additional data are required. This determination is based on user-defined criteria, which could include the total number of samples collected, the consistency of the data, or the attainment of a certain level of statistical significance.
- *S4.1.6 Experiment Conclusion:* If the script determines that no further data collection is necessary, it proceeds to safely conclude the experiment, ensuring the system's operation is properly terminated.

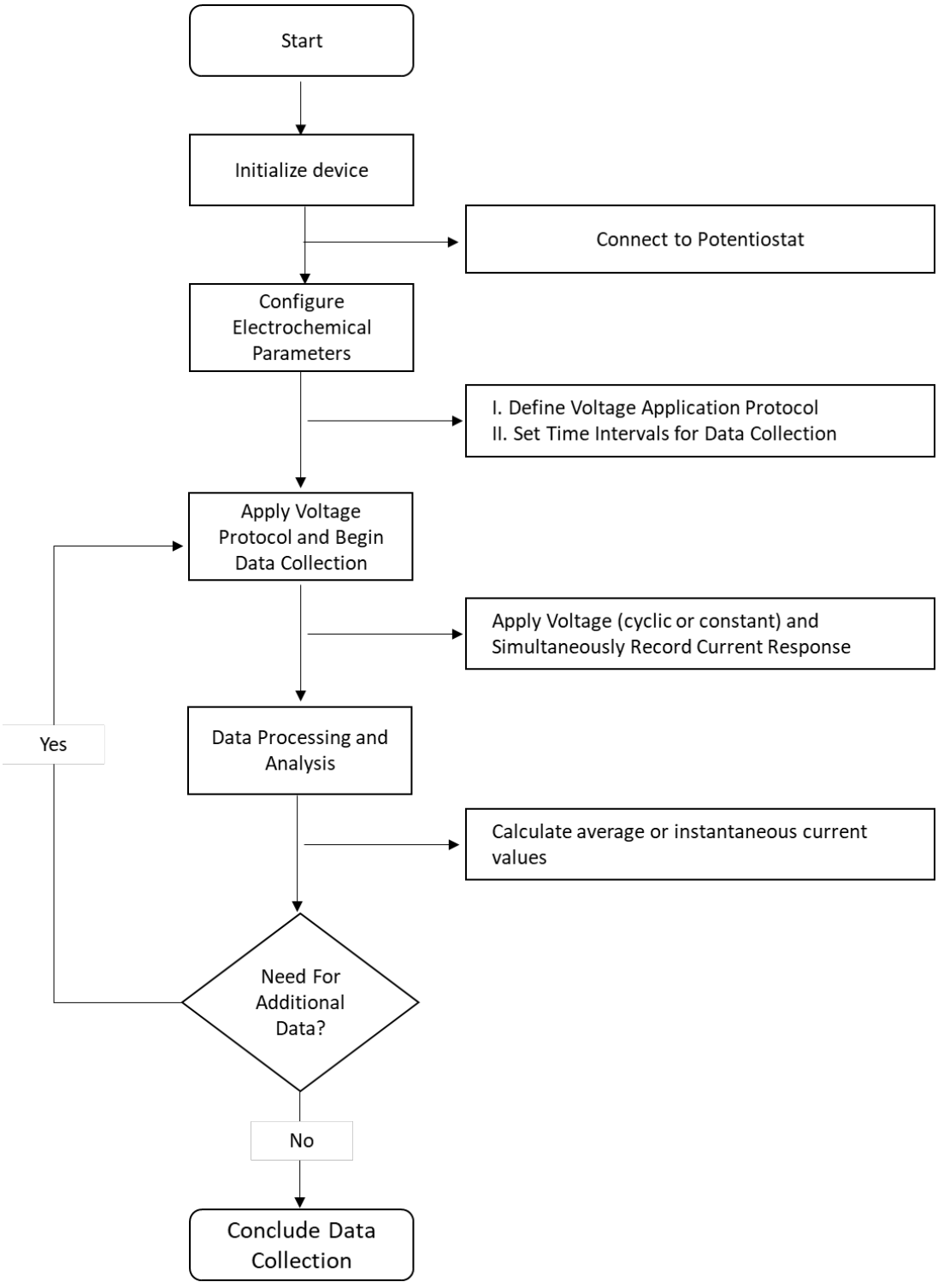


**Figure S1 – Flowchart for the control script for calibration between $H_2O_2$ concentration and current generated at the electrodes.**

## S5. Process Flow for the Control Script for Maintaining and Monitoring $H_2O_2$ Concentration

S5.1 This appendix outlines the process flow for the control script that is responsible for maintaining the concentration of $H_2O_2$ during electrochemical experiments. This script integrates with the hardware setup to regulate the $H_2O_2$ concentration by managing the peristaltic pump operations based on real-time data analysis. The flowchart illustrates the sequence of operations from the initialization of the control system, through the application of electrochemical parameters, to the activation of corrective measures when $H_2O_2$ levels deviate from the target range. Key elements of the control script include:

