Photologic Experiment Rig

Control Software User Guide

Blake Hourigan

March 30, 2025

Contents

1 Introduction

This guide provides simple overviews of included features in the Photologic-Experiment-Rig control software. It also explains the specifics of these features and how to use them. Some features may not have robust explanations within the software as to what *type/unit* of value goes in a text box, what a certain value displayed in a menu means, among other various confusions.

This guide is intended to make up for areas of the program that may lack in clarity. Each primary user interface window has its own section here that you can jump to using the table of contents.

2 Installing the Software

An installation guide for all software required to get the rig up and running has been written in another article. For convenience this is provided in the pages that follow. Because the guide included is a PDF in itself, the page numbers will appear out of order until the software guide is complete.

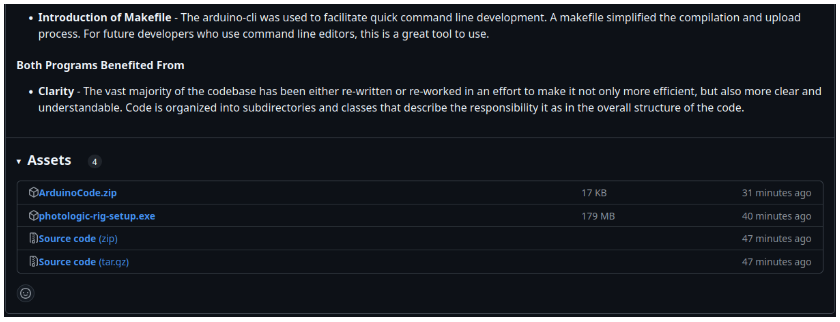
2.2 Software Migration Steps

Now we have finished the hardware migration portion of the migration! All that is left is installing and uploading the updated software packages! This is a two part process, and should be very quick and painless.

2.2.1 Installing the New Rig Control Software

1. **Download the Installer** - First, we will install the new Photologic-Experiment-Rig control software. This is the program that you interact with on the PC when running experiments.

   For this version of the software, a installer program has been created that will do all of the hard and confusing work for you. This installer program can be found on the release page for this version. Scroll to the bottom and click on the file named 'photologic-rig-setup.exe'. You can see what this download section will look like in Figure 2. You should notice the installer appear in your active downloads.



- **Introduction of Makefile** - The arduino-cli was used to facilitate quick command line development. A makefile simplified the compilation and upload process. For future developers who use command line editors, this is a great tool to use.

Both Programs Benefited From

- **Clarity** - The vast majority of the codebase has been either re-written or re-worked in an effort to make it not only more efficient, but also more clear and understandable. Code is organized into subdirectories and classes that describe the responsibility it as in the overall structure of the code.

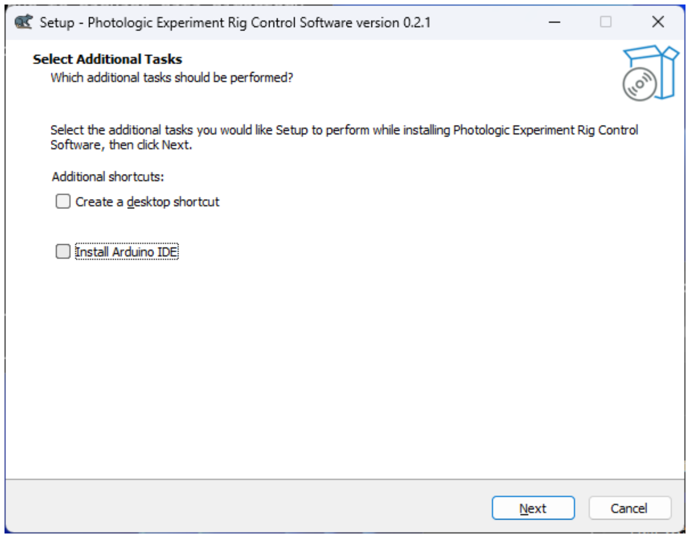
Assets 4

ArduinoCode.zip	17 KB	31 minutes ago
photologic-rig-setup.exe	179 MB	40 minutes ago
Source code (zip)		47 minutes ago
Source code (tar.gz)		47 minutes ago

Figure 2: Download the Installer Package Under the Assets Heading on the Github Release Page.

2. **Open the Installer -** To begin the installation of the program, open the '.exe' file just downloaded. It will look like the image shown in Figures 3 and 4. If you see a message saying 'Windows Protected Your PC', press the text that says 'More Info', then click the 'Run Anyway button.

   You will need to make a selection here. If you **DO** currently have the Arduino IDE installed, you can leave this unchecked. If you **DO NOT** have this software installed, you need to check the box and install the software. If you are unsure, check the box to be safe.


Setup - Photologic Experiment Rig Control Software version 0.2.1

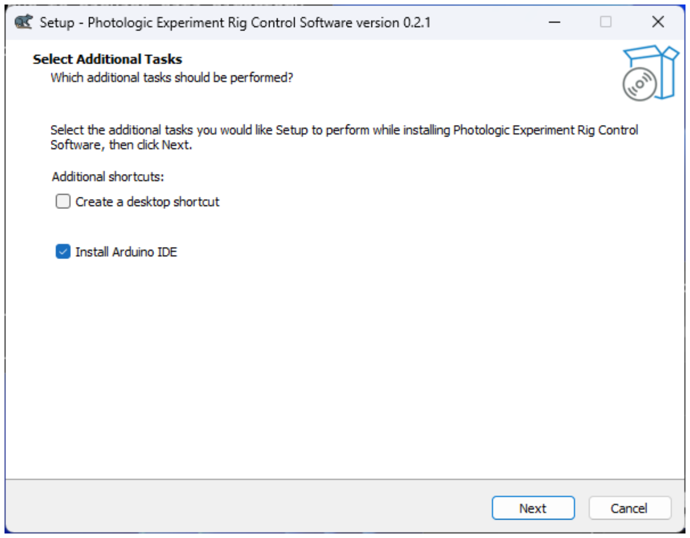
Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Photologic Experiment Rig Control Software, then click Next.

Additional shortcuts:

- [ ] Create a desktop shortcut
- [ ] Install Arduino IDE

Figure 3: Installing only the Control Software.



Setup - Photologic Experiment Rig Control Software version 0.2.1

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Photologic Experiment Rig Control Software, then click Next.

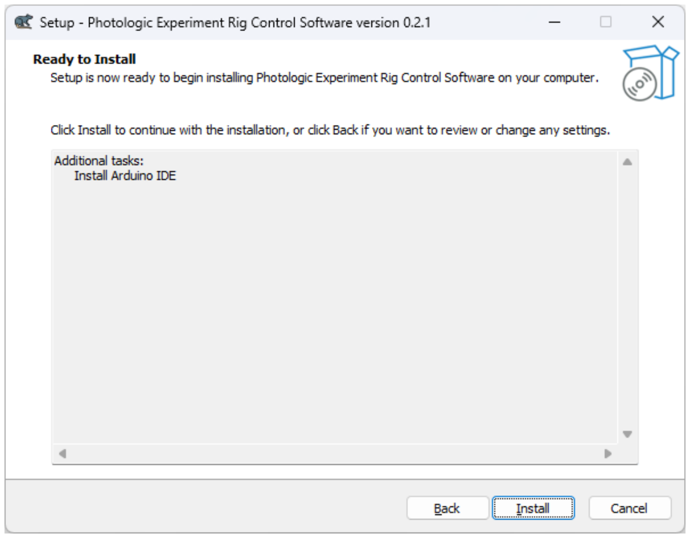
Additional shortcuts:

☐ Create a desktop shortcut

☑ Install Arduino IDE

Figure 4: Installing the Control Software with the Arduino IDE.

3. **Click Next and Verify** - Click next, then verify the additional packages to be installed. If you checked the box, the Arduino IDE installer should be listed here as in Figure 5. If not this will just be an empty page.



Setup - Photologic Experiment Rig Control Software version 0.2.1

Ready to Install

Setup is now ready to begin installing Photologic Experiment Rig Control Software on your computer.

