BioBox User Manual

C. Perkins and K. Burczyk
09-08-23
v2.1.0

This document provides the information on how to use the BioBox UI to interface with the BioBox system.

Change Log

+ Vial calibration and selection pages changed to work with up to 99 vials

Contents

1 Introduction

BioBox is a containment system for biohazard group 1 (and in the future, 2) viruses, allowing them to be placed safely in front of a beamline. The BioBox UI (this software) is the software component to interface with the BioBox system. The software is intended to communicated with a pair Arduinos via COM ports, with the Arduino containing it's own control and safety code to directly interact with the motor drivers, relays, and sensor. It also is intended to interface with the AccelNET software for the Pelletron ion accelerator.

This document is the quick start guide for using the BioBox UI application. It contains the information necessary to install and use the software, as well as develop Arduino code to interface with the UI if necessary.

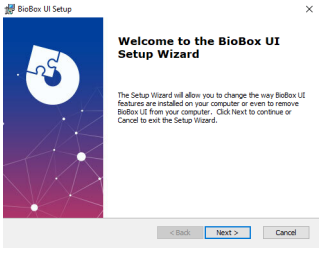
For more information please visit the repository on GitHub at https://github.com/Chaddyfynn/BioBox-Controller for the source code, or contact the developer by sending an email to charlie.perkins@student.manchester.ac.uk.

2 Quick Start Guide

The Quick Start section is intended to give the user an easy tutorial for using the application. Detailed information about the logic and programming are contained within the Developer Information section (Section 3).

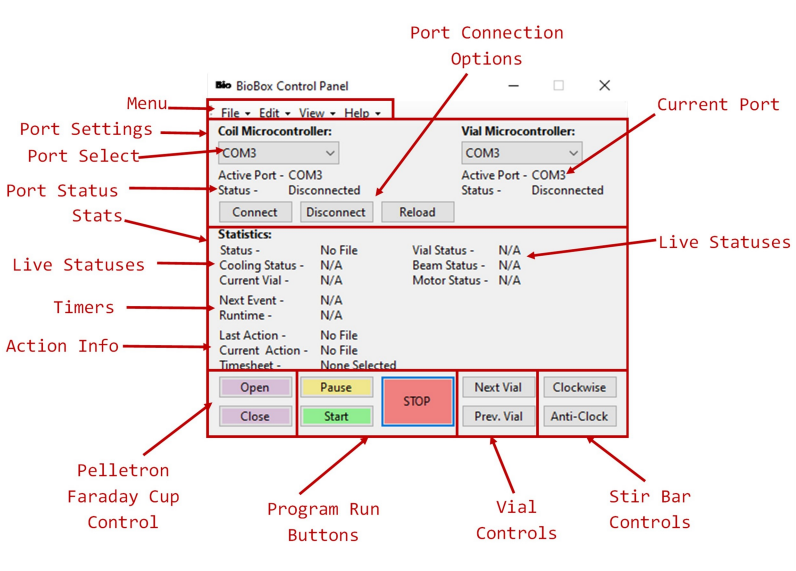
2.1 Installation

To install BioBox UI visit the GitHub repository release section from https://github.com/Chaddyfynn/BioBox-Controller/releases for the most recent release. This guide will show how to install v2.1.0, but the installation process is unlikely to change with future versions. From the GitHub release page select the most up-to-date release and click the BioBox.UI.Installer.msi link to download the installer. Open the installer and follow the on screen instructions. Once this is complete, you should be able to open BioBox UI from your start menu by searching for BioBox UI.



2.2 User Interface

2.2.1 Control Panel



The Control Panel interface is the first interface that opens upon startup of this application. The Control Panel's purpose is to run and monitor pre-programmed experiments (see 2.2.2). The Control Panel has three main areas - the Port Settings area, the Statistics area, and the lower button array.

The Port Settings area allows the user to select which COM Port to send specific requests to. The selection of which COM port to use will be specific to the user's computer, and hardware. To select a port, click on the dropdown menu and select the COM port which is responsible for communication with the relevant microcontroller. This information can be found in the Device Manager which comes as standard in all versions of Windows since Windows 95, or you can find this information in the Arduino IDE. Once selected, click Connect to allow the BioBox UI permission to send requests over the serial connection. If the device was connected after opening BioBox UI, then you may need to reload the available ports.

