Photologic Experiment Rig
Arduino Migration Guide

Blake Hourigan

March 18, 2025

Contents

1 Why the Migration?

If you are not interested in why but only *how to perform* the migration, skip to Section 2.

When I began working on this project again after about a year-long hiatus, I was struck by various aspects of the program that were much less effiecient than they should be. I began work to resolve some of these issues individually on the two different Arduino boards. I changed data structures, algorithms, moved to simpler structs rather than classes, etc. and I noticed incredible speed-ups in the tasks being performed on each board. As things became faster and more efficient, I began to wonder if having *two distinct Arduino boards* was necessary.

The processor on board is incredibly fast and the tasks being performed are so light that it seemed like overkill to have two individual boards performing different tasks. Not to mention the overhead of having two separate boards. Handling communication between two different boards meant that code was required to identify these boards. Often commands needed to be sent to two boards at once rather than combining commands to one board. Furthermore, the Arduino boards needed a mechanism to communicate with *each other*. The Laser Arduino needed to somehow communicate the fact that a lick has occurred so that the Motor/Valve Arduino could properly respond.

So I became interested in the idea of simplifying this setup. It was just a test at first, but it quickly became clear that this setup would perform similar to, if not better than the previous setup. With the added benefit of being able to easily collect and send lick, valve, **and** motor timestamp data in one place. This allowed for much easier communication of this data to the controller program.

The most difficult part of the process was creating a algorithm for detecting licks, opening valves, and marking closing times while prohibiting lick detection until this marking occured. The real complication here was that the algorithm could not *loop or* ***block*** the execution of the rest of the code. But once this was finished and various pieces of code were optimized for speed (e.g properly ordering command conditional statements) and testing simulated licks with an onset-to-end duration of as low as 15ms, it was clear that the board could handle these tasks all together. It is able to record and report these data points, handle door motor operation, valve operation, and lick detection all in the same board.

This allows for a simpler setup and slightly lower cost of the rig. But the real benefit here is a greater wealth of data and reduced complexity in communication protocols and physical setup.

For the rest of this article I will detail the steps you must take to migrate your setup from 2 Arduino boards to 1, including hardware and software steps.

2 The Migration Process

I've worked quite hard during this period of development to make distribution of both of the required software packages for the rig as simple and easy as possible. While the Arduino portion of the code is slightly more complex, I've tried to detail every step as simply and as verbose as possbile to ensure that anyone can do it. If you follow each step closely, you should be up and running in no time!

If you are someone who is savvy enough, you may use the KiCAD PCB design file, along with the new Arduino source code files to assist you during this process. These are **NOT** required to complete the migration, but they may be helpful if you get confused or make a misstep along the way.

If you have downloaded the Github repository locally on your computer, the KiCAD project file you are looking for will be located at '/Circuitry/Final_PCB.kicad_pro'. The Arduino Code is Located at '/Code/Arduino Code'. The root Arduino '.ino' file will be located here, with specific function code such as optical sensor pin definition and LED pin definition located under the respective directory inside of the /src directory.

2.1 Hardware Migration Steps

First, we will start with the hardware migration steps. Here you are going to eventually remove the Arduino board previously labelled 'Laser Arduino', leaving only the Arduino board previously labelled or known as the 'Motor Arduino'. If you don't know which is which, identification steps are given below.

2.1.1 Identify the 'Laser Arduino' Board

You are going to first identify which Arduino board is the 'Laser Arduino' in your setup. If the board is unhelpfully **not** labelled, it is the board with wires running to the PCB (circuit board) whose connectors are labelled 'Laser_Power_Pins1' and 'LED_Logic_Harness1'. One-by-one, we will migrate every connection here to the board previously known as the 'Motor Arduino'.

You **do not need to remove** the actual connectors on the PCB, that will only complicate the process unnecessarily.

2.1.2 Obtain a Small flat-head Screwdriver

Locate the small flat-head screws that sit atop the Arduino screw terminal boards shown in Figure 1 that secure the non-connector (loose wire) end to the Arduinos. Again, we will remove these wires **ONE AT A TIME** to simplify this process.

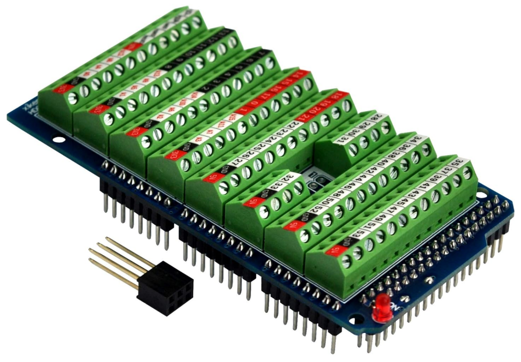


Figure 1: The Arduino Terminal Screws Sit atop the Arduino Board, and Face Away from You if You Look at the Arduino Head On.

