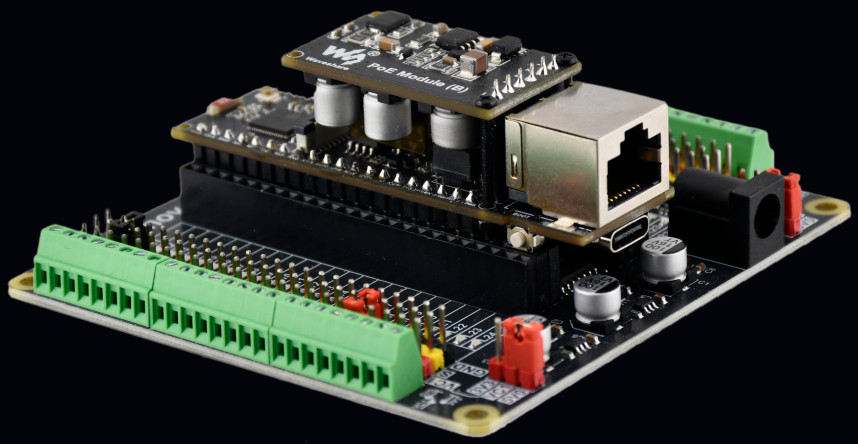
Nexus Ultimate Prop

Controller User Manual

Configuration, Game Modes, Lighting, and Operator Guide

v 1.0



nexusworkshop.com/propcontroller info@nexusworkshop.com

Contents

Introduction & First Connection

What is The Nexus Ultimate Prop Controller?

The prop controller. The controller. It goes by a lot of different names and it does many more different types of jobs. Ultimately, it is the culmination of over a decade of building Escape Rooms and Immersive Experiences with almost a decade before that of theatrical technical design. It is every prop we ever wanted to make and it is every tool we wish we had to do it with, and beyond that, it is ever growing and expanding.

That said, its goal is to be user friendly and open doors to allow users at all levels of technical experience to get more out of their creative endeavors. We've tried to give it the power and flexibility to never be the weak link but at the same time to allow quick and easy options to take away the low level programming that can become repetitive prop to prop and for many creators is the biggest hurdle between what they've envisioned in their head and what makes it into the real world.

Throughout this manual you will find a lot of thoughts behind how to accomplish similar goals and what you will quickly notice is how many different ways there are to go about any given challenge. Of course, if you aren't finding exactly the right fix for your setup you should reach out to our Facebook community or message us directly and we can see if there is a built-in solution for it today, or if it's simply another new tool that needs to be put into the system.

With that let's dive in!

Note: There are affiliate links within this document to modules that we frequently use in conjunction with the prop controller. As an Amazon Associate we do earn commission if you use these links to make a purchase at no extra cost to you. We ask that if you are purchasing these components you use these links, every bit helps, we greatly appreciate your continued support.

> HOW TO USE THIS MANUAL The breadcrumb callouts show the shortest path through the web UI. The step callouts are the practical checklist to follow on a real controller. Tip, warning, and checkpoint callouts are there for the places where the controller is doing something important behind the scenes.

The Physical Controller

The default setup for the controller is an ESP32-S3-ETH dev board which includes a built-in ethernet connection, Power over Ethernet(PoE) capability and an RGB LED that we use for identification and status functionality.

This board is placed in a terminal breakout board that can be powered from 12V and provides 5V and 3.3V regulators on board, screw terminals for all available pins and some extra pins for voltage and ground rails if you want to use those.

There are options to get a WiFi only version, use it without the breakout board (soldering to the pins instead) and options that include a housing and fixed terminal connections for each configured module.

If you purchase the board as part of a ready-to-go prop from us it will have a housing and dedicated connectors for each thing that it connects to but many of you will be purchasing the controller itself and connecting various I/O modules to it ad hoc.

The controller is designed to be an open-hardware system. What that means is that there is no "default" configuration on inputs and outputs. Instead there are 12 available pins that can be configured as:

- Analog Inputs (potentiometers, analog sensors)
- PWM Outputs (pins that control power by rapidly pulsing on and off, e.g.. servos, transistor modules)
- Digital Inputs (buttons, reed switches, toggle switches)

- Digital Outputs (relays and the like)
- WS2812b addressable pixel outputs
- I2C busses (shared two-wire communication lines for I/O expanders, color sensors, displays)
- SPI busses (fast communication lines used by RFID readers, other I/O modules)
- Serial connections (send and receive data over dedicated TX and RX pins ie. DMX outputs, Sound modules)

Being an open-hardware system means that there are many, many more options and ways to configure than most controllers you may be used to and there will also be the possibility of hitting performance limits before technically running out of pins (especially if using lots of expanders or heavy modules). If you ever have a question about limitations let us know and we will attempt to get a solid answer for you, testing directly if possible.

Note that not all pins are available for use. The board has several pins that have special functions, in addition to the ones that would appear in a web search GPIO17 (general-purpose input/output, Physical pin on the controller) has a special firmware function on the controller and should never be connected to anything. In the configuration it will always show you the available pins and we'd suggest starting there building out your modules then use the Board Pinout feature to give you a quick one-sheet to connect up your project.