The Statistics area shows what messages have been sent to the microcontrollers, as well as useful information about the current statuses of the various hardware elements, and also information regarding the runtime, and timing of the experiment. Most statistics are self-explanitory, however it is important to note that this information is not a reflection of what the microcontroller is doing per se, but rather what BioBox UI has sent to the microcontroller to do. Malfunctions in the microcontroller are delt by the microcontroller, not BioBox UI, so do not appear in this version of BioBox UI.

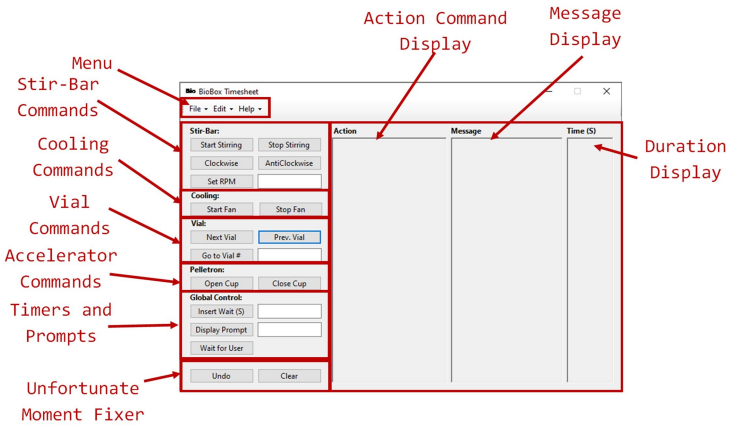
The lower button array contains controls to start, stop, or pause the current program. It also has a selection of buttons to control some of the hardware live - such as opening or closing the faraday cup, changing the vial selection, or rotation direction of the stir bar.

To load a program, click 'File' and then 'Open Timesheet'. You may then select a valid BioBox Timesheet CSV file to load into the control panel. To run the show simply click Start and wait until

either a pre-programmed dialog is displayed, or until the program terminates. You can pause the program by clicking Pause. This disables the runtime and next event clocks and clock cycles so that no future events will occur until Pause is clicked again. To terminate the program prematurely, click Stop. This will immediately end the running of the program and send requests to the microcontrollers to turn off cooling, and stir-bar rotation.

From the Control Panel, you can open all the other features of this software. Under the 'Edit' dropdown, you can access the Timesheet Editor. And under 'View' you can open the Stir Bar Controller (the main interface in all versions up to v2.0.0) and the Vial Controller. Finally, this document is accessible from the 'Help' dropdown under 'View Documentation', and the program source code is accessible from 'View Source'.

2.2.2 Timesheet Editor



To open the Timesheet Editor panel, in the Control Panel interface click on the 'Edit' dropdown menu, then 'Timesheet'.

By default, this opens with no loaded file, even if one has been opened in the Control Panel. To open, view, and edit a pre-existing file, click 'File' and then 'Open Timesheet'. Otherwise, you can begin programming immediately, a blank file is loaded by default.

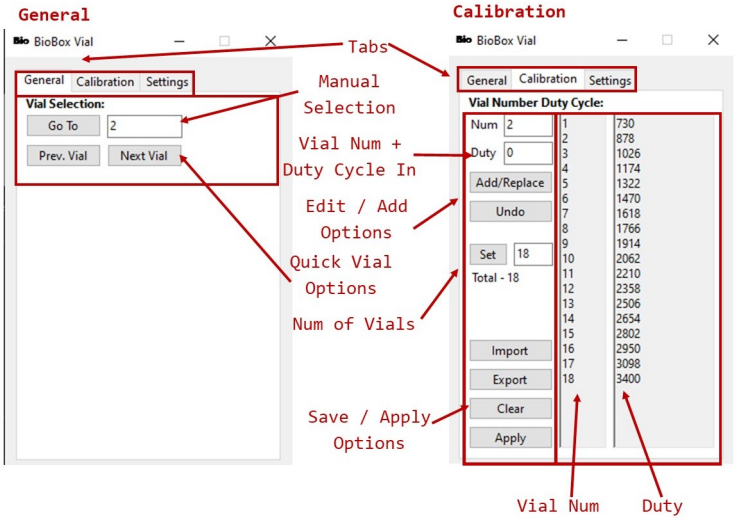
The interface has two main panel - the left most area contains buttons that correspond to actions performed by the microcontrollers. The right area has three empty boxes which show the current programmed experiment. When the user clicks on one of the buttons from the left panel, it appears as an event in the three right boxes. The Action box contains the description of the event, as well as which number it appears in the list (starting from 1). The Message box shows either what user inputted message is displayed (for user prompt), or what command is sent over the serial port to the microcontrollers. The duration display shows how long the period of time is between this and the next event. This usually only displays when the user selects a Wait event, with the user defining the time period in seconds.