2.1.3 Remove Unnecessary Connectors

Now we are going to remove the wires that connect to both Arduino boards that facilitate communication between them.

1. **Digital Pin 8** - We will begin with digital pin number 8 which functioned as the lick side indicator previously. This pin should be the same on both Arduino boards.
    (a) **Locate Digital Pin 8** - Locate the wire attached to digital pin 8 on the **'Laser Arduino' board**. It is the terminal with the big number 8 on it and the square will be black.
    (b) **Unscrew the Terminal** - Unscrew the terminal with your screwdriver and remove the wire. Follow the wire to the connection on the 'Motor Arduino' and repeat the process.

   (c) **Tighten the Terminal Screw** - Tighten the terminal screw back down on both Arduino screw terminal boards.

2. **Digital Pin 7** - Next, we will remove digital pin 7 which functioned previously as the lick detected signal. This pin is slightly different because it is **not the same** on both boards. It is pin 7 on the 'Laser Arduino', and is pin number 2 on the 'Motor Arduino'.

   (a) **Locate Digital Pin 7** - Locate the wire attached to digital pin 7 on the **'Laser Arduino' board**. It is the terminal with the big number 7 on it and is shaded black.

   (b) **Unscrew the Terminal** - Unscrew the terminal with your screwdriver and remove the wire. Follow the wire to the connection on the 'Motor Arduino' and repeat the process.

   (c) **Tighten the Terminal Screw** - Tighten the terminal screw back down on both Arduino screw terminal boards.

2.1.4 Migrating the Remaining PCB Connectors

Now that we have removed the inter-Arduino communication connections, we just need to migrate the 'Laser Arduino' PCB connector wires to the 'Motor Arduino'. As dicussed above, these include the below connections from the PCB. As before, we will move wires one-by-one starting at the top of the list and moving downwards.

1. **Laser_Power_Pins1**
   This connector is the easier of the two to migrate. It is just two connectors, 5V and GND.

   (a) **Locate the 5V Wire** - This wire should have red tape wrapped around it and run to a 5V terminal on the 'Laser Arduino' board.

   (b) **Remove the Wire from Screw Terminal** - Unscrew the terminal screw that this wire runs to on the 'Laser Arduino', and remove the wire from the terminal.

   (c) **Locate a Free (unused) 5V Terminal on 'Motor Arduino'** - Find a free 5V slot on the 'Motor Arduino' and unscrew the terminal screw.

   (d) **Secure the Wire into the Terminal** - This step is likely to be one of the more challenging parts of the migration. The wire may not reach because it was not originally designed to stretch that far. You may have to play with the location of the 'Motor Arduino' board or solder extra wire to get it to reach.

   Once you have situated the boards such that all wires reach their terminals, insert the wire into the open terminal and fasten the wire by tightening the screw until the wire is secure and will not pull from its terminal.

NOTE: You should pinch the wire with your finger and twist the end around until all of the wire is bundled together cleanly to avoid the wire touching another terminal or not fitting into the terminal well. You should do this every time you insert a wire from this point forward.

(e) **Locate and Unscrew the Terminal for the GND Wire** - Now, take the other wire from the same connector (the one not wrapped in red tape) and remove it from its screw terminal on the 'Laser Arduino'.

(f) **Locate a Free GND Terminal on the 'Motor Arduino'** - Locate a open GND Terminal on the 'Motor Arduino', unscrew the terminal, insert the wire, and tighten the screw until the wire is secure and will not pull from its terminal.

2. **LED_Logic_Harness1**
   Now we are almost done with the hardware migration process! We have just the 4 wires on the LED_Logic_Harness1 connector left to go. These wires are easier to differentiate as they are each a different color.

   (a) **Locate the ORANGE Wire** - This wire produces the logical output of the optical sensor for spout 1. Looking at the connector plugged into the PCB from which these wires originate, ORANGE lies directly right of YELLOW and above BROWN.

   When I speak of 'left', 'above', 'below', or 'right' moving forward, this is what I am referring to.

   (b) **Remove the Wire from Screw Terminal** - Follow the wire to its terminal on the 'Laser Arduino', unscrew the terminal screw that this wire runs to, and remove the wire from the terminal.