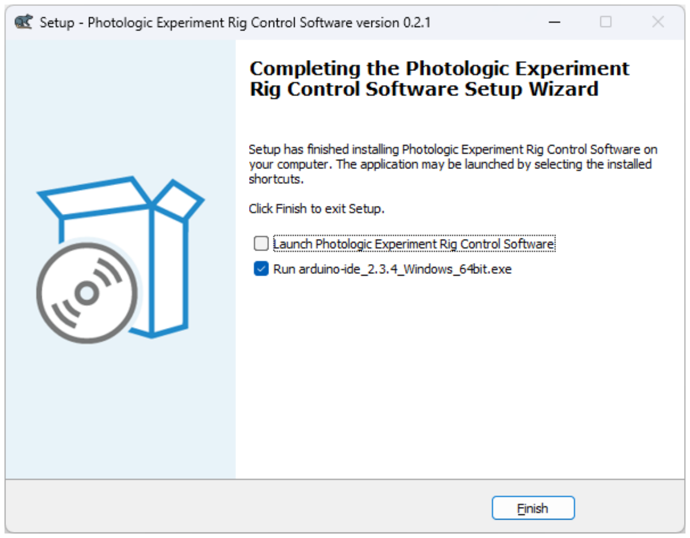
Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Additional tasks:
    Install Arduino IDE
```

Figure 5: Confirm the Packages and Perform Installation.

4. **Perform the Installation** - Click install to perform the intallation.
5. **Run the Arduino Installer** - When the installer has finished, if you

have selected to install the Arduino IDE you will see a screen like that in Figure 6. Make sure you have ‘Launch Photologic Experiment Rig Control Software’ **unchecked** and run the Arduino IDE setup. Inside of the Arduino installer, you can procede through the installer without changing any defaults. Once the installation is finished, run the Arduino IDE.


Setup - Photologic Experiment Rig Control Software version 0.2.1

Completing the Photologic Experiment Rig Control Software Setup Wizard

Setup has finished installing Photologic Experiment Rig Control Software on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☐ Launch Photologic Experiment Rig Control Software

☑ Run arduino-ide_2.3.4_Windows_64bit.exe

Figure 6: Do NOT Run the Control Software, but DO Run the Arduino Installer.

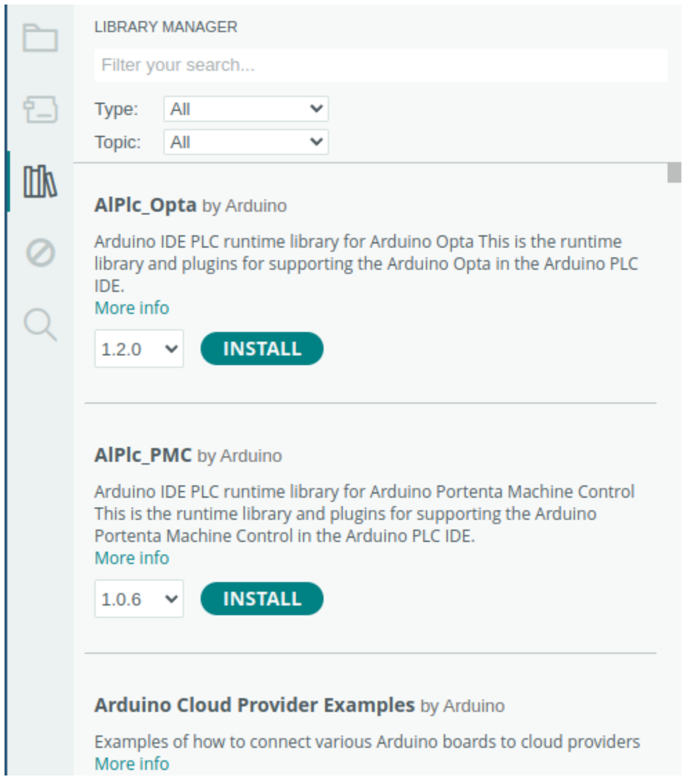
2.2.2 Installing the Arduino Software Update

Now that you have installed the control software and have the Arduino IDE installed and open on your computer, we will download and install the update to the Arduino board.

1. **Prepare for File Upload** - To upload files to the Arduino board, you will need some ‘drivers’ so that the computer understands how to communicate information to the Arduino. Open the Arduino IDE, connect your Arduino board via its USB cable, and wait for a moment. If you have not opened the IDE before, it will automatically detect that you do not have the packages that you need and will prompt you to install them. If these prompts appear, click **install** on each one of them.

2. **Install Dependencies** - There is one more file that we need to be able to install the update to the board. This is the ‘AccelStepper’ community Arduino library. To install this package, navigate to the ‘Library Manager’

by clicking the third icon down in the left-hand menu in the IDE. This button and window is shown in Figure 7.



AlPlc_Opta by Arduino

Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins for supporting the Arduino Opta in the Arduino PLC IDE.
More info

1.2.0 INSTALL

AlPlc_PMC by Arduino

Arduino IDE PLC runtime library for Arduino Portenta Machine Control This is the runtime library and plugins for supporting the Arduino Portenta Machine Control in the Arduino PLC IDE.
More info

1.0.6 INSTALL

Arduino Cloud Provider Examples by Arduino

Examples of how to connect various Arduino boards to cloud providers
More info

Figure 7: Open the Library Manager to Install Dependencies.

Click the search bar of this window, and type ‘AccelStepper’. You will see an entry at the top as shown in Figure 8. In the drop-down with numbers in it, left of the install button, ensure version number 1.64 is selected. Now click install. This should only take a moment. If the install is successful you should see a message ‘Installed AccelStepper@1.64.0’ in the black terminal window inside the IDE.

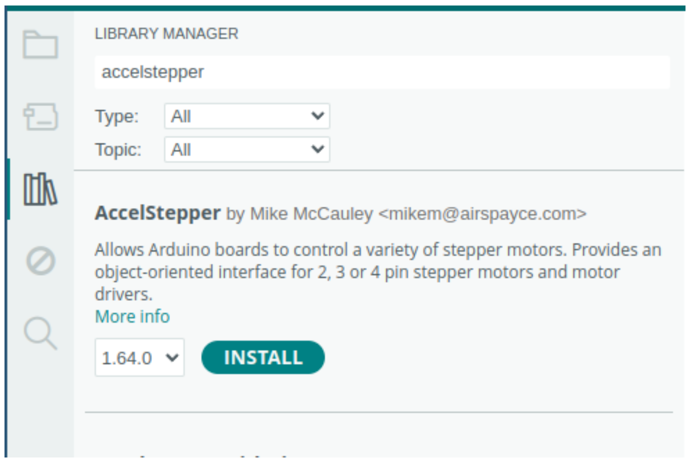
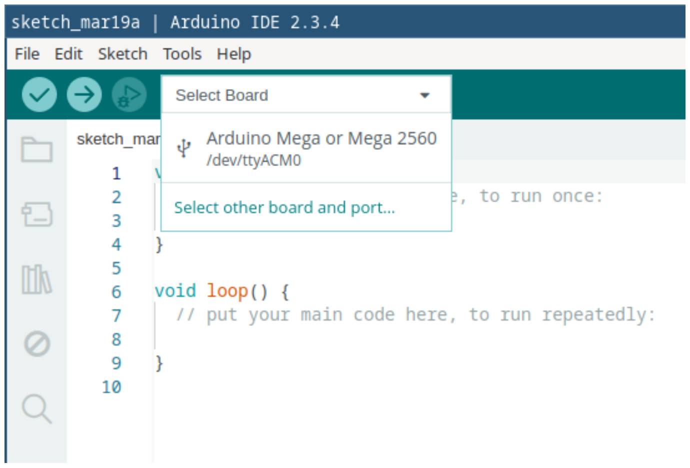

LIBRARY MANAGER
accelstepper
Type: All
Topic: All
AccelStepper by Mike McCauley <mikem@airspayce.com>
Allows Arduino boards to control a variety of stepper motors. Provides an object-oriented interface for 2, 3 or 4 pin stepper motors and motor drivers.
More info
1.64.0
INSTALL

Figure 8: Install Version 1.64 of the AccelStepper Library.

3. **Select Your Arduino Board** - If it is not already, plug your Arduino Mega 2560 board into the computer via its USB cable. Once plugged in, click on the drop-down towards the top of your screen titled 'Select Board' as shown in Figure 9. Here you should see your board listed. Click on it to select it. Now we are ready to upload code to our board.


sketch_mar19a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Select Board
Arduino Mega or Mega 2560
/dev/ttyACM0
Select other board and port...
sketch_mar

```
    , to run once:
}

void loop() {
  // put your main code here, to run repeatedly:

}
```

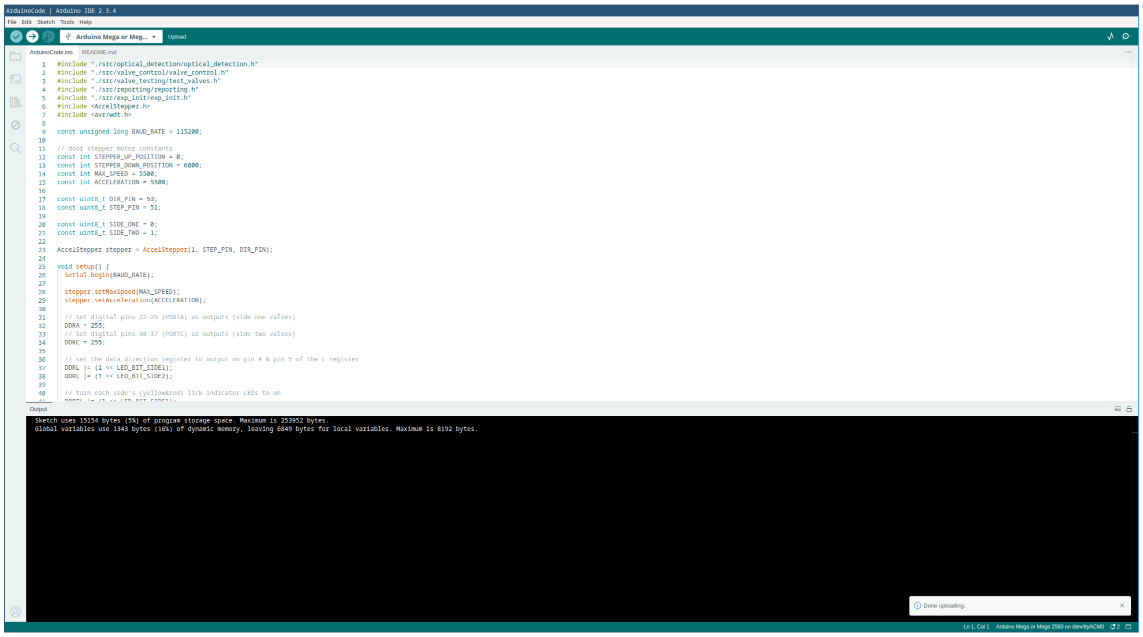
Figure 9: Select Your Arduino in the 'Select Board' Menu.

4. **Download the Update** - If you did not do so earlier, download the ArduinoCode.zip file to your computer from the Github release page. This file is listed in the release page 'Assets' downloads sections as shown in Figure 2.

5. **Unzip the Compressed Folder** - Navigate to the folder where you downloaded this file and right click on it. Somewhere in this menu, there will be an option of the form 'Uncompress Archive', 'Unzip archive', 'Extract All', or something very similar. Click on this option to produce a regular folder containing the file contents.

6. **Open the Update in Arduino IDE** - Navigate back to the open Arduino IDE window, and open the update file by selecting the 'file' menu in the top left hand corner, then 'open'. Here, navigate to the ArduinoCode folder that you just decompressed, and select the file named 'ArduinoCode.ino'. You will see the file content appear in the editor window in the center of your screen.

7. **Upload the Update to the Board** - In the green top bar, hit the middle (second) button with a right arrow in it. This will upload the update to your board. If the update succeeds, you will see a small 'Done Uploading.' message in the bottom right hand side of your screen. This is shown in Figure 10.


ArduinoCode | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Arduino Mega or Meg...
Upload
ArduinoCode.ino
README.md
Output
Done uploading.
Ln 1, Col 1
Arduino Mega or Mega 2560 on /dev/ttyACM0
2