Each of the buttons in the left panel (except Undo, and Clear) specify an event during the experiment. Events without wait times between them will be performed 5ms after the last event. If the user wants no action to be performed for some period of time, they may select Wait

and specify the duration. The user can also choose to display a dialog box (with or without a custom message) between events which the user during the experiment will have to click OK on to continue to the next event.

To save the program, click 'File' then Save As. This will open the default Windows Save Dialog and prompt you to choose a name and location. The file should be saved as a CSV by default, but text files are also permitted if necessary. CSV files give the user the option to edit the program in an external application (such as Google Spreadsheets, Microsoft Excel, or even custom applications written in Python for instance). Please note however, that changing the file without an in depth knowledge of the formatting of the file can have adverse consequences when running the program in the Control Panel. Please see 3 for detailed information.

2.2.3 Vial Controller



To access the Vial Controller, in the Control Panel click 'View' then 'Vial Controller'.

The Vial Controller panel is used to control the vial position live. There are three tabs: General, Calibration, and Settings.

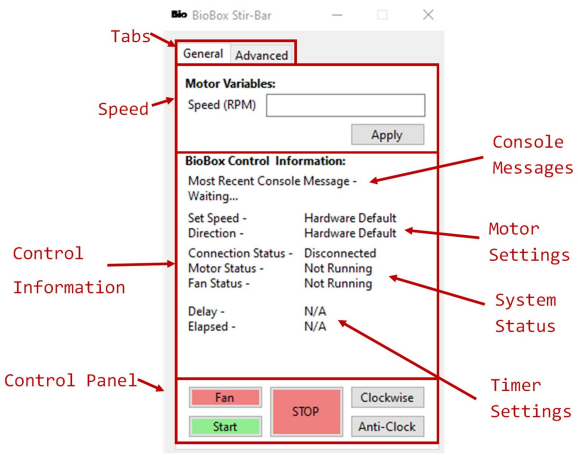
In General, the Vial Selection area allows the user to type in a specific vial number, and click 'Go To' to instantly move the carousel to the selected vial number. The firmware on the microcontroller currently positions the servo motor at the desired duty cycle + 100, then goes to the desired duty cycle, so that the servo motor always approaches the final position from the same direction (back-lashing algorithm). There also exists options to choose to move to the next vial or previous vial using the 'Next Vial' and 'Prev. Vial' buttons.

The Calibration page allows the user to input the duty cycles for turntable servomotor at each vial position. Firstly the user must configure the setup by typing in the number of vials that span the range of rotation. As of writing this, the turntable has 18 vials in 270 degrees. The number of vials can be inputted by next to the 'Set' button. The user must input the first and last vial duty cycles at a minimum. But they may also insert the duty cycles of any particular vial number, subject to the vial

number being less than the maximum number of vials. This can be done in any order. Clicking 'Apply' will fill in the rest of the vial duty cycles based on the first and last vial duty cycles. Clicking 'Apply' will also re-order the sequence into a a sequence with ascending vial numbers. If the user wishes to change a duty cycle, they can type in the vial number and duty cycle into the top two boxes respectively, and then click Add/Replace. This will add it to the end of the list. However, upon clicking 'Apply' the setting will be subsumed into the large sequence, overwriting the old number. If the user wishes to set this back to the default, they should type in the vial number into the vial number box and 0 into the duty cycle box. The they should click 'Add/Replace', then 'Apply'. Clicking 'Apply' also sends the current settings to the microcontroller sequentially in 5ms intervals. The calibration settings can be exported to a *.csv file for import in the future, or used to generate a calibration curve (using external software/code - not provided). Finally, the 'Clear' button deletes the contents of the text boxes to the right.

The Settings tab contains the port connection settings as seen previously in the 2.2.1

2.2.4 Stir Bar Controller



To access the Stir Bar Controller, in the Control Panel click 'View' then Stir Bar Controller.

The UI for BioBox is very minimalist, so there is not much the user needs to get used to. There are two main pages: General, and Advanced. The General page is where the user sends commands to the BioBox system by pressing on-screen buttons. All the buttons on the General page act instantly, so any changes will be instantaneous. The only exception to this is the Start button, which will act with a delay if the user has entered a valid argument. The Advanced page controls any non-instantaneous changes, such as port and timer settings.

The General page has three main areas - Motor Variables, BioBox Control Information, and the control panel.