> **WARNING** Do not wire GPIO17 to a prop input, output, button, sensor, or expansion board. GPIO17 is reserved by firmware for last-resort factory reset behavior.

As a general note on the controller the pins are 3.3v logic and 5v tolerant, if you are working with 12v things connected to input pins it's important to use optocouplers on your inputs to protect them.

> **TIP** Treat the Board Pinout screen as the wiring source of truth after you finish I/O setup. It reflects reserved pins, shared buses, and module pin assignments from the saved configuration.

If you buy a prop from us with digital inputs it will have these already connected. While you won't need to write any code yourself, the connections and modules you will find yourself using are the same ones you'll find lots of examples for in the Arduino/ESP32/Pi communities.

How to get help

Along with this manual there are a few key resources that you'll want to keep close at hand. Within the system's UI you'll find tool tips pointing you to which pages and sections handle what elements and reminders for some of the key terms. We have a Facebook community that is always available to post questions and get advice both for direct technical issues as well as the underlying logic or approach behind how to set it up.

On our website you'll also find links to a couple different YouTube playlists where we have a full video guide walking through the interface and connections, basically a video version of this manual. Another playlist we have on there has a series of prop examples showcases many of the different features and ways you can use the controller. Every example game you see on our channel will also be found in the prop wizard so you can immediately play with it, use it as a starting point or set it and walk away.

And of course we are here to help, we can answer technical questions and if it's something that would help you get going faster we offer remote configuration where we will hop on a call with you to go over your setup and what behaviors you need, then we will build out the actual configuration and help you get it loaded up so you don't need to do anything. We offer this as a 30 minute session directly from our web store but of course if you need more time or a deeper dive, training, or anything else relating to Escape Rooms we are here to help and can get you a custom quote.

Connecting to the system

The controller is configured through the network (either WiFi or hardwired Ethernet). Out of the box if you plug it in to a 12v power source (or USB to the board directly) it will turn on and start broadcasting its own network that looks like **nexuscontroller-xxxx**. You can join this network with your device and it will walk you through an initial setup.

Give it a name > Connect it to a network > Run the Prop Wizard

STEPS 1. Power the controller from 12V, PoE, or USB. 2. Join the WiFi network named **nexuscontroller-xxxx**. 3. If the setup page does not open automatically, browse to **192.168.4.1**. 4. Give the controller a clear device name, connect it to your room network if needed, then choose a Prop Wizard preset or manual configuration.