   (c) **Locate Digital Pin 49 On the 'Motor Arduino'** - The pin that we will relocate this wire to is the 'Motor Arduino' terminal with the big white number 49 on it.

   (d) **Secure the ORANGE wire to Digital Pin 49 On the 'Motor Arduino'** - Unscrew the screw terminal, insert the wire, and tighten the screw until the wire is secure and will not pull from its terminal.

   (e) **Locate the YELLOW Wire** - This wire produces the logical output of the optical sensor for spout 2. It lies directly left of ORANGE and above RED.

   (f) **Remove the Wire from Screw Terminal** - Follow the wire to its terminal on the 'Laser Arduino', unscrew the terminal screw that this wire runs to, and remove the wire from the terminal.

   (g) **Locate Digital Pin 47 On the 'Motor Arduino'** - The pin that we will relocate this wire to is the 'Motor Arduino' terminal with the big white 47 on it.

(h) **Secure the YELLOW wire to Digital Pin 47 On the 'Motor Arduino'** - Unscrew the screw terminal, insert the wire, and tighten the screw until the wire is secure and will not pull from its terminal.

(i) **Locate the BROWN Wire** - This wire produces the LED 'Spout Licked' indicator spout 1. It lies just below the ORANGE wire and and right of RED.

(j) **Remove the Wire from Screw Terminal** - Follow the wire to its terminal on the 'Laser Arduino', unscrew the terminal screw that this wire runs to, and remove the wire from the terminal.

(k) **Locate Digital Pin 48 On the 'Motor Arduino'** - The pin that we will relocate this wire to is the 'Motor Arduino' terminal with the big white 48 on it.

(l) **Secure the BROWN wire to Digital Pin 48 On the 'Motor Arduino'** - Unscrew the screw terminal, insert the wire, and tighten the screw until the wire is secure and will not pull from its terminal.

(m) **Locate the RED Wire** - This wire produces the LED 'Spout Licked' indicator spout 2. It lies just below the YELLOW wire and and left of BROWN.

(n) **Remove the Wire from Screw Terminal** - Follow the wire to its terminal on the 'Laser Arduino', unscrew the terminal screw that this wire runs to, and remove the wire from the terminal.

(o) **Locate Digital Pin 46 On the 'Motor Arduino'** - The pin that we will relocate this wire to is the 'Motor Arduino' terminal with the big white 46 on it.

(p) **Secure the RED wire to Digital Pin 46 On the 'Motor Arduino'** - Unscrew the screw terminal, insert the wire, and tighten the screw until the wire is secure and will not pull from its terminal.

2.1.5 Remove the 'Laser Arduino' board from the Setup

Remove the board which you have removed all connections from, and disconnect its USB cable from the PC. You can set this Arduino aside as a backup Arduino board in case anything ever happens to the one currently in use.

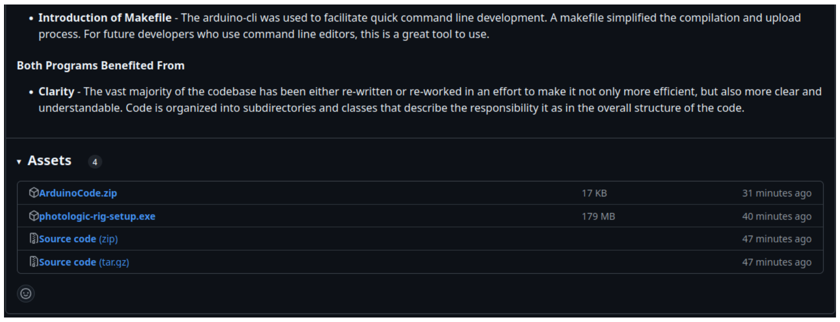
2.2 Software Migration Steps

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

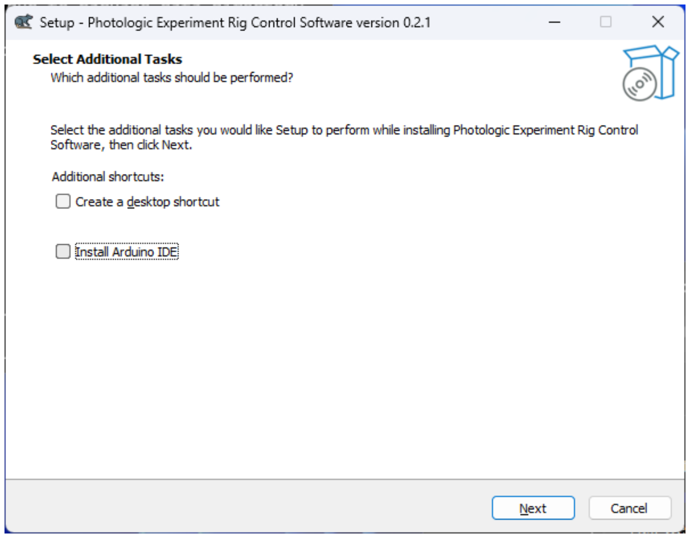
- **Clarity** - The vast majority of the codebase has been either re-written or re-worked in an effort to make it not only more efficient, but also more clear and understandable. Code is organized into subdirectories and classes that describe the responsibility it as in the overall structure of the code.