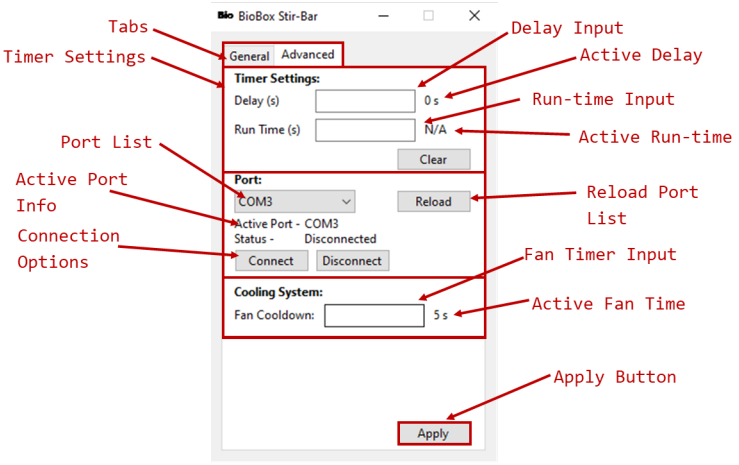
The Motor Variables area contains an input box to control the frequency of rotation of the stir bar. The input box sanitises text, so only zero and positive numbers will be accepted as possible speeds. Changes will not be accepted until the user clicks Apply.

The BioBox Control Information

area contains useful text detailing the current system properties.

1. Most Recent Console Message: Displays debug messages that would be printed to the console if this were a debug release.
2. Set Speed: Displays the currently selected motor speed.
3. Direction: Displays the current spinning direction of the stir bar (Rotate mode).
4. Connection Status: Displays whether or not the application will send a request over the port to the Arduino.
5. Motor Status: Display what the application is currently controlling the motor to do. Options include (1) Not Running, (2) Running, (3) Delayed, (4) Finished, and (5) Cancelled. (1) only appears on start-up. (2) displays any time the software has sent a request to run the motor. (3) displays when the user has pressed Start, but a delay time was entered so the motor is currently idle, but imminently running. (4) displays when the motor has finished running either through user interruption (STOP button) or the run time has been fulfilled. (5) displays when the delay timer was counting down, but the user interrupted the countdown with STOP.
6. Fan Status: Displays whether the last fan control message to the arduino was a running (T1) which shows 'Running' or not running (T0) message, which shows 'Not Running'.
7. Delay: Displays the amount of time until the motor begins running after Start has been pressed. This is only a valid item when the Motor Status is 'Delayed'.
8. Elapsed: Displays the current amount of time the motor has been running (valid when Motor Status is 'Running'), or displays the amount of time the motor ran on the last run (Motor Status is 'Finished').

From the control panel the user can choose to Start or Stop the motor, as well as change the direction of rotation or vibration. The Fan button allows the user to control whether or not the fan will run during the operation of the stir bar. When the button is green, the fan will turn on when the start button is pressed. When the button is red, the fan will not operate. The fan operates within a cooldown period to further cool the coils after the stirbar has stopped. Double clicking the stop button immediately stops the fan rotation - single clicking will use the cooldown timer.



The Advanced page has three main areas - Timer Settings area, and the Port settings area, and the Cooling System area . The top row in the Timer Settings area controls the delay settings. The user can input an *integer* number of seconds they would like the system to wait after Start has been pressed before the motor begins running. The currently set delay time in seconds is displayed to the right of the delay input box. The user can cancel the countdown by pressing STOP at any point during the delayed period. The next line down controls the Run Time settings. The text input box allows the user to restrict the duration of action of the motor to a given number of seconds. Likewise, it must be an integer. Decimal inputs are not accepted. After this amount of time, the BioBox UI will send a request to stop the stir bar motion. The user can still stop the stirring before the end of the timer by pressing STOP on the General page. The text to the right of the input box displays the currently selected run time in seconds. All changed settings must be updated by pressing Apply at the bottom before any changes are secured. The Clear button resets the Delay and Run Time to their defaults, 0s and indefinite respectively.