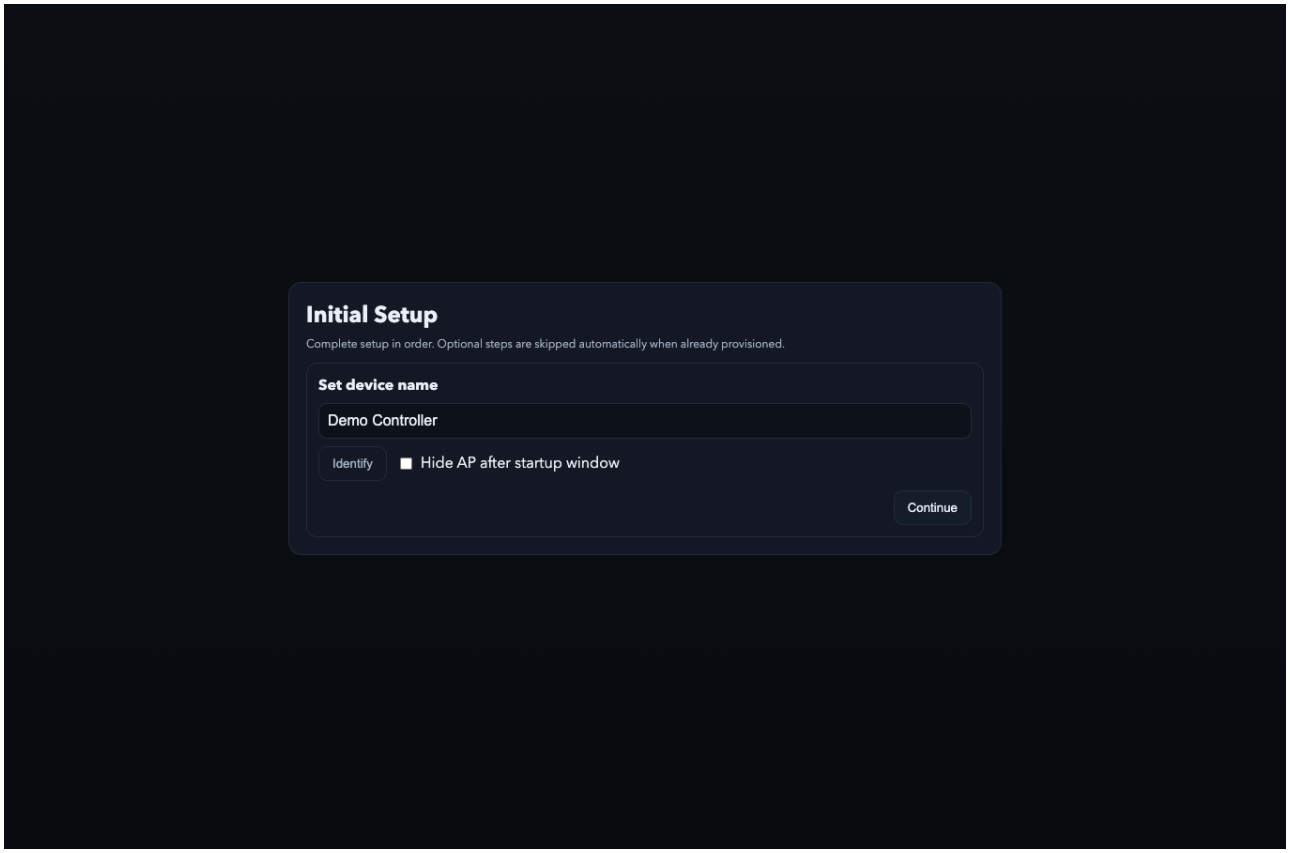


Figure 1: Initial setup screen.

Each of these steps is somewhat optional. You could keep the existing name (though best practice is to use something more meaningful to its purpose/prop). You don't ever need to connect it to a network if you are planning to just configure and let it run (though we think the added functionality, monitoring and control are a huge benefit). Within the Prop Wizard you can choose a predefined configuration or select manually configure.

When you connect to its built-in WiFi network, the configuration screen may open automatically. If it doesn't, open a browser and navigate to **192.168.4.1**. This should take you to the correct page.

If you hardwire the controller to your network via Ethernet it will skip the network question in the initial setup.

Once you have a controller on your network you can open a browser (Google Chrome is the preferred and most tested one for this) and navigate to **nexusProp.local** where "nexusProp" is the name you gave to your device. This UI is definitely easiest to navigate on a full desktop style browser but you should be able to get around using a tablet or phone browser as well.

CHECKPOINT Before mounting the controller inside a prop, confirm that you can reach it by name, open the Operator Page, open Configuration, and trigger Identify so the onboard RGB LED flashes on the physical board you expect.

The Nexus Controller Manager app

We hope that this controller proves useful enough that you want to put lots of them all throughout your room(s) running different props and part of the overall experience, and we know that managing large sets of things can get tedious. So we created an app that can be installed for Mac or PC and

allows management of your entire fleet. We will have a chapter going over the app in full later but for now know that it is a way to automatically discover all of the prop controllers on your network (in case you forget what you named them), easily open the UI for any of them, update them to the latest firmware, and send us logs/config files if you ever need support.

The Configuration UI

Logins

Whenever you first connect to/open the UI you will find yourself on the Operator page. This page is not secured, rather your network itself is assumed to be secure. We will have a chapter for this page a bit later after we have gone through configuration. At the bottom of the page though is a link to Open Configuration.

If you setup an admin password during initial setup you will be prompted for it to get in. Later you'll have the ability to change this password. If you lose this password there are a couple ways to reset the board but you will lose your configuration so make sure to remember this password and keep it somewhere safe.

System Features

There are a few buttons that are at the top of each configuration page. System Setup, Board Pinout and Identify. Identify is the simplest of these options, when you click it the built-in LED on the controller will start flashing. You can use this to make sure the physical controller is the one you think you are working on and also to confirm that the connection is working and that the board is responding.

> **BREADCRUMB** Operator Page > Open Configuration > top toolbar: **System Setup**, **Pre-Compile**, **Board Pinout**, **Identify**

The next button is the Board Pinout, this feature gives you a single view at every physical connection to the board so you can quickly get all of your configured modules connected. It won't show you power and ground connections but it does show you every pin that is configured including the I2C/SPI bus pins. Also on this list you will see GPIO17 listed with a reminder that you should never connect anything to it. This pin is used in firmware as a factory reset if all else fails and will completely wipe the board. Leave it disconnected, don't use it. The Board Pinout has a handy print button that can let you print out a clean one-pager with all these connections to take to the workbench, and the device name of the controller shows up in the top left.

The last button up here is the biggest of the three and that is the System Setup option. In here you will see the ability to import and export JSON configurations, this allows you to backup and share controller configuration quickly between your controllers, your users and even with the community. If you have a 3rd party (like us or others) build out a configuration for you this is where you'll go to load it into your board's memory.

You can select which sections to import/export though most of the time you'll be using this as an all or nothing thing. There are options to check your firmware version, and update firmware. If you choose the Update Firmware option it will need the internet and will try to automatically pull the latest version from us and update over the network. This is one of the only features that needs internet access versus simply having internal network access. If you don't have internet connected to your control network, are using the device's on-board network/access point or have gotten a special firmware build from us then you can use the Local Update feature and that will prompt for a file on your local computer/device you are connecting with to use.

Also within this System Setup window you have the ability to do a complete factory reset (wipe everything) or to wipe the configuration and keep the device network credentials so you can immediately get to programming your controller again. The last button in this window is the Prop Wizard, this will also reset your configuration but keep the network credentials and send you back through the Prop Wizard setup to start fresh.

When you do a reset action it will first prompt you to download the current configuration, we would suggest you do this just in case you need to go back or re-load something. If you accept it will simply download the full JSON config to your local machine.