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. 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

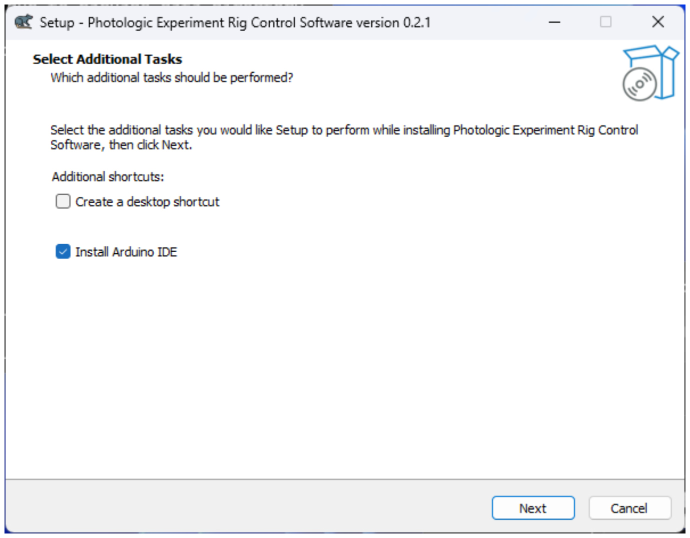
Which additional tasks should be performed?

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?

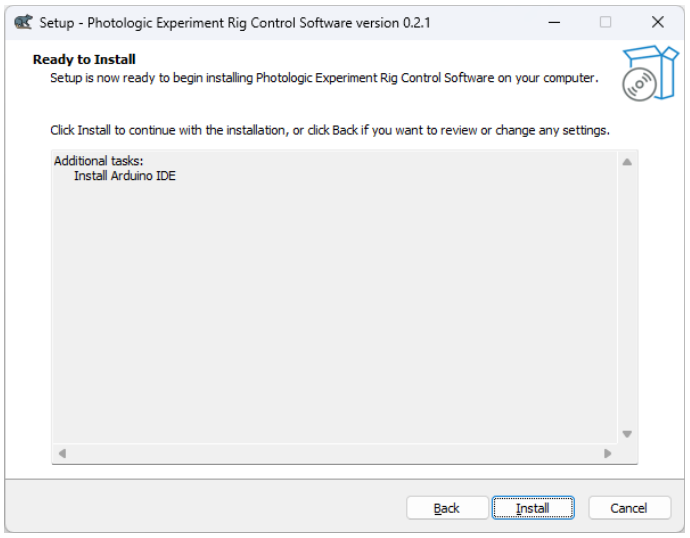
Select the additional tasks you would like Setup to perform while installing Photologic Experiment Rig Control Software, then click Next.

Additional shortcuts:

- [ ] Create a desktop shortcut
- [x] 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.

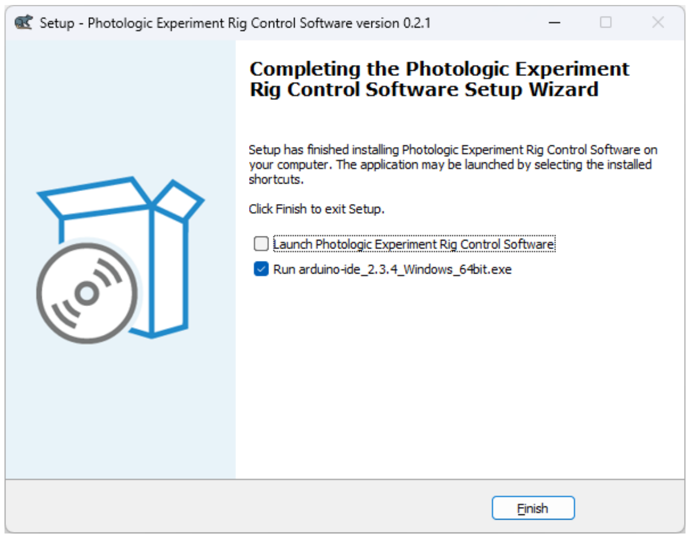
Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
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