The Port settings section allows the user to connect to the BioBox system using COM ports. The BioBox UI checks what ports are currently active on loading of the application and displays them in the dropdown menu. If BioBox is connected to the computer after the application started then the ports list will need reloading using the Reload button to the right. The Active Port text below the ports list shows which port is currently selected and will be used to communicate. COM5 is usually used for Arduino communication. Check the Arduino IDE for the selected COM port if you have any doubts. The status text displays Connected, Disconnected, or User Abort. Though the text implies that this means BioBox UI is *currently* communicating over that port right now, this is false and this simply means whether or not the application will connect to the port when a request is made by pressing one of the buttons on the General page. The app does not 'connect' to the port until a request is made by pressing one of the buttons on General. For most purposes, the user will want to always click Connect upon opening the application and selecting the correct port. But often for debugging purposes it is useful to disable the connection request to stop BioBox UI from crashing after sending a request over a non-Arduino COM port. Upon clicking Connect, a dialog box will pop up to ensure the

user is aware that they are about to remove the safety net over the General page that stops unintended requests being sent over COM ports.

The Cooling Settings area contains an input box to choose how long the fan should run for after the stir bar coils have stopped. The number cannot be 0 for safety reasons, and inputting so will revert the setting back to its default which is 5s.

2.3 Connecting to an Arduino

To connect to the Arduino, plug the Arduino into the USB port on the computer and have the port the Arduino uses to communicate handy. You can find this information in the Arduino IDE, or by contacting the BioBox Arduino programmer.

Open the BioBox application and in whichever environment is neededе (Control Panel, Vial Controller, or Stir Bar Controller) open the page containing the port connection settings. Open the port dropdown menu and select the correct port. Click Connect. This will open up a dialog box to ensure you want to connect to the correct port. Click Yes to close the dialog if you are sure this is correct.

The Arduino should now be connected to the application. To check, go back to the General page and click Single Step to move the stir bar once. You should be able to tell upon visual inspection whether the Arduino has connected or not.

2.4 Simple Stirring

In this section we will create a simple stirring program to rotate the stir bar at 700 RPM for one minute.

To set up a simple stirring program, connect the Arduino as described in the last Section (Section 2.3) in the Stir Bar Controller. Travel to the Advanced page and type in a desired number of seconds for the stir bar to run in the Run Time input box. In this example we want a 1 minute run time so we will type 60. Then select Apply at the bottom of the interface.

Go back to the general page, then in the speed input box type 700. The default is 500. Then click Apply. You should not have to select a rotation direction as the Arduino should be programmed to rotate clockwise by default - though this will not appear in the BioBox Control Information section unless the user has pressed Clockwise. Click on the Fan button and ensure it has turned green.

Finally, press Start. The Motor and Fan statuses should have changed to 'Running'. Similarly, you should see the Elapsed timer now tick about once a second (subject to PC performance) and the stir bar should be rotating clockwise. At 60s the software will automatically request the stir bar to stop rotating and the Motor Status will change to 'Finished'. At this time the Elapsed timer will continue to display '60 s' and the stir bar should have stopped. 5s after these events, the fan should turn off too, and the Fan Status will display 'Not Running'.

2.5 Simple Automated Stirring

In this section, we will recreate the same program as seen in the last section, but in the Timesheet Editor and the run the program in the Control Panel.

Open the Timesheet Editor from the Control Panel. Firstly, we will want to configure the stir bar rotation settings so that when the stir bar starts rotating, it is doing so correctly. Begin by setting the rotation speed to 700RPM by typing 700 into the input box to the right of the 'Set RPM' button. Then click 'Set RPM'. This should now appear in the three right boxes as '1. Set RPM to 700', 'S700', '/'. Then set the rotation direction to clockwise by clicking 'Clockwise'. This should have added '2. Rotate Clockwise', 'r0', '/' to the three boxes. To turn on cooling click 'Start Fan', and then to start the stirring click 'Start Stirring'. Since we have not added any wait times, these commands will all be programmed to be requested serially within 5ms of the last. Now we want the stir bar to mix for 60s, so we will type '60' into the box to the right of the 'Insert Wait (S)' button under the 'Global Control' section. Then click 'Insert Wait (S)'. Finally we need to setup what happens after 60s. We need to stop the stirring, and then 5s later the fan. So click 'Stop Stirring', then type '5' into the wait box and click the 'Insert Wait (S)' button, then click 'Stop Fan'. This will do exactly the same as was setup in the last section.

To save the program, click 'File' then 'Save As' and save it to a suitable location with a name like 'Simple Automated Stirring'. Then navigate over to the Control Panel, click 'File' and then 'Open Timesheet'. Navigate to your save location and double click on the file you just made.

This should now have loaded into the Control Panel. You can check as the Status statistic will read 'Active File' to the right, and the Next Action statistic should have '1. Set RPM to 700' next to it. Everything is now in place to run the program. Simply click 'Start' watch everything take place, and once it is done the status will have turned from 'Running' to 'Finished'.