```
#include "./src/optical_detection/optical_detection.h"
#include "./src/valve_control/valve_control.h"
#include "./src/valve_testing/test_valves.h"
#include "./src/reporting/reporting.h"
#include "./src/exp_init/exp_init.h"
#include <AccelStepper.h>
#include <avr/wdt.h>

const unsigned long BAUD_RATE = 115200;

// door stepper motor constants
const int STEPPER_UP_POSITION = 0;
const int STEPPER_DOWN_POSITION = 6000;
const int MAX_SPEED = 5500;
const int ACCELERATION = 5500;

const uint8_t DIR_PIN = 53;
const uint8_t STEP_PIN = 51;

const uint8_t SIDE_ONE = 0;
const uint8_t SIDE_TWO = 1;

AccelStepper stepper = AccelStepper(1, STEP_PIN, DIR_PIN);

void setup() {
  Serial.begin(BAUD_RATE);

  stepper.setMaxSpeed(MAX_SPEED);
  stepper.setAcceleration(ACCELERATION);

  // Set digital pins 22-29 (PORTA) as outputs (side one valves)
  DDRA = 255;
  // Set digital pins 30-37 (PORTC) as outputs (side two valves)
  DDRC = 255;

  // set the data direction register to output on pin 4 & pin 5 of the L register
  DDRL |= (1 << LED_BIT_SIDE1);
  DDRL |= (1 << LED_BIT_SIDE2);

  // turn each side's (yellow&red) lick indicator LEDs to on
```

```
Sketch uses 15154 bytes (5%) of program storage space. Maximum is 253952 bytes.
Global variables use 1343 bytes (16%) of dynamic memory, leaving 6849 bytes for local variables. Maximum is 8192 bytes.
```

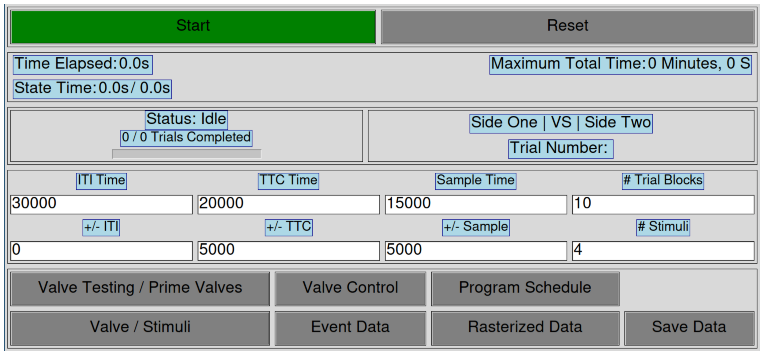
Figure 10: A Successful Upload Shows a 'Done Uploading' Notification.

That's it! You are done with the migration process from two to one Arduino boards! Now you should test the program to ensure proper functionality with your motor/valves/optical detectors. This process is detailed below in Section 3.

3 The Main Control Window

The ‘Main Control Window’ refers to the user interface window that appears first upon application startup. This window is the heart of the program and is where much of the required experiment configuration will take place. This window is shown below in Figure 1.

This window can be broken down into 5 primary sub-sections, and as such, this section of the guide is broken down into sub-sections that reflect this layout.



Time Elapsed: 0.0s

Maximum Total Time: 0 Minutes, 0 S

State Time: 0.0s / 0.0s

Status: Idle

0 / 0 Trials Completed

Side One | VS | Side Two

Trial Number:

ITI Time	TTC Time	Sample Time	# Trial Blocks
30000	20000	15000	10
+/- ITI	+/- TTC	+/- Sample	# Stimuli
0	5000	5000	4

Figure 1: The ‘Main Control Window’ is the root of all application logic and sub-windows. It allows for navigation to these sub-windows, and allows users to configure and begin experiments.

3.1 Row 1 - Start and Reset

Row 1 contains two of the most important buttons in the application interface. Start/Stop allows a user to command the program to begin or terminate an experiment depending on current state. Reset allows the user to clear **all existing experiment data** and ready the application for a new experiment run.

3.1.1 Start and Stop (an Experiment)

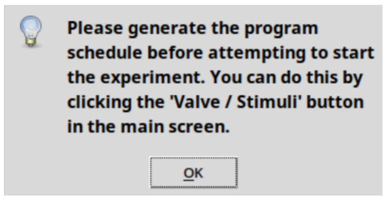
The start button is important because it allows the user to begin or end an experiment. Before the user can begin an experiment, the experiment must be fully configured. This means that all of the following steps must be completed to continue:

1. Base state interval times (ITI, TTC, Sample, etc.), random interval times (+/- ITI, +/- TTC, +/- Sample, etc.) and other configuration variables (# Trial Blocks, # Stimuli) **must** be defined. Discussed in Section 3.4.
2. All valves must be configured with corresponding stimulus names in the 'Valve / Stimuli' window. Discussed in depth in Section 7.
3. The program schedule must be generated. Discussed in depth in Section 7.

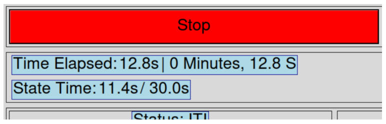
Once all of these requirements have been fulfilled the user can begin an experiment run. If a step has been missed, or something has gone wrong, the user may see a warning message such as the one in Figure 2a. This indicates that the program stimuli schedule has not yet been generated, and that the user should re-visit the 'Valve / Stimuli' window (Section 7) to attempt a proper schedule generation.

If a experiment run has been started, the 'Start' button will be changed to a stop button as shown in Figure 2b. Pressing this button will terminate the experiment early. This will begin the standard termination process. This includes:

- Halting program execution. No further experiment operations will be taken. Door will not close, program state will be switched to "IDLE".
- Terminating the connection to the Arudino board .
- Preparing all accumulated data to be saved.
- The user will be prompted to optionally save this data via a pop-up window in a location of their chosing.


Please generate the program schedule before attempting to start the experiment. You can do this by clicking the 'Valve / Stimuli' button in the main screen.

(a) Without an Experiment Schedule, the program isn't sure what to do. You must generate a schedule following the steps above to run an experiment.


Stop
Time Elapsed: 12.8s | 0 Minutes, 12.8 S
State Time: 11.4s / 30.0s

(b) Start Button is Converted to Stop Button on Experiment Start.

Figure 2

3.1.2 Reset (Preparing for a New Experiment)

The other primary button in this row of the main window is the 'Reset' button. This button is important because it is **required** after running an experiment that the user takes either of the two following actions:

1. Presses the gray 'Reset' button in the main program window.
2. Use the familiar red Windows 'X' in the corner of the main application window to close the application session and manually restart by clicking the 'Photologic-Experiment-Rig' control software icon on the desktop or in the taskbar. This is the icon with a cartoon rat on it.

Both of these actions perform the same effective operation which is to clear the data from the previous experiment and get the program ready for a new experiment. To reiterate, pressing 'Reset' will **destroy all experiment data accumulated** if the user has not saved this data by pressing the 'Save Data' button in the fifth row of the main experiment window since the last experiment was run. This means the user should be **very careful** when pressing this button.