> **TIP** Use **Pre-Compile** before a show when you have made behavior changes. It warms up the generated action logic so the first live trigger does not have to do that work in the moment.

Connections

This page largely handles the device level settings like its name and admin password, the modules you are physically connecting to it and the integrations which represent external systems you are communicating with.

> **BREADCRUMB** Operator Page > Open Configuration > **Connections**

Connection & Identity

This section starts with the Network Settings. Those include a Device Name which should not have any spaces and will end up as the DNS name (a network name that you can type in place of the numeric IP address) that can be used to open the UI in a browser.

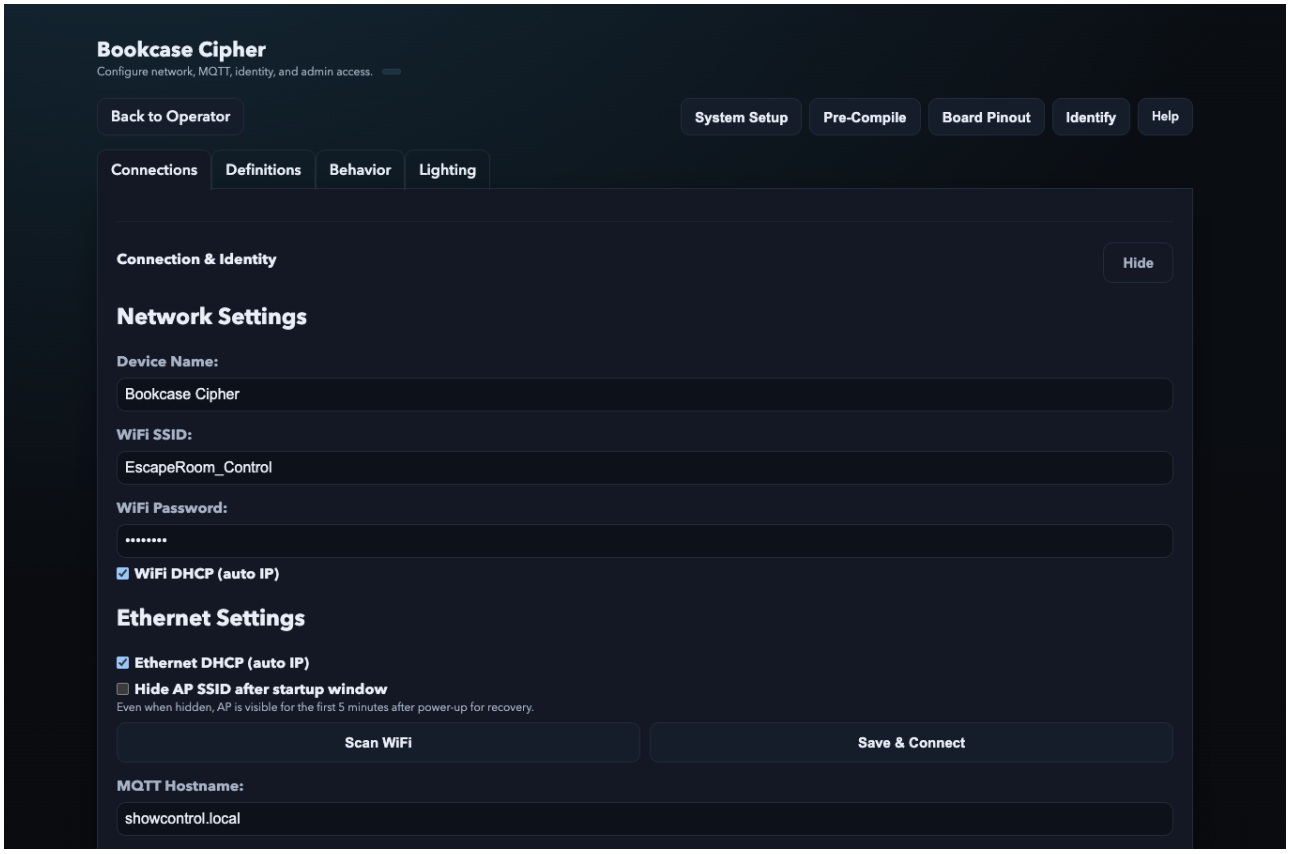


Figure 2: Connection and Identity settings.

The DNS name can also be used to send network based commands from external systems and will be the name of the built-in network if the device ever can't connect to a network or doesn't have one configured.

Below that are the WiFi SSID and Password fields if you would like to connect to your network wirelessly. There is also a checkbox to allow the WiFi connection to use DHCP (automatic network addressing) from your network router, if you uncheck this you can set a static IP (fixed network address that doesn't change) but when given the option using the hostname (devices network name) or setting an IP reservation in your router is usually a better bet.

The next sub-section is Ethernet Settings. If Ethernet is able to connect it will always trump WiFi automatically. Normally there is nothing to configure here for Ethernet but you can uncheck the box to set a static IP if you don't want to use DHCP.

There is also a checkbox here to Hide AP SSID (the name of the WiFi network created by the controller) after the startup window. Whenever the controller can't connect to a network (wirelessly or via Ethernet) it will turn its own AP (access point) on and start broadcasting a network so you can

reach it for configuration. This AP does not have a password on it and we don't want someone to come in and find it and start messing with your props.