Well done, you can now program full experiments using the Timesheet Editor and the Control Panel. This removes the need to do anything live during the experiment. But note, you can always override commands in the Stir Bar Controller and Vial Controller windows if absolutely necessary.

3 Developer Information

This section contains important information regarding the functionality of the software, such as potential errors, and serial port codes for the Arduino programmer to check for on the port connection.

3.1 Port Codes

All serial port codes are formatted [LETTER][NUMBER(s)]. The letter signifies the action required, and the number details the specific setting. The baud rate is 9600.

c. Control

- 0: Stops the motor
- 1: Starts the motor

S. Speed

- NUM: Set rotation speed to NUM

r. Rotate/Direction

0: Rotate clockwise

1: Rotate anti-clockwise

T. Cooling

0: Fan off

1: Fan on

V. Vial Movement

0: Move to next vial

1: Move to previous vial

P. Vial Number

NUM: Set active vial to NUM

Current communication is only 1 way (BioBox UI to Arduino), but a future version intends to bring two-way communication to allow the Arduino to contain safety measures, checks, and live updates on the BioBox system to be displayed in the BioBox UI.

3.2 Timesheet Format

The Timesheet Editor saves all programs as *.csv files by default. They are have a comma delimiter, no spaces with one full command per line, and four columns. The first column contains the description of the task with the format 'TASK NUM. TASK DESCRIPTION'. The second column contains the amount of time the event lasts for in an unsigned integer number of seconds. The third column contains either the message sent over the serial port (formatted like 3.1) or the text to display when a prompt appears (if prompt command). The fourth column contains a command string formatted as follows:

S. Send string over COM port to stir bar microcontroller

V. Send string over COM port to vial microcontroller

P. Send string over network to AccelNet PC for Pelletron

D. Delay the program specified number of seconds.

U. User prompts

0: Display user defined message in a prompt

1: Display a 'waiting for user' prompt

Any formatting besides this may have unintended consequences, which may disrupt the experiment.

3.3 Servo Circuit Connections

Components: DSS-M15S 270 deg Metal servo, Adafruit 16-Channel 12-bit Servo Driver PCA9685, 5V-7V power supply, Arduino Mega 2560.

The servo is controlled using the Arduino and the servo driver by adjusting the duty cycle of the pulse width modulated (PWM) signal. The servo driver can send PWM signals of 4096 different duty cycles, and the servo will move to a specific position according to the duty cycle input.

The Adafruit Servo Driver is connected to a power supply that can

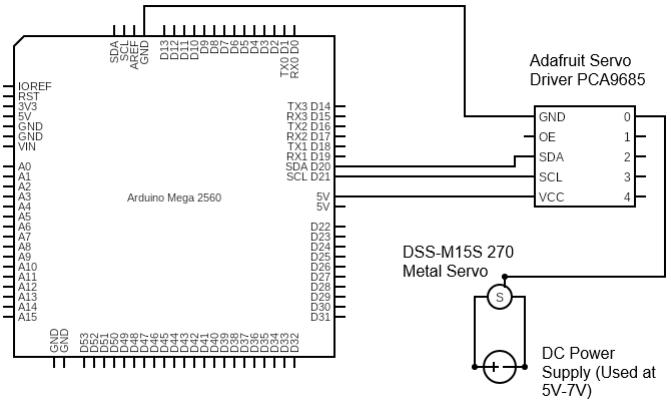


Figure 1: Circuit diagram controlling the servo using a Arduino Mega 2560, and a Adafruit Servo Driver PCA9685 (Simplified Diagram of the servo driver not including many of the unused pins). The servo is receiving 12 bit control signals from the servo driver, to give a maximum resolution of 4096 possible duty cycles. The servo is powered by connecting a 5V-7V DC power supply to the servo driver ground and power pins, and the servo data pin is connected to the 0 channel on the servo driver.

apply 5V-7V, and the servo ground, power, and data pins are connected to the first servo slot. The Adafruit is connected to the Arduino Mega via the 5V pin, ground pin, SDA pin, and SCL pin, so that data can be transferred between the servo driver and the Arduino.

3.4 Calibrating Turntable

Duty cycle values between 0 to 4096 can be chosen, as the Adafruit servo driver can handle 12 bit digital signals, the recommended servo frequency is 330Hz as the servo will respond to the greatest range of duty cycle values at this frequency, allowing a greater angular resolution to be achieved. The expected response range for the servo is a duty cycle value between 450 and 3600. Inputting these duty cycles will result in slightly more than a 270 degree range of motion, so these values are a good starting point for finding the 0 degree and 270 degree points.