It is for this reason that when a user presses the Reset button, a warning will be displayed as shown in Figure 3. If the user selects 'Yes' in this popup window all data will be destroyed and all windows will be closed, then the main program window will re-open automatically, ready for a new experiment.

NOTE: This action cannot be performed unless an experiment is *not currently running*. Otherwise, you must stop the current experiment and then reset the application.

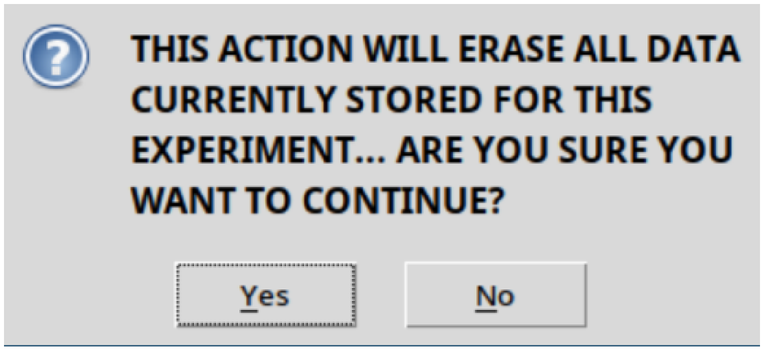

THIS ACTION WILL ERASE ALL DATA CURRENTLY STORED FOR THIS EXPERIMENT... ARE YOU SURE YOU WANT TO CONTINUE?

Figure 3: User must Confirm a Reset Action before it will Execute to Prevent Accidental Loss of Data.

3.2 Row 2 - Program Timers

The second row of the main program window features three timers that inform the user of the following pieces of information:

1. **Time Elapsed**: How long the program has been running in total
2. **State Time**: How long the program has been in the current state (ITI, TTC, DOOR OPEN, etc.).
3. **Maximum Total Time**: Maximum amount of time the program can possibly run.

These timers all begin at zero until the program schedule has been generated (discussed in Section 7) and the user presses the 'Start' button in the main experiment window. Once an experiment has begun, this row will be populated to reflect the real values that belong here as shown in Figure 4.

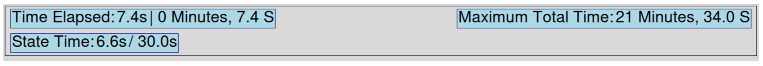

Time Elapsed: 7.4s | 0 Minutes, 7.4 S
Maximum Total Time: 21 Minutes, 34.0 S
State Time: 6.6s / 30.0s

Figure 4

3.2.1 Time Elapsed

This timer is the primary timer for the currently running experiment. This timer shows the time elapsed since the moment the user pressed the start button. It follows a format of ‘SS | MM:SS’, where the ‘SS’ before the format separator (‘|’) is the *total seconds* elapsed since program start time.

Because this is a bit unwieldy and perhaps unhelpful, the information following the separator ‘|’ shows the same figure converted to minutes and seconds. This allows for an easier understanding at a glance of total experiment time elapsed.

3.2.2 State Time

The ‘State Time’ timer keeps track not only of the amount of state time that has elapsed since the state began, but also the maximum time it could possible consume. The timer follows the format of ‘ELAPSED TIME / TOTAL AVAILABLE TIME’.

The only time throughout an experiment when the ‘TOTAL TIME’ figure *may* not be met is during the ‘TTC’ time. This time can be cut short if the rat chooses to engage in the trial. Thus the state timer may end before reaching ‘TOTAL TIME’.

3.2.3 Maximum Total Time

The ‘Maximum Total Time’ timer is included as a convenience function that gives the user a general idea of how long an experiment could possibly take. It is almost certain to be inaccurate and *over-estimate* the amount of time an experiment will take, because it calculates this value by *summing all base interval values and random interval values together.*

This means that the program assumes that all intervals — including TTC intervals — will use **all** of their allocated time. This is very unlikely to be true, as the rat is likely to engage in at least *some* trials. However, it still gives a general idea of the length of time an experiment may take.

3.3 Row 3 - Status Indicators

Row 3 contains many different experiment *status indicators* that inform the user of the current values of important trial variables. These include:

- **Status (Program State)**: This indicator informs the user of the current program state ("IDLE", "ITI", "TTC", etc.)
- **Trials Completed (Current Trial/Total Trials)**: Displays the current trial and total trials for this experiment. Includes a progress indicator to show experiment progress at a glance.
- **Side One: {Stimulus} |VS| Side Two: {Stimulus}**: Current Stimulus Delivery per Port (Side): Inform user which stimulus is being delivered on each port (side) for the current trial.

Examples of what these indicators look like during an experiment run are shown in Figure 5.

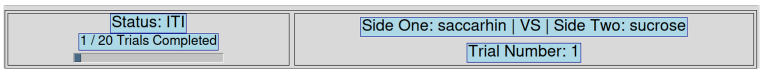
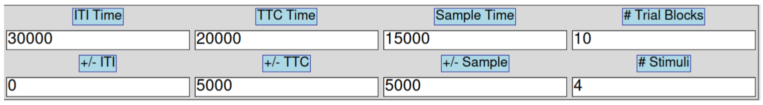

Status: ITI
1 / 20 Trials Completed
Side One: saccarhin | VS | Side Two: sucrose
Trial Number: 1

Figure 5: Status Indicators Aid the User by Informing them of the Current State of Experiment Variables.

3.4 Row 4 - Configuring the Experiment

Row 4 is one of the rows in the main window that the user will interact with the most. This row is where the user enters any desired configuration changes for an upcoming experiment. This row and the contained entry widgets are shown in Figure 6.

The primary experiment variables that may change from experiment to experiment are the base state intervals (ITI, TTC, Sample), random state intervals (+/- ITI, +/-, TTC, etc), '# Trial Blocks' (number of times each possible stimulus pairing should be delivered), and '# Stimuli' (number of valves in use).



ITI Time	TTC Time	Sample Time	# Trial Blocks
30000	20000	15000	10
+/- ITI	+/- TTC	+/- Sample	# Stimuli
0	5000	5000	4

Figure 6: The Entry Widgets in the Main Window Allow the User to Modify Default Experiment Variables.

These variables should be changed **before** the schedule is generated for the experiment.

NOTE: All *time-related values* (interval times) must be given in **milliseconds**. For example, 30 seconds should be entered in as 30000 milliseconds.

Ensure that no entry boxes are left *empty* before generating the schedule. Each box needs to contain some *positive integer* to correctly generate the experiment schedule.

3.4.1 Changing Default Values

If the default values need to be modified, this requires a change in the Python codebase. This is an involved process because new versions of the program need to be compiled and built into a new installer.

Default values are defined in the GitHub repository directory in the file found at subdirectory '/Code/Photologic Rig Main Application Code/src/models/experiment_process_data.py' under variables named self.interval_vars, and self.exp_var_entries.

This code is shown in Listing 1. Defaults can be changed here, but again a new program build and installer must be created and installed onto the user's machine. Users may need to ask the lab tech for assistance in this process.