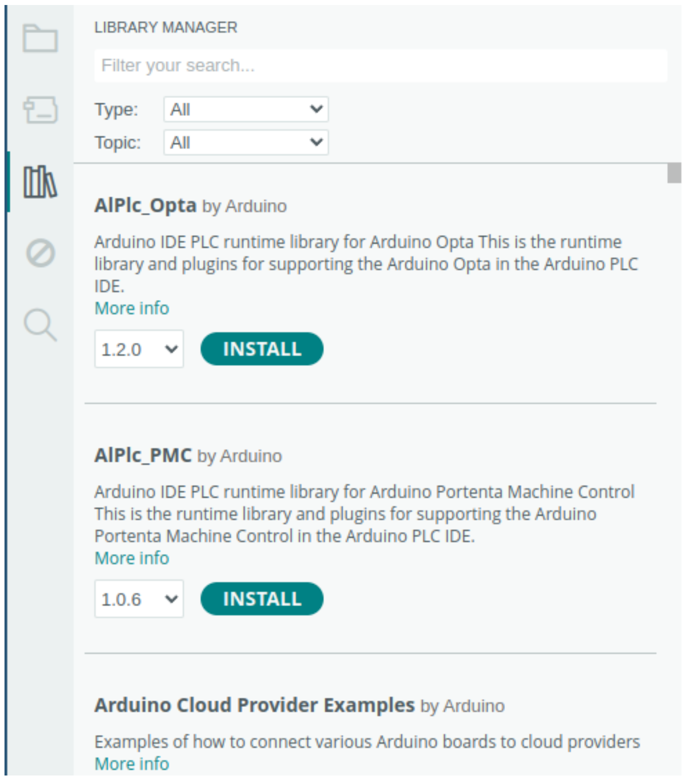
Figure 6: Do NOT Run the Control Software, but DO Run the Arduino Installer.

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

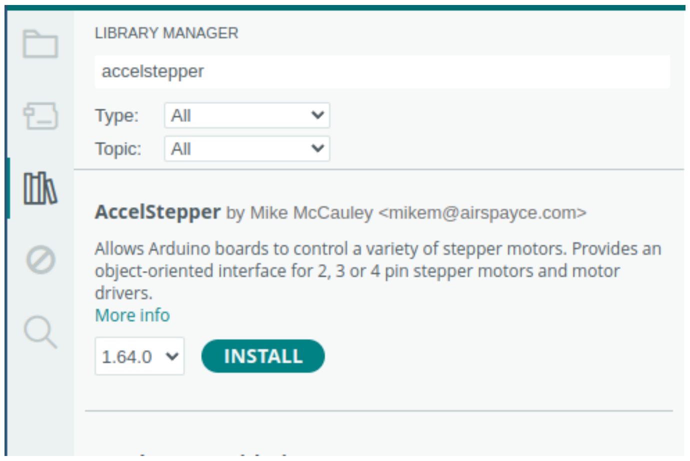
Arduino Cloud Provider Examples by Arduino

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>

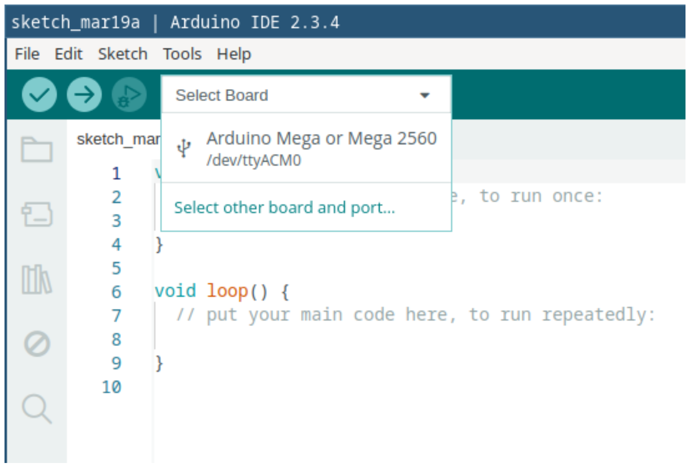
Allows Arduino boards to control a variety of stepper motors. Provides an object-oriented interface for 2, 3 or 4 pin stepper motors and motor drivers.