If you plan to leave your controllers without a network connection (because you don't need all of those awesome features) then you should turn on this option, it will automatically hide that AP after 5 minutes of being powered on. This means you can always connect manually to the SSID that you know is there (the device name) or you can power cycle the controller and it will start broadcasting for another 5 minutes to let you get in and change what you need.

There is a Scan WiFi button to see available networks and select one (or you could have just typed the name in if you know it) and this is one of a few sections throughout the controller that has a manual save process. It will warn you when you try to navigate away with unsaved changes but this entire section needs you to click Save & Connect to save after which it will reboot the controller and automatically refresh the page. That is because its location can change with these network changes. If you change its name or network you will need to reconnect at the new location.

> **STEPS** 1. Set **Device Name** first. Avoid spaces because this becomes the local network name. 2. Choose WiFi, Ethernet, or both. Ethernet wins automatically when it is connected. 3. Leave DHCP enabled unless the network plan specifically requires static addressing. 4. Configure MQTT only if another system will subscribe to or send controller messages. 5. Click **Save & Connect**, wait for the reboot, then reconnect at the new name or address.

Also part of the Connection & Identity section are the MQTT Broker Settings (MQTT is a lightweight messaging system for devices & controllers, the Broker is the server that receives and distributes the messages) which include the MQTT Hostname, IP Address (the numeric network address of a device), Username and Password. The hostname and IP can be somewhat interchangeable based on your network and you may need one but if you can put both it can help with connection if one fails. Username and password are frequently not needed but if your MQTT broker uses these then you will need to enter them to be able to connect.

The Room Prefix (a text prefix added to a default address) is part of MQTT configuration and is used as part of the default topic address. It's also used by other integrations like OSC (a network control messaging format).

The last option in this section is the Admin Password which will secure this entire configuration page(s). It is important that you get this stored somewhere safe because losing it can mean losing your entire configuration when you need to reset to get back in. This password is security from people accidentally making changes to your configuration, it is not security from bad actors. The security for that is your network itself and keeping unauthorized users off of it. Do not use a shared or common password for this, we don't want to be the cause of a password to a more important or vulnerable system getting exposed.

> **WARNING** The Admin Password is not a substitute for a secure show network. Keep unauthorized devices off the network, and do not reuse a password from an email, router, business account, or other important system.

Simple UI Setup

The Simple UI is essentially a "Config Light" option. There is a ton of flexibility and options within the configuration of the Nexus Ultimate Prop Controller, there are also lots of props that will only need a small fraction of what's available. Think of the Simple UI as the configuration options you want to give to a power user rather than full admin. A lot of ready-to-go props you get directly from us will have a Simple UI setup that exposes only the controls that are needed for that specific prop. This will include things like the network credentials, sequence codes, etc, but it may not have the ability to add new modules.

When Simple Mode is enabled and you click to open the configuration page it will ask for a password. You can enter either the Simple UI password or the full Admin Password, but either way it will open to the Simple UI. From there there will be an Advanced button in the top right of the page, this will prompt for a password again and that one needs to be the Admin Password. Upon entering it you will be able to access the full configuration which includes the option to reconfigure or disable Simple UI as well as any and all settings that it was hiding. When you leave these pages it will lock itself back

up and require passwords to get back in again.

It is possible to leave the Simple UI Password blank in which case it would skip prompting for a password and let anyone that clicks straight in.

When you configure Simple Mode you can opt to select individual controls granularly or just to turn on/off complete sections of controls. Below that option you will find checkboxes for all of the available options to enable and disable. Remember that this doesn't impact those settings themselves, only their visibility when looking at the Simple UI configuration page. This section needs to be manually saved using the save button in the connections section.

> **TIP** Simple UI is best for room staff or power users who need to change codes, WiFi, or a small set of operator-facing options without exposing the entire controller configuration.

Inputs, Outputs, and Modules

This section is where you configure your physical connections to the board. It is another section with manual saving when you click Apply I/O Changes. This will reboot the board and refresh your connection to the UI automatically so it can apply the hardware pin changes in the back end. This is one of the first places you will start to see the system dynamically populating options based on what has been set up. An example of that is the Buses section right below the Add and Apply buttons. This section will populate with dropdowns to configure your I2C and SPI bus pins once you add any devices that will need them. There is also an I2C Scanner function that shows up here when any I2C devices have been configured. The scanner will show which addresses are found on the I2C bus and it will give a guess as to what it thinks they may be based on your configuration but it can't know for sure which device is which.