```python
self.interval_vars: dict[str, int] = {
    "ITI_var": 30000,
    "TTC_var": 20000,
    "sample_var": 15000,
    "ITI_random_entry": 0,
    "TTC_random_entry": 5000,
    "sample_random_entry": 5000,
}
self.exp_var_entries: dict[str, int] = {
    "Num Trial Blocks": 10,
    "Num Stimuli": 4,
    "Num Trials": 0,
}
```

Listing 1: Default values can be seen Above. Time-Related Interval Values are Separated from Other Values like Number of Trials.

3.5 Row 5 - Accessing Other Control Windows

Row 5 of the main program window contains 7 gray buttons that allow for navigation to other important experiment control windows. (Figure 7)

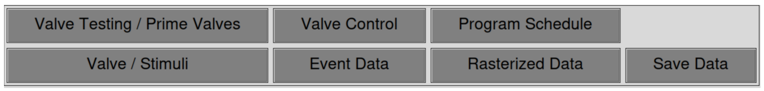


Figure 7: The Buttons in Row 5 Allow for Navigation to Other Experiment Windows.

Descriptions of buttons and their functions will begin on the first row and column of this section. Each button in a row will be described, then the next row will be described.

1. **Valve Testing / Prime Valves**: This button allows for navigation to the 'Valve Testing / Prime Valves' window. A user would navigate here if they desired calibrating valve opening durations via the automatic testing procedure or if they desired to 'Prime' the valves prior to an experimnet run. This window is discussed in depth in Section 4.

2. **Valve Control**: This button allows for navigation to the 'Valve Control' window. A user would navigate here if they desired opening or closing valves for arbitrary lengths of time. This window also allows for the manual toggling of the rig's door motor to the up or down position for arbitrary lengths of time. This window is discussed in depth in Section 5.

3. **Program Schedule**: This button allows for navigation to the 'Program Schedule' window. This is where the user would find most experiment data including:

    - Trial block numbers.
    - Trial numbers.
    - Licks-per-port for a trial.
    - Stimulus delivered on each side for a trial.
    - ITI, TTC, Sample times for a trial.

    This window is discussed in depth in Section 6.

4. **Valve / Stimuli**: This button allows for navigation to the 'Valve / Stimuli' window. This is where the user navigates to assign stimuli names to specific valves for the experiment. This is required to generate the experiment schedule. This window and associated functions are discussed in depth in Section 7.

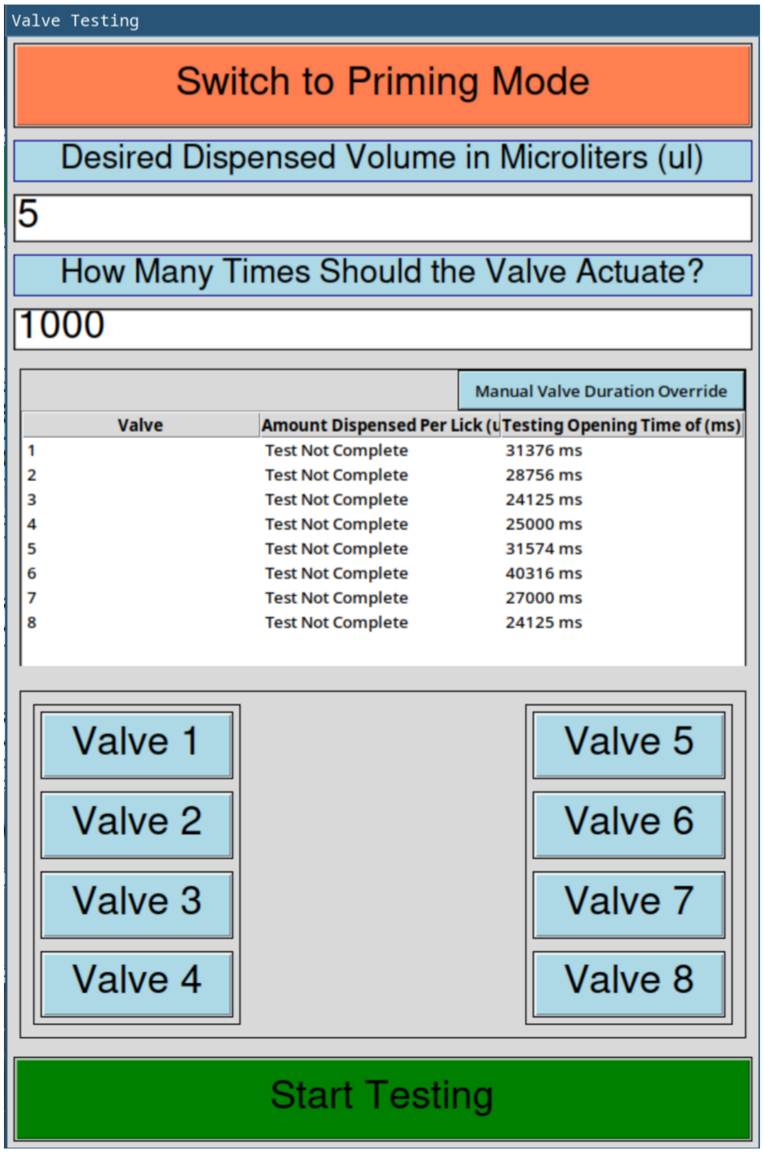
5. **Event Data**: This button allows for navigation to the 'Event Data' window. This is where all data not included in the 'Program Schedule' window is found. A user might navigate here to view detailed data about specific events that take place during the program like licks or door motor movements. This window is discussed in depth in Section 8.

6. **Rasterized Data**: This button allows for navigation to the 'Rasterized Data' window. A user might navigate here to view lick data per trial in a raster plot. This window is discussed in depth in Section 9.

7. **Save Data**: This button is used to save all currently held data into 2 separate excel spreadsheet files. Data in the program is split into 2 primary 'DataFrames'. These DataFrames include the 'Program Schedule DataFrame' and the 'Event Dataframe'. The data stored in each frame is as described previously for each respective window corresponding to these DataFrames.

4 'Valve Testing / Prime Valves' Window

This window is where the user would navigate for valve opening duration calibration or valve 'priming' before a experiment run. The 'Valve Testing / Prime Valves' window is shown below in Figure 8.


Valve Testing
Switch to Priming Mode
Desired Dispensed Volume in Microliters (ul)
5
How Many Times Should the Valve Actuate?
1000
Manual Valve Duration Override
Valve 1
Valve 2
Valve 3
Valve 4
Valve 5
Valve 6
Valve 7
Valve 8
Start Testing

Valve	Amount Dispensed Per Lick (u	Testing Opening Time of (ms)
1	Test Not Complete	31376 ms
2	Test Not Complete	28756 ms
3	Test Not Complete	24125 ms
4	Test Not Complete	25000 ms
5	Test Not Complete	31574 ms
6	Test Not Complete	40316 ms
7	Test Not Complete	27000 ms
8	Test Not Complete	24125 ms

Figure 8: The 'Valve Testing / Prime Valves' Window opens in Testing Mode upon First Button Press.

4.1 Valve Open Duration Calibration (Running a Valve Test)

This window opens in ‘Testing’ mode by default. A user can be sure that they are in test mode if the large orange ‘Switch to Priming Mode’ button is in view at the top of the window.

In this window a user will notice several interface components that are critical to a successful valve calibration. Calibration is important to run often to ensure that each valve is delivering the proper amount of stimulus on each valve in use during an experiment. The process of running a valve test procedure is described below.

1. **Enter Desired Amount of Stimulus Per Lick**: First, the user should enter how much liquid that is desired each time the valve opens in the entry box just below the label ‘Desired Dispensed Volume in Microliters (ul)’. The default value for this entry is 5 microliters.
2. **Enter Number of Valve Actuations**: Then, enter the number of times a valve should open and close for each test in the entry box just below the label ‘How Many Times Should the Valve Actuate?’.

   The default value for this entry is 1000 and is a good starting point for testing. This attribute of the procedure is *variable* in the event that a user wants to change this value, and new duration calculations will adjust to the value input in this field. However, higher numbers will yield more accuration calculations.
3. **Select Valves to be Tested**: Next, the user should select the valves to be tested during this procedure. This is done by selecting valves via the buttons below the valve test information table. The valve test information table is the interface widget that lists all valves along with ‘Amount Dispensed Per Lick (ul)’ and ‘Testing Opening Time (ms)’ columns. Figure 9 shows the difference between the interface with no buttons selected vs some selected.
4. **Start Testing**: From the perspective of the interface, a test can now be started. The user should ensure that the valve cylinders (vials containing stimuli for the valves) are full and all other rig specific guidelines set for the lab are now met. Once the user has ensured that the physical rig is correctly setup for a valve calibration procedure, they can press the big green ‘Start Testing’ button at the bottom of the window.

   NOTE: *Some* number of valves *above zero* must be selected, or bugs may occur. Ensure that some valves are selected before starting a test procedure.

5. **Confirm Test Settings**: Once the user has pressed the green 'Start Testing' button, they will see a new pop-up window as shown in Figure 10. Review the settings selected using the Valve Test Table, and the entry boxes to ensure the correct settings have been selected. If all setting look okay press 'Yes' to begin the procedure. Valves should be heard actuating on the rig.

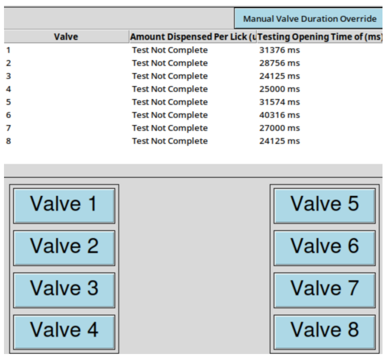
6. **Enter the Amount Dispensed**: In order to understand what the opening time for the valve should be, the actual amount dispensed over some number of trials must be found. Find this value either by wighing the item used to collect the dispensed volume or by marking the volume line on the container. Enter this value **IN MILLILITERS** in the text box pop-up that appears after a test has completed. An example pop-up is shown in Figure 11. Do this for each of the valves that have been tested during this cycle (1 or 2).