To calibrate the turntable, use the calibration.ino code (or BioBox UI Vial Controller - Calibration tab), and vary the duty cycle by inputting integer values into the Serial, until the calibration vial is aligned with the beam line. Note down the duty cycle that corresponds to each vial so it can be written to the vial_selection.ino code (or submitted immediately to the Arduino by clicking Apply). It is easiest to start with finding the duty cycle which correspond to the first and last vials, so that a linear relationship between the duty cycle and the servo angle can be found by using the servo_calibration.py program, to make identifying the duty cycle for each angle easier.

To reduce the inaccuracy in the calibration, only change the value of the duty cycle in one direction, to avoid any backlash error, as during irradiation phases the servo will be expected to only turn in one direction to switch between the vials. By changing the duty cycle in only one direction, the maximum resolution becomes changes of 3 duty cycles, corresponding to a change of turntable angle of 0.25 degrees.

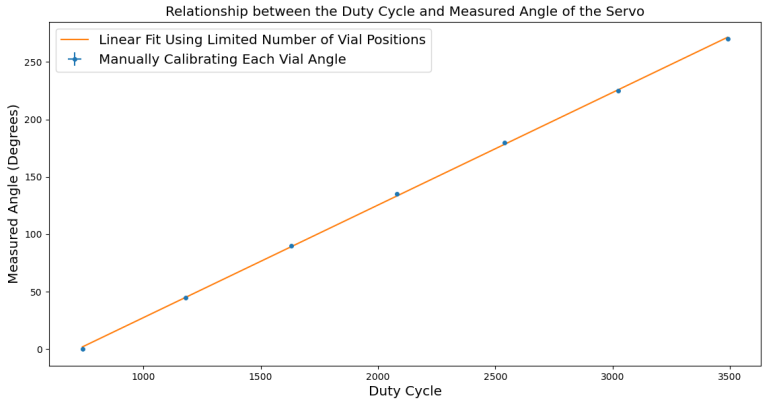


Figure 2: The plot shows the manually calibrated duty cycle to angle values, and a linear fit which was calculated using a 3 point calibration at angles of 0, 135, and 270 degrees. The error on the duty cycle is estimated to be 5μs and the error in the angle measurement is 1 degree, but they are too small to be seen on the graph.

3.4.1 Expected Calibration Results

Using the method described above, the duty cycle corresponding to each vial was noted, and the values were entered into the Vial Selection file. The plot of duty cycle against measured angle is expected to be perfectly linear however some discrepancy is seen, likely due to manufacturing errors in the servo, backlash errors, and human error in alignment.

The servoMin value corresponds to the duty cycle at which the servo is on the 0 degree vial. The servoMax value corresponds to the 270 degree vial.

The linear fit was calculated by using the manually calibrated duty cycles at the vials positioned at $\theta = 0, 135, 270$, however, it does not match the duty cycle values that were gathered by manually calibrating each vial. Therefore if the linear fit was used, the vial positions would not be accurately placed with respect to the beam line, leading to larger dose uncertainties. The plot in 2 shows that the linear approximation does not represent the angular range accurately, meaning that the most appropriate calibration method is to manually identify the duty cycle corresponding to each vial position, instead of relying on the linear relationship, as accuracy of the vial positions is essential.

After calibration of each vial position, the duty cycles values were input into the vial_selection.ino file and the turntable was ready to be used in the irradiator.

3.5 Using Vial Selection Tool

After the calibration is complete and the vial_selection.ino duty cycle values have been amended, the turntable can now be controlled by selecting which vial you wish to place in front of the beam line. For the most accurate movements of the servo, try to only move the turntable in one direction from the 1st vial corresponding to 0 degrees, to the last vial corresponding to 270 degrees.

Then when you wish to change the vial being irradiated, select the vial on the GUI and click go. Once the servo has finalized moving, irradiation and magnetic mixing can begin for the next irradiation experiment.

3.6 Removing the Turntable

Once all irradiation experiments are concluded and it is safe to enter the target room, the turntable can simply be lifted from the servo mount and the vials can be transported for analysis.