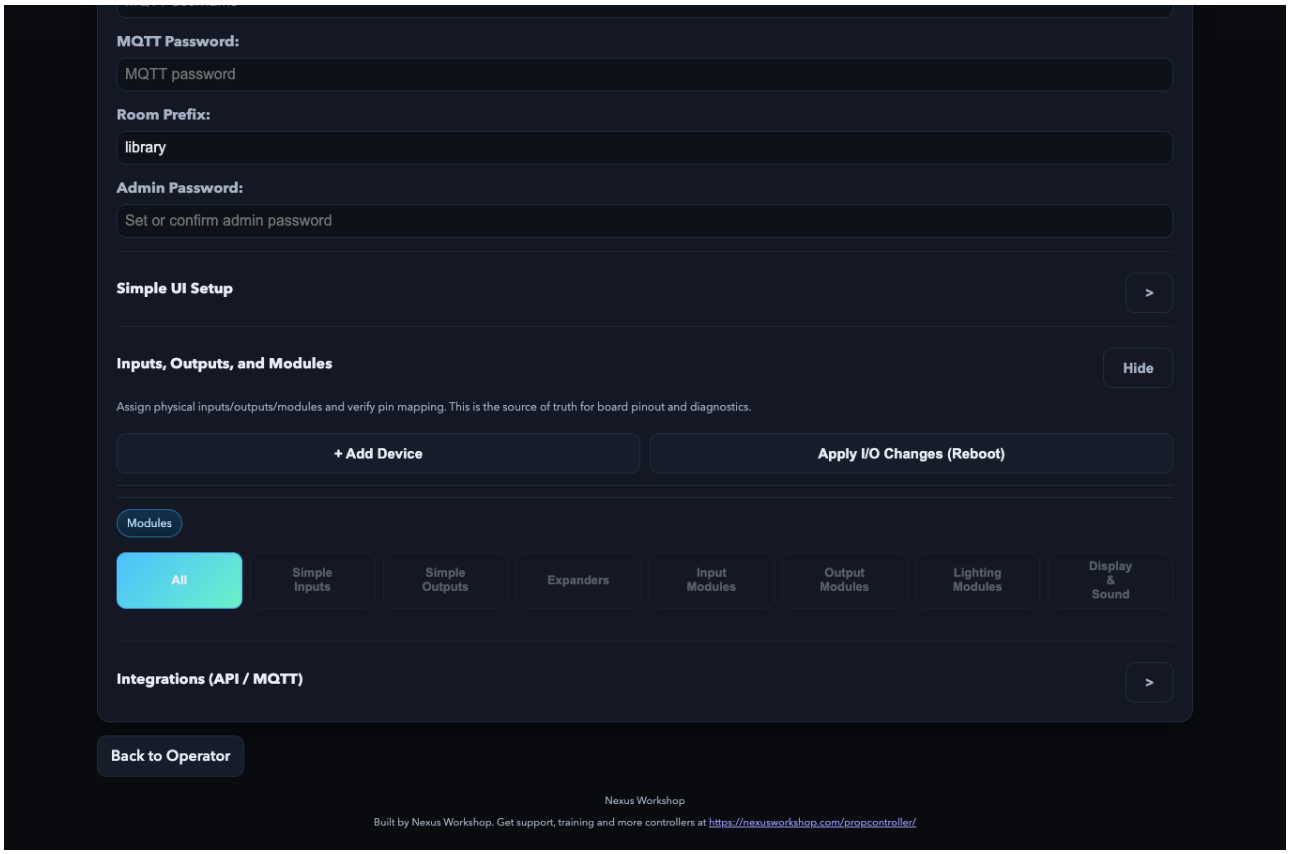


Figure 3: Inputs, outputs, and modules section.

There are a set of module filter buttons at the top of the list of configured modules. All is the default view but you can use these buttons to show only specific types of modules to quickly navigate as your prop gets more complex and you have more devices connected.

STEPS 1. Add expanders first if the prop uses them. 2. Add simple inputs and outputs next. 3. Add bus-based modules such as RFID, displays, audio, color sensors, or DMX. 4. Resolve pin conflicts before clicking **Apply I/O Changes**. 5. After the reboot, use the live rows and test buttons to confirm the physical wiring.

DATA-DRIVEN UI The module list drives later dropdowns throughout the controller. If a relay, input, pixel strip, RFID reader, or audio module is not defined here, it will not appear as a valid target in Behavior, Lighting, or the Operator Page.

Simple Inputs

The Add Device button gives you a table of available options broken down by category.

Simple Inputs have two states - on/off, 1/0, true/false or pressed/released. This includes simple buttons, toggle switches, reed switches as well as digital sensors like capacitive touch, motion sensors, and more.

The various options in the Simple Inputs category are there mostly to help match how the device is used and labeled.

When you add one of these devices you will be able to assign/change its Type, the Pin it is on, mark whether it is Active Low (or conversely active high) and apply the built-in Pull-up (a default high state that helps the input read reliably when idle). You can also apply your own On and Off labels. You will also be given a live view where you can (after save applies these changes) easily test the input and make sure it's doing what you expect.

A lot of this interface layout is shared for each module. You will have a name which can include spaces and is for you to remember what a device is. This name/ID will be what you choose in later dropdowns to target the input/module/device. You have delete and duplicate buttons and a drag handle to re-order your modules (which will reflect on the operator page). The pin dropdown will always show you only the available options (since some pins are reserved), it will show pins that are in use by another module but it will have a label there showing what is using it. If you try to assign a pin to 2 different jobs it will cause a conflict and prevent saving. The conflict will tell you where to look and fix it and if you really need to you can choose no connection to allow a save that you will finish working on later (obviously your module won't work with no connection).