   NOTE: The test procedure has been designed to be flexible and adapt to each specific test. This means that each test may be slightly different from the last. Specifically, the user may select any number of valves to be tested between 1-8. All valves on one side can be tested and none on the other, two on each side, or any number of configurations.

   Because of this, the user must pay attention and be careful when input pop-ups appear. If there are more than zero valves remaining **on each side** to be tested, there will be **2 entry boxes** that appear at the end of the test cycle. **Pay attention** to which valve the program is asking the user to enter the dispensed volume for. This is *incredibly easy* to mess up and the test will need to be restarted if this happens.

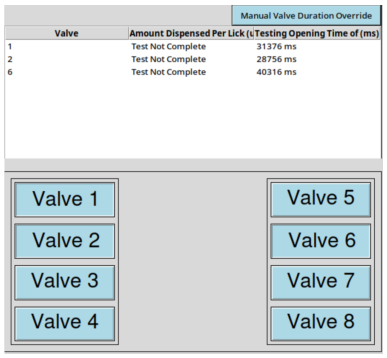
   Finally, if the user presses 'Cancel' on **ANY** of the input pop-up windows, the test procedure will be cancelled and will need to be restarted. If the red 'ABORT TESTING' button is pressed *at any time* during the test procedure, the test will be cancelled. Test settings can then be modified as needed and a test can be restarted. *No program restart* should be necessary to run multiple test procedures.

7. **Confirm Duration Changes**: Once all valves selected for a test procedure have been tested, duration changes will be *automatically* calculated. Then, a new pop-up window will appear that will display the change in open duration for each valve that was tested. An example of this pop-up window is shown in Figure 12. If these changes seem legitimate (no very large jumps or reductions in opening times) confirm them by pressing the green 'Confirm Changes' button. Otherwise, press the red 'Abort Changes' button and retry the procedure.



Valve	Amount Dispensed Per Lick (u	Testing Opening Time of (ms)
1	Test Not Complete	31376 ms
2	Test Not Complete	28756 ms
3	Test Not Complete	24125 ms
4	Test Not Complete	25000 ms
5	Test Not Complete	31574 ms
6	Test Not Complete	40316 ms
7	Test Not Complete	27000 ms
8	Test Not Complete	24125 ms

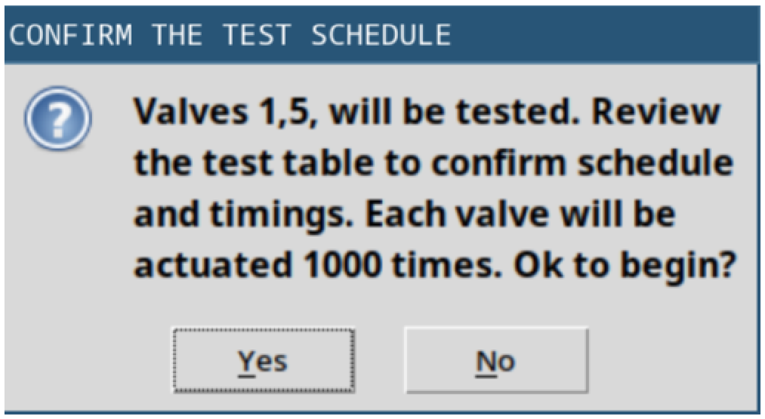
(a) Default State of the Valve Test Window. Buttons not Pressed, all Valves in View in Valve Test Table.



Valve	Amount Dispensed Per Lick (u	Testing Opening Time of (ms)
1	Test Not Complete	31376 ms
2	Test Not Complete	28756 ms
6	Test Not Complete	40316 ms

(b) Valves 1, 2, 6 Selected. Valve Test Table Shows entries *only* for *these* Valves, Corresponding Buttons Depressed.

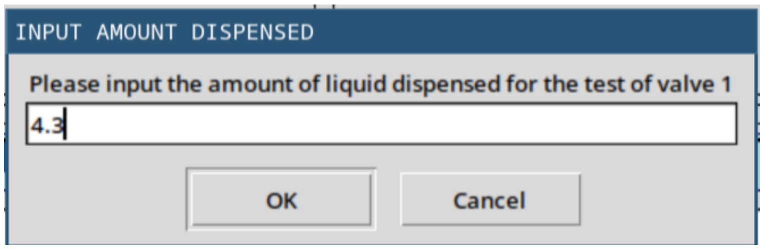
Figure 9: The User will Know a Valve has been Selected when its Corresponding Button has been Depressed (pushed in), and the Valve Test Table has been Updated with a row that Includes the Selected Valve.



CONFIRM THE TEST SCHEDULE

Valves 1,5, will be tested. Review the test table to confirm schedule and timings. Each valve will be actuated 1000 times. Ok to begin?

Figure 10: The User Must Confirm Test Settings before the Procedure will Begin.

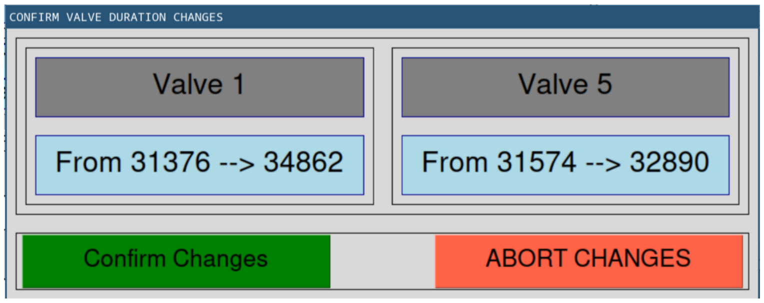


INPUT AMOUNT DISPENSED

Please input the amount of liquid dispensed for the test of valve 1

4.3

Figure 11: 1 or 2 Input Pop-Up Windows will Appear Depending on Number of Valves Remaing to Test on Each Side.



CONFIRM VALVE DURATION CHANGES

Valve 1

From 31376 --> 34862

Valve 5

From 31574 --> 32890

Figure 12: Users can Confirm or Abort Updating the Valve Durations in the ‘CONFIRM VALVE DURATION CHANGES’ Pop-Up Window following a Successful Test Procedure.

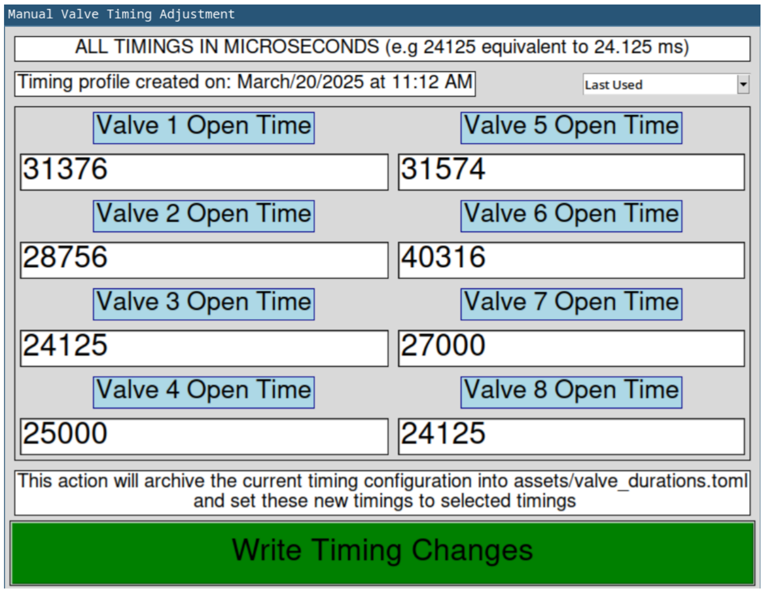
4.2 Manual Valve Duration Adjustment / Duration Profiles

Manual adjustment of valve timings or changing the selected valve timing 'profile' is available by selecting the 'Manual Valve Duration Override' button just above and to the right of the valve test information table in the 'Valve Testing' window.

4.2.1 Why was this Necessary?

Sometimes mistakes are made in the testing procedure. One might overlook an abnormally large adjustment in the automatic calculations that could cause issues with proper valve actuation during experiment runtime. Perhaps it is found that a previous duration profile seemed to work better than the current one for one reason or another.

Whatever the reason, it is nice to be able to restore things to working order with the press of a single button. This is the purpose of valve duration profiles and the manual time adjustment window. This window is pictured below in Figure 13.



ALL TIMINGS IN MICROSECONDS (e.g 24125 equivalent to 24.125 ms)

Timing profile created on: March/20/2025 at 11:12 AM

Valve 1 Open Time	Valve 5 Open Time
31376	31574
Valve 2 Open Time	Valve 6 Open Time
28756	40316
Valve 3 Open Time	Valve 7 Open Time
24125	27000
Valve 4 Open Time	Valve 8 Open Time
25000	24125

This action will archive the current timing configuration into assets/valve_durations.toml and set these new timings to selected timings

Figure 13: Users can Override Mistankenly Accepted Automatic Valve Duration Adjustments by Manually Entering New Times in the 'Manual Valve Timing Adjustment' window.

4.2.2 Okay, How do I do it? (Setting Valve Duration Profiles)

Here, users can manually enter different opening times for any of the rig's currently configured 8 valves and save them as the new selected timings by pressing 'Write Timing Changes'. However, most people likely do not have the exact timing for a given valve memorized. To aid the user here are different valve timing profiles. To find previously configured or default profiles, click the dropdown menu in the top right of this window as shown in Figure 14.

In this dropdown menu, users have the ability to set durations to one of the following profiles:

- **Default**: Reset all durations back to the default value of 24125 milliseconds by writing the default profile as the new active profile. This is a good starting point for calibration on fresh valves.
- **Last Used**: These timings are the most recently used timings and serve as the active timings for the program. At all times, this profile is the one used during experiments. When new timings are written, they are written into *this* profile.
- **Archive 1**: Archives are used as 'back-ups' of older profiles. Anytime a new profile is written (automatic from a valve-test or manually) the previous 'profile' is set to the newest slot in the archive. The newly created profile is then set as the 'Last Used' or current profile.

  Each time a new archive is added, each archive moves down a slot. For example, if a user writes a new timing profile, the oldest archive profile ('Archive 3') is destroyed. Then the contents of archive 2 move to archive 3 ('Archive 2' -> 'Archive 3'). Archive 1 then follows the pattern, moving to slot 2 ('Archive 1' -> 'Archive 2'). This clears the way for the last used profile to be moved to the most recent arhive slot ('Last Used' -> 'Archive 1'). Finally, the newly created profile can be set as the active profile ('Last used').
- **Archive 2**: Archive 2 is the next archival slot, the second oldest of the archive slots.
- **Archive 3**: Archive 3 is the final archive slot, the oldest of all archives, and is the next profile that will be destroyed upon a new write.

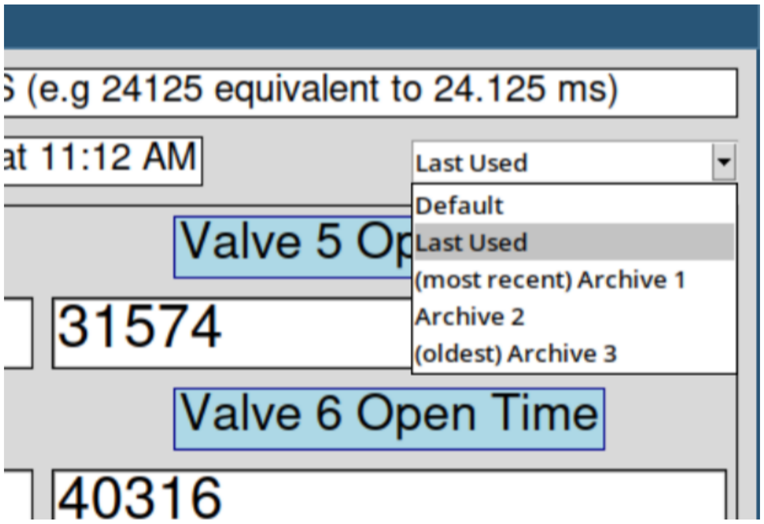


Figure 14: Users can Restore old Timings via the Timing Profile Dropdown Selection Menu in the ‘Manual Valve Timing Adjustment’ window.

4.3 Valve Priming

4.3.1 Why is this Useful?