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
sketch_mar

Arduino Mega or Mega 2560
/dev/ttyACM0

Select other board and port...

```
v
                                e, to run once:

}

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

}
```

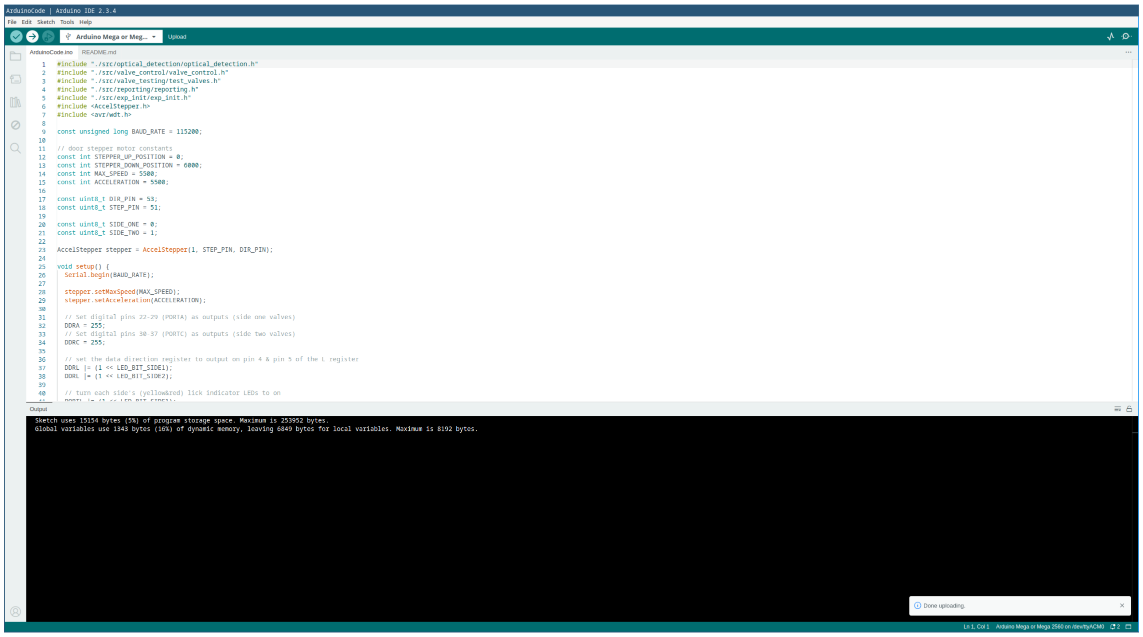
Figure 9: Select Your Arduino in the 'Select Board' Menu.