- *S5.1.1 Initialization and Connection*: Initiates communication with and connects to the electrochemical device, identifying the device using its unique identifier. This step also involves configuring the activation of the pump by the computing system.
- *S5.1.2 Parameter Configuration:* Inputs calibration constants and experimental parameters such as target $H_2O_2$ concentration and voltage application protocols.
- *S5.1.3 Voltage Application and Data Collection:* Applies the predetermined voltage protocol, while simultaneously recording the current response.
- *S5.1.4 Data Analysis and Pump Activation:* Processes the recorded data to calculate $H_2O_2$ concentration and activates the peristaltic pump if the concentration is lower than the target $H_2O_2$ concentration.
- *S5.1.5 Monitoring and Adjustment:* Continuously monitors the system, executing the control loop that includes data collection, analysis, and pump activation as needed, based on the real-time data.
- *S5.1.6 Experiment Conclusion:* Concludes data collection and pump operation based on user-defined criteria tailored to the experimental objectives, such as a specified duration or achieving a steady-state concentration.

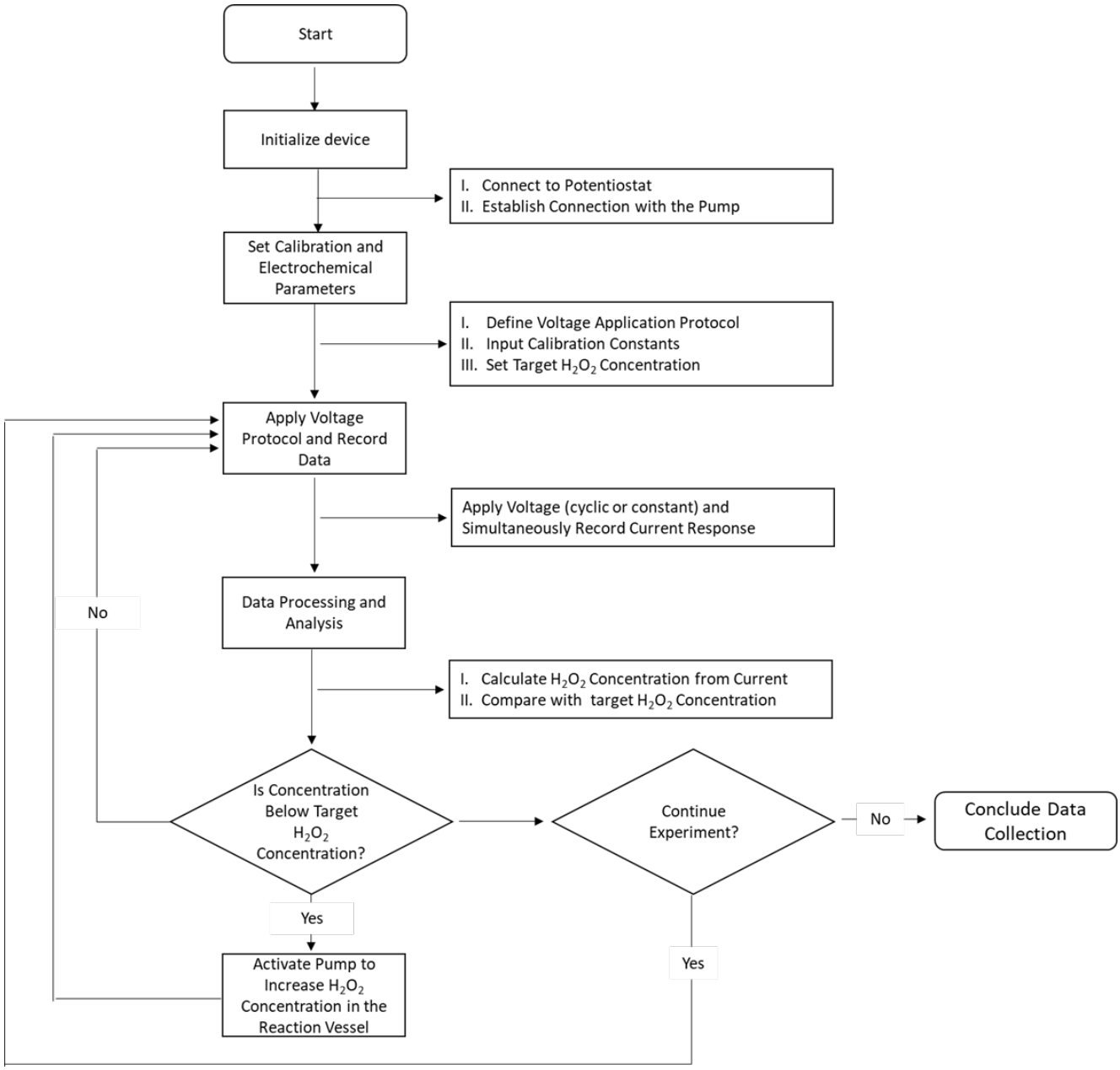


**Figure S2 – Flowchart for the control script for maintaining and monitoring $H_2O_2$ concentration.**