You can also add I2C and analog mux expanders- modules that provide additional physical pins. Because these pins support a more limited set of functions, they will only appear for compatible modules. The system will automatically assign pins when you add devices and will prioritize those expander pins whenever it can to preserve your native on-board pins for more complicated devices.

Simple Outputs

Simple Outputs are things that also just have an on/off, this will mostly be Relays but you can control simple LEDs or whatever you'd like from them as well. With relays there are some additional features like a built-in flash mode to be used with solenoid latches. When flash mode is enabled the relay will never latch toggled on, instead an on or flash command will toggle for your Flash ms duration and will repeat if Flash reps is higher than 1. Outputs have test buttons on them like the Trigger button for Relays which will either toggle the current state or in flash mode will trigger the flash function.

Expander Modules

There are a few types of expanders supported, the first is the PCF8575 (Paid Link) which connects via I2C and allows adding 16 digital pins that can be used for input or output and are fully featured with internal pull-ups. The next is the TCA9548A I2C Multiplexer (Paid Link) which can combine isolated I2C busses for when you need to connect multiple devices that don't have selectable addresses and would cause a bus collision. The last expander is the CD74HC4067 (Paid Link) which creates additional analog inputs. When you configure one of these they will automatically populate as an option for pin selection on modules that can utilize them. We suggest you add your expanders before other modules because the system will automatically use their pins anytime it can while adding new devices but once the devices are there it won't re-evaluate moving them onto these expansion pins automatically.

Lighting Modules

The primary lighting module is WS2812b (Paid Link) addressable pixels, commonly referred to as NeoPixels because of Adafruit's excellent promotion of them to the community. This module doesn't have much to it because it has multiple entire sections later on to actually configure and use them. For now all you need to do is pick a pin and once applied you get a few test buttons here which will light all pixels on that pin (based on its later config). When you first add it will make a guess and assign 20 pixels to it but you can add many many more pixels to a single pin. You can create multiple pixel pins if you have huge qty needs or if it makes sense within your wiring to simplify things.

Output Modules

Each of these modules has a slightly different set of parameters based on their usage.

Linear Actuator Module This accepts 2 pins which can drive either standard relays or a dedicated linear actuator controller. You can select whether the relays/controllers are active high or low and you can define the Extend and Retract type. If you leave at none then it is relying only on timing to determine its location, if you choose active_low or active_high it will ask for a pin number for each of them to put limit switches at the extend and/or retract positions. For these limit switches when configured it will use these limit switches instead of the timing to determine when it has hit its stopping point in either/both directions. You can define the Default ms which is the time the relays should be active to travel the full distance if not using limit switches. There is a Preset dropdown which will populate this time with a best guess at how long it may take but really you can just time it and enter your number. There are text buttons to Extend, Stop and Retract.

Servo Controller Module This module will use PWM to control standard position based servos. You define a pin and then a Min and Max Pulse along with the Initial Angle you want it to go to on system boot. There is a preset option which can tweak its performance but you can set this manually from the manufacturer's specs on your servo. You may need to play with the min/max pulse especially if you need the servo to have hard stops without constantly pushing and fighting to move further. Test buttons provide a quick way to confirm everything is working as expected (but will only work after applying the IO changes)

Analog PWM Output Module This module allows you to drive generic PWM. This could be used for an analog VU style voltage meter, traditional LEDs with dimming or a transistor to control higher power loads. A pin is the only configuration and some simple test buttons confirm all is working correctly.

Input Modules

These modules are types of inputs and sensors that have more advanced communication or multiple pins compared to the simple inputs.

RFID Reader RC522 (Paid Link) The most ubiquitous RFID reader card for escape rooms and tinkerers alike. This communicates via SPI so you will configure your pins for that with the buses so on the actual model you will only need to configure CS (commonly labelled SDA) and RST pins. After saving in the Live section you will see feedback on the boards connection status, FW and Chip values. If these show a firmware version then you are all set up and your module is communicating successfully.

Color Sensor TCS3472 (Paid Link) These color sensors use the I2C bus to communicate and typically do not have a changeable I2C address so you will need an I2C multiplexer to connect multiple of them. They will report color data in the Live area including RGB values and a generated color number that you can process logic based on

Matrix Keypad (Paid Link) While it's a simpler module we put this one here because it uses a lot of pins and uses them in a different way than standard buttons. You can define the number of Rows and Cols then can configure which pin represents each.

Display & Sound Modules

Sound I2S + SD By adding a I2S DAC you can use the built-in SD card reader (on the bottom of the board) to store and playback WAV audio files. The best format is 16-bit PCM WAV 22050kHz but it's forgiving on the sample rate. There are modules available that have amps built-in as well as those with line-level outputs only.

7-Segment I2C Display (Paid Link) These displays typically have 4 characters and a number of decimals along with a ':' to provide a standard digital clock display. You just need to configure the I2C address and route (if using an expander). You get a simple test and clear function to make sure it's working.

I2C LCD These displays typically come in 16x2 and 20x4 (Paid Links) variants with an I2C backpack to control them. You can configure the columns and rows

along with the I2C address and route. You get a simple test and clear function to make sure it's working.