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

```
#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 Verifying Functionality

At this stage all migration steps are complete. All that we need to do is make sure that everything functions as expected. This guide is going to assume that you have your Arduino board plugged into your PC, the door motor is in the 'UP' position, and the power supply for the rig is turned on. Please take these steps before following the below guide.

3.1 Open the Control Software

Open the control software by beginning to type 'Photologic Experiment Rig Control Software' in your start search menu. It should quickly appear. It has a cartoon rat icon as its program icon. Click on this entry to open the software.

3.2 Test that Valves Open and Close Properly

First, you will need to test that all valves are being recognized. To test that each valve is being recognized properly by the software, follow the steps below:

1. Open the valve control window by pressing the 'Valve Control' button on the main screen.
2. Next, press each button one-by-one and ensure that each LED light corresponding to a valve is properly toggled on and that you hear a valve switch on at the same time. (Figure 11.)

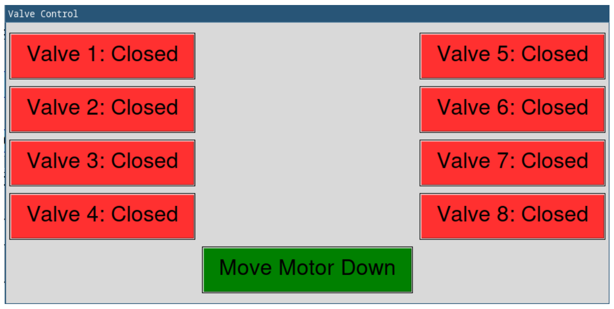


Figure 11: Toggle Valves On and Off by Pressing each Button Once.

3. If you get this feedback for each valve, and each valve properly turns off again when clicked a second time, valve control is confirmed.

4. If not all valves are being recognized, refer to Section 4 for troubleshooting details.

3.3 Test the Door Motor

Next, we will test to ensure that the door opens and closes as expected.

1. As before, open the 'Valve Control' window (Figure 11).
2. Click the green button that reads 'Move Motor Down'. This will send a signal to the Arduino that the motor should move to the down position.
3. Once the motor has moved down, wait a moment, then push the blue 'Move Motor Up' button.
4. Repeat the process a few times to ensure no issues arise after a few pushes of the button.

   NOTE: If the button is pushed again **during** a movement cycle, this will interfere with the Arduinos' sense of where the door is. At this stage you will need to power off the power supply, move the door back up by hand, power on the supply, then reset the Arduino by hitting the physical reset button next to the USB port on the board.
5. If any errors were experienced, refer to Section 4 for troubleshooting details.

3.4 Test Run an Experiment

If the rig has passed all tests thus far, you are ready to test run an experiment. Follow the steps below to test this functionality. This assumes the door is in the 'UP' position and the power supply is on. You will also need a long thin object to run between the optical lick sensors.

1. **Select Your Experiment Settings** - First, select the parameters of the experiment such as ITI TTC, Sample, and random +/- times, along with number of valves in use and number of trial blocks.
2. Then, label the names of the stimuli inside of your cylinders in the 'Valve/Stimuli' window.
3. Submit these changes by pressing 'Generate Schedule'.
4. You can now start an experiment by pressing 'Start' in the main program window.

5. Wait the duration of the first ITI, then during the TTC time run your object between the lick sensor on one side. If all is working you should see the lick indicator LED blinking for that side.

6. Repeat this 3 times to jump to a Sample time state and observe if valves begin to actuate with lick onset.

7. Repeat this process for all trials, or if the program is clearly functioning correctly, press 'Stop' in the main program window.

8. If any errors were experienced, refer to Section 4 for troubleshooting details.

3.5 Test a Valve Test Procedure

To test the valve testing / calibration functionality, follow the steps listed below. I will assume that you have filled the cylinders with liquid, have left the tops off, and have placed cylinders under the side(s) with valves to be tested.

1. Press the button on the home screen that reads 'Valve Testing / Prime Valves'.

2. Enter the amount of microliters that you would like valves to dispense **on every opening**. Then, enter how many times you would like the valve to actuate. This is **variable** for convenience, but higher numbers will result in greater accuracy. Calculations will adjust around this number.

3. Select which valves you would like to test by pressing their respective buttons below the table listing the current durations for each valve. This can be any combination. It can be 1 valve or 8, valves on both sides or just one. Any combination except for 0 works.

4. Now you're ready to start the test procedure, press 'Start Testing'.

5. In the confirmation window, review the valves to be tested and the amount of times each valve will open. If this is okay, press 'Yes' to start the procedure.

6. Wait for the first set of valves to complete. If there are valves remaining to test on each side, both sides will test. If there is only one side with a valve test remaining, only that side will be tested.

7. When the test has ended, you will get either one or two pop-up windows asking the volume dispensed for each valve tested (in mL) in this round of tests. **PAY ATTENTION TO THE VALVE NUMBER**, it is really easy to mess this up if you look at the **valve test TABLE** in the main window rather than the pop-up windows specified valve. Find and

enter this volume and press ‘OK’. If you press ‘Cancel’ here, the testing sequence will be terminated. You can restart from there if you like or quit testing altogether.

8. Repeat this process for as many valves as you selected to test.

9. When all valves have completed, you will see a popup window that displays the previous duration and the duration that you are moving to based on testing. If these changes look good, press the green ‘Confirm Changes’ button. If they are not satisfactory, press the red ‘ABORT CHANGES’ button.

10. The durations that you had saved previously are now saved in the newest archive duration ‘profile’. These can be found and re-applied if desired in the ‘Manual Valve Duration Override’ window from the main ‘Valve Testing’ window.

11. If any errors were experienced, refer to Section 4 for troubleshooting details.

3.6 Test a Valve Prime Procedure