3.7 Coil Microcontroller and Electronics Setup

Components: 1x Arduino Mega 2560, 2x L298N DC Motor Driver, 1x Variable DC Power Supply Figure 3 shows the circuit diagram concerning the stir-bar electronics. Not illustrated in Figure 3 is a power board from which the motor drivers receive their supply power, as well as their common ground. The Arduino has a custom firmware written in C++ which reads characters from the serial port sent from the BioBox UI. These form char arrays which are then interpreted as defined in Section 3.1. The firmware contains a preset pin out sequence which switches on and off outputs of the L289N as well as the polarities too. The following table shows the clockwise running preset:

Sequence Position	In 1	In 2	In 3	In 4
1	1	0	0	1
2	0	0	0	1
3	0	1	0	1
4	0	1	0	0
5	0	1	1	0
6	0	0	1	0
7	1	0	1	0
8	1	0	0	0

The anticlockwise preset is simply the reverse running order (8, 7, 6, ...). This sequence has a high power output since half the time both all 4 coils are powered, and the rest of the time only 2 are powered. This sequence creates a smooth control.

In the system developed at the time of writing, the following table outlines which pins must connect to which:

Control Pin	Prehiperal Pin	Control Device	Prehiperal Device
8	IN 1	Mega 2560	L289N (Coil)
9	IN 2	Mega 2560	L289N (Coil)
10	IN 3	Mega 2560	L289N (Coil)
11	IN 4	Mega 2560	L289N (Coil)
-	GND	Power Board	Arduino Mega 2560
12	IN 1	Mega 2560	L289N (Fan)
-	GND	Power Board	L289N (Coil)
+	VSS	Power Board	L289N (Coil)
-	GND	Power Board	L289N (Fan)
+	VSS	Power Board	L289N (Fan)
-	-	Supply	Power Board
+	+	Supply	Power Board

The variable power supply can be swapped for a wall power supply with a suitable 5.5mm x 2.1mm DC connector.

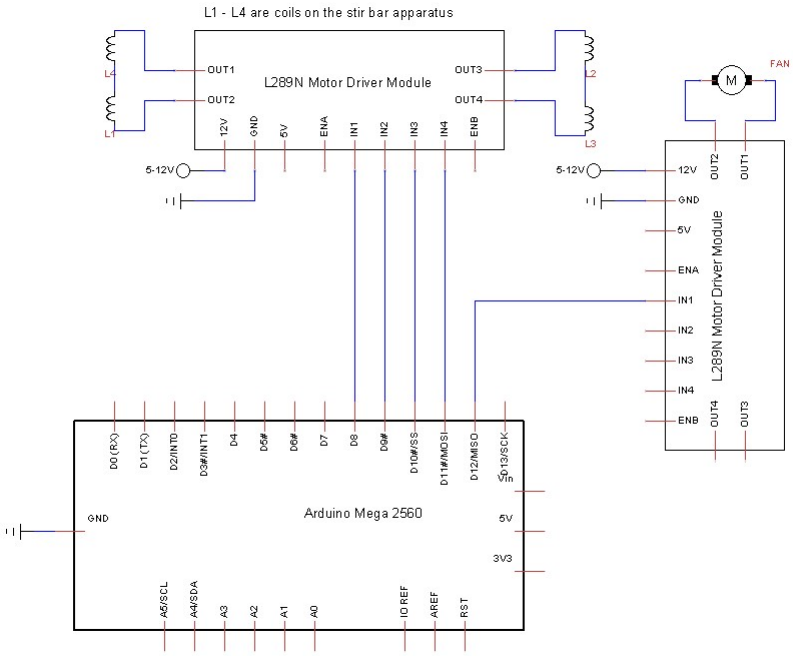


Figure 3: A circuit diagram of coil/stir-bar controller hardware. The Arduino Mega 2560 microcontroller is wired directly into the coil and fan L289N DC motor drivers with 4 and 1 individual signal cables respectively. All devices share a common ground.

3.8 Coil Heating / Cooling

The GikFun coils used in the stir-bar can get very hot. All components exist to support powered cooling if the developer chooses to add additional cooling. In its current state and at 12V with 2A current limiting, the coils reach a temperature of about 46C after 30 mins of run time with firm contact to the heatsink. Please not this is *without* thermal paste, where we expect to see lower running temperatures. 44C was reached by the coils after only 20 mins of run time (i.e a temperature rise of 2C in 10 mins), so it is sensible to assume that the temperature would not rise significantly had the coils been running for longer.

3.9 Known Bugs / Issues

As of writing this, there are two minor inconveniences. One minor inconvenience is that the input boxes accept C# int type only and do not round decimals. Inputting a decimal is treated the same as typing 'Why will this not take decimals?!!!!', and the program will disregard this input. The other minor inconvenience is that if the user incorrectly presses 'Yes' on the Connection dialog box without ensuring that the COM port is the correct one, then the program will crash and need closing in Task Manager.