The final primary feature of the ‘Valve Testing / Prime Valves’ Window is valve ‘priming’. This is a fancy way of saying "open and close the valves a bunch of times". The reason this is an included feature is that it was found that simply leaving valves open for an extended period of time while downstream tubes were full of *only air* (in an attempt to fill them with stimulus) did *not* seem to allow for flow to the tip as desired.

However, it was found that rapidly opening and closing the valves repeatedly *did* effectively allow for flow all the way down to the tip of the tubes (albeit slowly).

4.3.2 Okay, How do I do it? (Priming Valves)

This process is very similar to the process described in Section 4.1. Follow the steps below. This guide assumes that you have the ‘Valve Testing’ window already open on your display as shown in Figure 8

1. **Switch to Priming Mode**: First, the user should click the orange button labelled ‘Switch to Priming Mode’ as shown in Figure 8. This will change the interface to look like Figure 15.
2. **Enter Valve Acutation Number**: Next, the user should enter how many times the valve should actuate (open and close) before stopping. The use should enter this value in the entry box under the label titled ‘How Many Times Should the Valve Actuate?’.
3. **Select Valves to Prime**: Finally, the user should select the buttons corresponding to the valves to be primed. To understand what a valve button looks like when it is selected, refer to Figure 9.
4. **Begin Prime Procedure**: The application is now ready to begin the procedure. Press the orange ‘Begin Priming’ button to begin.
5. **Confirm Prime Settings**: The application will generate a pop-up window similar to that shown in Figure 10. The user should review the settings in the pop-up window to confirm these are correct. If they are correct, pressing ‘Yes’ will begin the test procedure.

Note: The procedure can be stopped at any time by pressing the yellow ‘STOP PRIMING’ button. The software *does not* need to be reset to start a new priming procedure. Once can start and stop as many times as is necessary, changing settings if desired.

After the procedure is complete, check the tubes for the valves just primed. If stimulus has filled the tube to the tip (end nearest spout), then the procedure was successful and the tubes are ready for an experiment. If not, repeat the process.

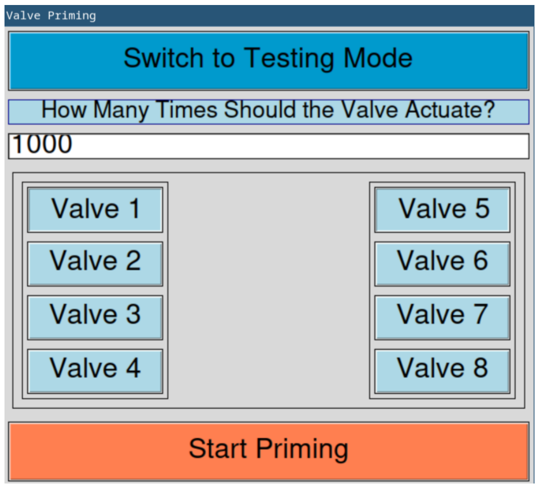


Figure 15: Users can 'Prime' Valve by Clicking 'Switch to Priming Mode' in the 'Valve Test' Window.

5 The ‘Valve Control’ Window

5.1 What does this Window do?

The ‘Valve Control’ Window is where users can navigate to manually switch on or off valves for arbitrary lengths of time. They may also manually move the rig door up or down in this window. The ‘Valve Control’ window is shown in Figure 16.

5.2 Controlling the Valves

All valves begin in the off state and can be switched on by pressing the button corresponding to the valve the user would like to open. Valves will remain open until user selects this button again.

This functionality is particularly useful for pressurizing valves and quickly running cleaning solution through the stimulus delivery tubes.

NOTE: Be aware and mindful that the valves get *very hot* when left open for long periods of time.

5.3 Controlling the Door Motor

The door motor can be moved down by pressing the green ‘Move Motor Down’ button in this window. The door can be left open or closed as long as the user would like and can be moved up and down as many times as necessary.

NOTE: The user should *wait* until the motor movement is *completely finished* before pressing the button again to initiate another movement.

Pressing the button before the movement has completed will interfere with the Arduino board’s ability to keep track of the current location of the door and will result in the door moving too far out of range, potentially **damaging rig components**. Ensure movements have completed before beginning another.

IMPORTANT: The door motor **must be** in the **UP** position before starting an experiment. The application assumes this to be the case. If the door is left down, similar symptoms as described in the above note will occur. Ensure the motor is in the UP position before leaving this window and beginning an experiment run.

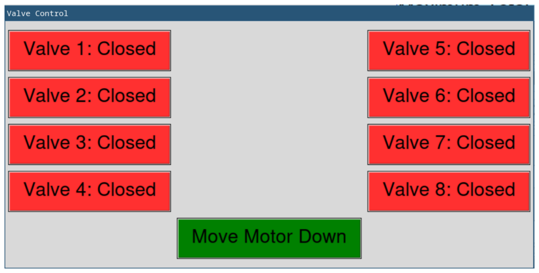


Figure 16: Users can Manually Control Valves as well as the Door Motor in the 'Valve Control' Window.

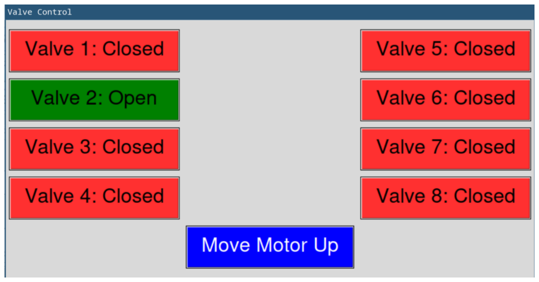
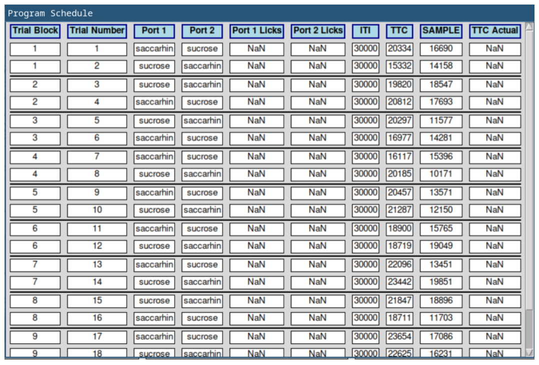


Figure 17: Valve Buttons Turn Green on User Press Indicating Valve is Now Open. Motor Button Turns Blue upon User Press to Indicate Door is now Down.

6 The ‘Program Schedule’ Window

6.1 What does this Window do?

The ‘Program Schedule’ Window allows the user to view one of the primary ‘dataframe’ components where the software stores experiment data. An example of this window when populated with data from an upcoming experiment is shown in Figure 18.

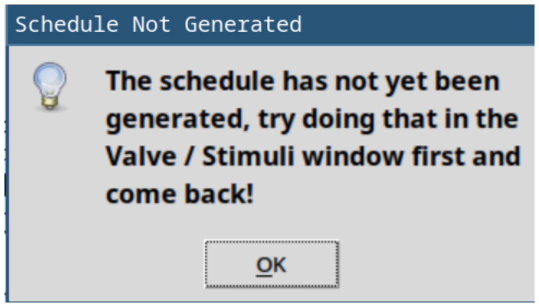


Trial Block	Trial Number	Port 1	Port 2	Port 1 Licks	Port 2 Licks	ITI	TTC	SAMPLE	TTC Actual
1	1	saccarhin	sucrose	NaN	NaN	30000	20334	16690	NaN
1	2	sucrose	saccarhin	NaN	NaN	30000	15332	14158	NaN
2	3	sucrose	saccarhin	NaN	NaN	30000	19820	18547	NaN
2	4	saccarhin	sucrose	NaN	NaN	30000	20812	17693	NaN
3	5	sucrose	saccarhin	NaN	NaN	30000	20297	11577	NaN
3	6	saccarhin	sucrose	NaN	NaN	30000	16977	14281	NaN
4	7	saccarhin	sucrose	NaN	NaN	30000	16117	15396	NaN
4	8	sucrose	saccarhin	NaN	NaN	30000	20185	10171	NaN
5	9	saccarhin	sucrose	NaN	NaN	30000	20457	13571	NaN
5	10	sucrose	saccarhin	NaN	NaN	30000	21287	12150	NaN
6	11	saccarhin	sucrose	NaN	NaN	30000	18900	15765	NaN
6	12	sucrose	saccarhin	NaN	NaN	30000	18719	19049	NaN
7	13	saccarhin	sucrose	NaN	NaN	30000	22096	13451	NaN
7	14	sucrose	saccarhin	NaN	NaN	30000	23442	19851	NaN
8	15	sucrose	saccarhin	NaN	NaN	30000	21847	18896	NaN
8	16	saccarhin	sucrose	NaN	NaN	30000	18711	11703	NaN
9	17	saccarhin	sucrose	NaN	NaN	30000	23654	17086	NaN
9	18	sucrose	saccarhin	NaN	NaN	30000	22625	16231	NaN

Figure 18: The Program Schedule Window allows Users to View Basic Current Experiment Data. This Figure Shows the Window Prior to Experiment Beginning, thus the NAN Values for Lick Data.

6.2 This Window won't Open due to Error... Why? (Experiment Schedule Error)

The user may see an error when attempting to open this window if they have not yet generated the experiment schedule for the experiment run. The error one may see is shown in Figure 19. The steps to generate the schedule are outlined with references to specific sections in Section 3.1.1.


Schedule Not Generated

The schedule has not yet been generated, try doing that in the Valve / Stimuli window first and come back!

Figure 19: The Program Schedule window will not Open Until the Program Schedule has been Generated for the Experiment.

6.3 What do All these Fields Mean?

This section explains each column, its purpose, and the data/units held within the column's rows where applicable. Columns are explained from left to right as they appear in the Program Schedule window.

1. **Trial Block**: A trial *block* is a group of trials that contain all possible pairings (unique stimuli pairings) of stimuli delivery during an experiment.

   For example, for an experiment with two stimuli in use for each port, (sucrose, saccarhin Port 1 | sucrose, saccarhin Port 2), each stimulus must be delivered on each port. So there are number of valves / 2 pairings for a trial block. Here there would be 2 trials per block. For the first trial there may be Port 1: Sucrose | Port 2: Saccarhin. There must also then be the opposite of this delivery for trial two: Port 1: Saccarhin | Port 2: Sucrose. This example is shown more clearly in Figure 18.