Testing a valve prime procedure is similar to, but even easier than the testing procedure described above. Follow the steps below to verify this functionality.

1. As before, open the ‘Valve Testing / Prime Valves’ window, but this time push the orange ‘Switch to Priming Mode’ button at the top of the window.

2. Enter how many times you would like each valve to actuate.

3. Select which valves you would like to prime. Again, this can be any combination. It can be 1 valve or 8, valves on both sides or just one. Any combination except for 0 works. Push the buttons that correspond to the valves you want to test.

4. Start the procedure by pressing the orange ‘Start Priming’ button at the bottom of the window.

5. Confirm the valves to test, and the number of actuations in the pop-up window by pressing ‘Yes’.

6. Wait until the prime is finished, or, if you’ve had enough at any time you can press the yellow ‘STOP PRIMING’ button. You do not have to do anything special to restart the procedure. You can stop and restart as many times as you wish. If you started but found you actually want different settings, that option is available.

7. Once the valves stop actuating, the procedure is finished and functionality is confirmed.
8. If any errors were experienced, refer to Section 4 for troubleshooting details.

4 Troubleshooting

4.1 First, the Golden Rule

As is true in almost all of electronics, if it is not working, just turning it all the way off and back on again is the best first step in debugging odd infrequent issues. This is certainly true here. Control software suddenly not working? Close the main window by pressing the red X and try it again. Arduino not opening valves or moving the motor? Press the red restart button on the Arduino, close the control software and restart and try it again. None of that worked? Restart the PC and see if that helps.

The point is, if you see an error that doesn't come up often, a restart will generally fix it. If problems persist after taking these actions a few times, try the below steps or get into contact with me at `jbhour01@louisville.edu`.

4.2 General Arduino Troubleshooting Steps

The most common error that you are likely to experience is an error in communication with the Arduino board for one reason or another. This error will show itself in unresponsive valves, door motor, and/or optical detectors. Sometimes the Arduino may get stuck in a loop or have corrupted memory unexpectedly that will cause these or other symptoms. You may get an 'error sending durations', or 'error receiving schedule' in the controller program.

Other errors you may get are less likely to be resolved by taking the below steps, but you can always try them.

- **Restart the Arduino Board -** As stated above, usually the easiest and most effective solution is to just restart the Arduino and controller program. Close the controller program if it is open, push the red button on the Arduino board, then reopen the controller program.
- **Check the USB Connection between the PC and the Arduino -** It is always a good idea to check the physical connection between the PC and the Arduino. Unplugging the Board at every junction and plugging

everything back in may resolve whatever communication issue you are experiencing.

- **Reupload the Arduino Software** - Furthermore, if neither of the previous steps help you, sometimes the board just needs its software reuploaded. Follow the steps described above in section 2.2.2 to reinstall the software to the board.
- **Check PCB Connections** - It is always a good idea to make sure that there are NO loose connections on the Arduino board. Pull on them lightly and see if any of them move. If they do, try re-inserting them and tightening the screw.
- **Try the Other Arduino** - Finally, if you problem is Arduino related it could be your individual board. You can remove the terminal screw 'hat' that sits on top of the Arduino board, transfer it to another Arduino Mega 2560 and upload the software to it (Section 2.2.2).

If none of these issues fix your Arduino problem, email be at `jbhour01@louisville.edu`.

4.3 What to do if Valves do not Open as Expected?

If you find after testing each valve individually that the valves (particularly side two) are not opening as expected, and the Arduino troubleshooting steps did not help you please send the following information along with a problem desciption to my email.

- Document which (if any) valves do open correctly when you press the buttons in the control software.
- Photograph the connections between the PCB and the Arduino board. The connection I am most interested in here will be the largest connector on the board with 16 connetions. If you could get a good photograph of this setup and what pins each wire runs to on the screw terminal board, this should be a simple issue to debug quickly.
- Logfiles. Your logfiles will be in your documents folder, under 'Photologic-Experiment-Rig-Files'. Please locate the most recent logfiles and attach them to the email.

4.4 What to do if the Door Motor does not Move as Expected?

If you find that the door motor does not work at all, but other functions such as valve opening DO work, first make certain that there are **NO** loose connections on the Arduino board. Pull on them lightly and see if any of them move. If they do, try re-inserting them and tightening the screw.

If none of these things or the above Arduino troubleshooting steps help and you've restarted the program a few times to no avail, email me at `jbhour01@louisville.edu`. I will look into the issue and try to get back to you with a fix as quickly as possible.

4.5 Other Issues

For other issues that persist despite your best efforts to troubleshoot, please email me at `jbhour01@louisville.edu`. I will look into the issue and try to get back to you with a fix as quickly as possible.

5 In Closing

I hope that this guide makes the transition as smooth as possible. I tested this at my house and everything worked on multiple PCs, so I hope this continues to your lab. I am currently working on a full user guide that will teach you how to use every part of the software in detail. I am also working towards code documentation that will make future development easier.

Any questions you may have would be best asked in the near future because everything about this project is fresh in my mind right now. Once again, please send any questions or concerns that you have to my email at `jbhour01@louisville.edu`.

Blake Hourigan