2. **Trial Number**: Trial number is simply the trial number of the current trial beginning at 1 and ending at total number of trials.

3. **Port 1**: Port 1 displays the stimulus to be delivered on Port 1 for the trial in the row.

4. **Port 2**: Port 2 displays the stimulus to be delivered on Port 2 for the trial in the row.
5. **Port 1 Licks**: Port 1 *Licks* is updated after a trial occurs and shows the number of times Port 1 was licked for this trial.
6. **Port 2 Licks**: Port 2 *Licks* is updated after a trial occurs and shows the number of times Port 2 was licked for this trial.
7. **ITI**: ITI shows the allocated time for ITI state for a given trial. This time is given in milliseconds.
8. **TTC**: TTC shows the allocated time for a TTC state for a given trial. This time is given in milliseconds.
9. **SAMPLE**: SAMPLE shows the allocated time for a Sample state for a given trial. This time is given in milliseconds.
10. **TTC Actual**: TTC Actual shows the *actual time used* for a TTC state for a given trial. This is necessary because three rat licks will transition the program into a Sample state early, and will not use all allocated TTC time. This time is given in milliseconds.

7 The 'Valve / Stimuli' Window

7.1 What Does this Window Do?

The 'Valve / Stimuli' Window is where users should navigate to label the stimuli that will be delivered for a given valve. This is an important step to generating the program schedule as described in Section 3.1.1. The 'Valve / Stimuli' window as it will be found on first open is shown in Figure 20.

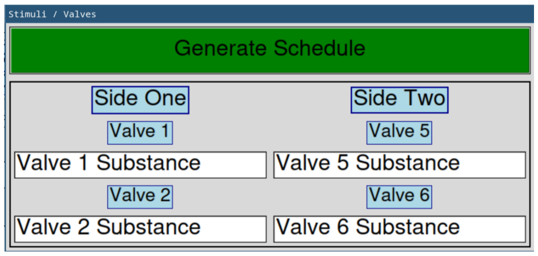


Figure 20: The 'Valve / Stimuli' Window is Initially Filled with Default Values Corresponding to the Labelled Valve Number of a Valve.

7.2 How Do I Use this Window? (Generating an Experiment Schedule)

The 'Valve / Stimuli' window is where the stimuli schedule is generated for an upcoming experiment. This process will generate not only stimulus lineup for each port per trial for the experiment, but also state base interval times (ITI, TTC, etc) and the random times to be added to these base times. Assuming that interval times and other experiment configuration variables have been set as desired in the main program window (Section 3.4), the remaining step prior to schedule generation is naming the stimuli for each valve.

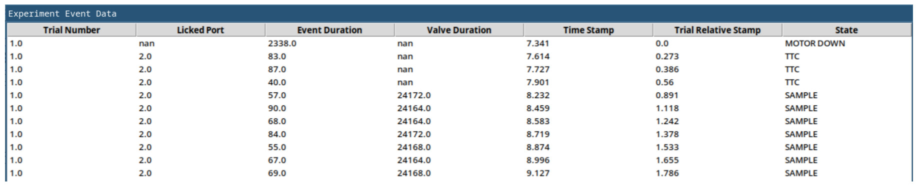
When assigning stimuli names to valves, the user should start at valve one and continue downwards (enter only in new rows, do not enter in column two). This will simplify the process, as the program will automatically assign names to valves on column two based on names entered in column one. When names have been assigned, users can press the large green 'Generate Schedule' button at the top of the 'Valve / Stimuli' window.

If this process is successful, the 'Valve / Stimuli' window will disappear, and a new 'Program Schedule' window will appear displaying the schedule to be followed for the next experiment as shown in Figure 18.

8 The ‘Event Data’ Window

The ‘Event Data’ Window shows the user more detailed information about events that occur during an experiment run. While the program schedule shows much of the desired information about the experiment to the user, it lacks information like event timestamps, durations, etc. The ‘Event Data’ window fills in these data gaps.

Details about rat licks as well as door motor movement details are contained in this window. Because two different event types are stored here, some columns may not always contain a value. This is discussed in more detail below.


Experiment Event Data

Trial Number	Licked Port	Event Duration	Valve Duration	Time Stamp	Trial Relative Stamp	State
1.0	nan	2338.0	nan	7.341	0.0	MOTOR DOWN
1.0	2.0	83.0	nan	7.614	0.273	TTC
1.0	2.0	87.0	nan	7.727	0.386	TTC
1.0	2.0	40.0	nan	7.901	0.56	TTC
1.0	2.0	57.0	24172.0	8.232	0.891	SAMPLE
1.0	2.0	90.0	24164.0	8.459	1.118	SAMPLE
1.0	2.0	68.0	24164.0	8.583	1.242	SAMPLE
1.0	2.0	84.0	24172.0	8.719	1.378	SAMPLE
1.0	2.0	55.0	24168.0	8.874	1.533	SAMPLE
1.0	2.0	67.0	24164.0	8.996	1.655	SAMPLE
1.0	2.0	69.0	24168.0	9.127	1.786	SAMPLE

Figure 21: The event data window

8.1 Detailed Column Breakdown

Here the contents of each column is discussed in detail. This includes general details as well as units of displayed information where applicable.

1. **Trial Number**: This column displays the trial that an event occurred in.
2. **Licked Port**: This column will **not** always contain a value. It will only be filled if the event being recorded is a rat lick. Motor movement recordings will mark this columns value for this row as ‘nan’.
3. **Event Duration**: This column is used for *both* licks and motor movements. It records the length of time the event occurred in from onset to offset.

   For licks, this is the time difference between the moment the lick beam is broken until the moment it is restored. For door motor movements, it is the time difference between motor movement instruction until the motor reaches the target position. This time is represented in **milliseconds**.
4. **Valve Duration**: Valve duration is only marked for the subset of lick events that occur within the ‘SAMPLE’ state. This is because valves are not opened during ‘TTC’ state. Value stored as nan for licks in ‘TTC’ state or motor movement events. This time is represented in **milliseconds**.

5. **Time Stamp**: This value is the time the event occurred relative to the program start time (t=0) in **seconds**.
6. **Trial Relative Stamp**: This value is the time the event occurred relative to the trial start time. Trial start here is defined as TTC start time, because this is the moment the rat can begin to engage in the trial. This time is represented in **seconds**.
7. **State**: State can hold different values depending on the *type* of event that occurred. Motor movements will state the *type* of motor movement that occurred (MOTOR [UP, DOWN]). Licks will show the program state at the time of the lick (TTC, SAMPLE).

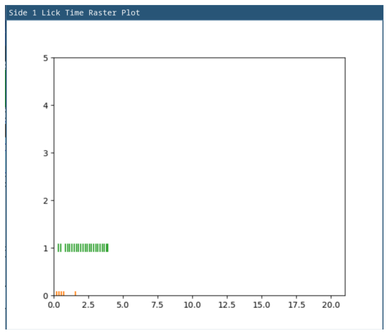
NOTE: The method by which data in this window is refreshed is pressing the ‘Event Data’ button in the main program window. This button serves as the method to access the window and update with new information that has occurred. The window will **NOT** automatically refresh. The user must manually trigger updates via this method.

9 The 'Rasterized Data' Window

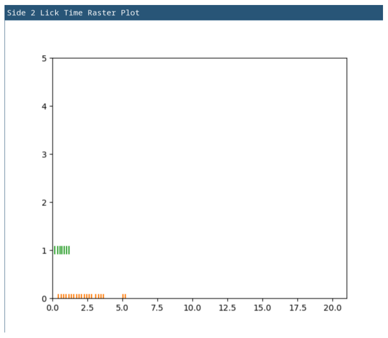
Pressing the 'Rasterized Data' button in the main program window produces two identical raster plots representing the licks per trial per port. This window is useful to ensure that reported licks from the Arduino board seem to match with the accepted understanding of how quickly rats can lick maximally.

In other words, it allows users to ensure that the data being recorded is accurate, and no 'phantom licks' are being recorded due to equipment inadequacies or failures.

In this window, the Y-axis represents trial number, while X-axis values represent the time stamp associated with a given lick.



(a) Example of Data Filling the Side (Port) 1 Raster Plot.



(b) Example of Data Filling the Side (Port) 2 Raster Plot.

Figure 22: Raster Plots are Seperated by Port to make Analysis more Straightforward.