Integrations

This section is where we will define Systems that will let our controller communicate with other devices. This could be a room control system, a GM dashboard/timer app, QLab or other media control software or something like a network based relay board.

It's important to remember that particularly in this section the prop controller starts to feel like it could run everything but we feel that for most larger experiences these should be kept as an edge device with a central and more powerful system handling the overall game state and monitoring. As your game grows that more fine-tuned system will provide a better view into it and more granular control over it. That said there are a large number of problems that can be touched and fixed directly by the controller itself.

Once you add a System you will be able to give it an ID (name) and a Type. There are many types available including MQTT, HTTP (web-style request messaging), OSC and others. Each type has a slightly different set of parameters specific to it.

Some global behavior that we have for all systems though are the checkboxes for what you want to broadcast out to the system and the ability to set a Rate Limit Per Topic which will avoid certain very fast polling topics from spamming your network too fast. This doesn't slow down unique topic traffic, only repeated polling of a single one. It also doesn't impact your local processing, it just drops intermittent messages going out.

You can also define Manual Routes (custom listener addresses, topics, or paths that the controller reacts to). Each route is assigned an ID that can be referenced throughout the controller config and each includes a Path/Topic/Address.

When defining this you can use a '*' as a wildcard. You can use the Set Variable dropdown to tie the payload of this route directly into one of your defined variables so you can easily use it elsewhere.

The other way you will use it is by creating things that respond to it. Manual Routes can be thought of as listeners or things that you are configuring your system to respond to. They will trigger something further in your configuration. Your other external system will define the commands it gives and here's where you can build the response to receive them. An example could be a system that sends an HTTP command to override a puzzle's solution.

You would build the system to connect to it and use a route to configure your controller to react to its specific message. Or if you can configure the message it sends you can set it up to just send the message the controller is already listening for because along with creating each system the controller dynamically populates a full list of inputs and outputs for every system that listens and speaks to everything it has and does.

Definitions

This page starts to deal with the processing on the board but is still about building up the tools that we will access later. In general you can think of this as your dictionary of things that will later be used to actually make things happen. On this page you will possibly have missing sections if they don't have any modules that use them. An example of that is the Audio Files section that only shows when you have a Sound Module added to the configuration.

States

While nothing about the controller requires using States they are a very clean way to think about what is happening on it. In a simple prop we would probably default to having at least an Armed or Ready state and a Solved state. If the prop should have part of the game where it doesn't take inputs yet then we would add an Off or Disarmed state. These labels are all up to your discretion and only as meaningful as you make them. As you add states you will see the Auto To and Delay columns, these allow an automatic transition to another state after the configured time. In my simplest prop example after it becomes solved I might automatically transition back to Ready and now it's able to be solved again. I could also have a longer delay and have it automatically reset itself but from a game perspective we usually prefer triggering our reset so things can remain unlocked until the room reset. It is also possible to have many more stages that are linear, conditional or help you break out logic and become part of what allows something to happen or route.

Variables

Variables are a common programming term but if you don't have programming experience you can think of them as a thing with a value. That thing could be a place you write down a note to use later, or it could be tracking the value of something else to help you refer to it. Think back to Algebra when you used X to represent a number. The variable is X and it is allowed to change. The Type will let you configure it as a string (text), number or boolean (true/false). Most of the time you can just leave it as a string. The Startup Value is what it should be when you turn the system on before anything changes it. A real-world example of using a variable would be to keep track of an external prop that got solved, I could store that as a variable "propSolved" and then use later logic to decide whether we are allowed to do something based on that without having to tie my actual prop state to it. Variables have a lot of overlap with other ways of track conditions but are a very useful tool.

Debounce Defaults

This one is pretty straightforward, here you can configure how long different things have to "sit" before we register them. Imagine a button push that could just be electrical noise or accidentally pushed vs a long enough press that I know they meant it. Or another example is a reed switch where they have to place a piece with a magnet, I want to use a debounce to make sure they don't brush over it quickly and instead have actually placed the piece in the right place because they actually know it goes there.

Input Gestures

Input Gestures are primarily for buttons but they define behaviors that they can do like how long it has to be pressed for a short press event, double tap, long press and how long before they have "held" the button. These drive an event system in the controller which runs alongside the checks of "pressed/released" and the actual current state of the button. In general gestures can be a more sophisticated way to make sure you are reading things correctly.

Audio Files

This section is where you can upload your audio files to the SD card (not included). You'll need to make sure your SD card is formatted as FAT32 and insert it to the built-in slot on the bottom of the board between it and the breakout board. After that you'll be able to upload tracks over the network and manage them. There are test buttons for each track and you can apply a friendly name to use within the user interface.

The system is only able to play WAV files (not MP3) and the best settings for it are 16-bit PCM 22050Hz sampling rate. There are free converters available to make sure your files are the right format to play back well.

Behavior

This is where it all comes together. We've connected to the controller, added I/O modules, defined states and variables and now we want to actually tie together what HAPPENS with our prop. The Behavior tab is all about different ways to build forms of If This Then That logic. To say there are multiple paths here is an understatement, figure out what works for your process and workflow and don't be afraid to solve things in a new way than you've seen others do. There isn't a right and a wrong.

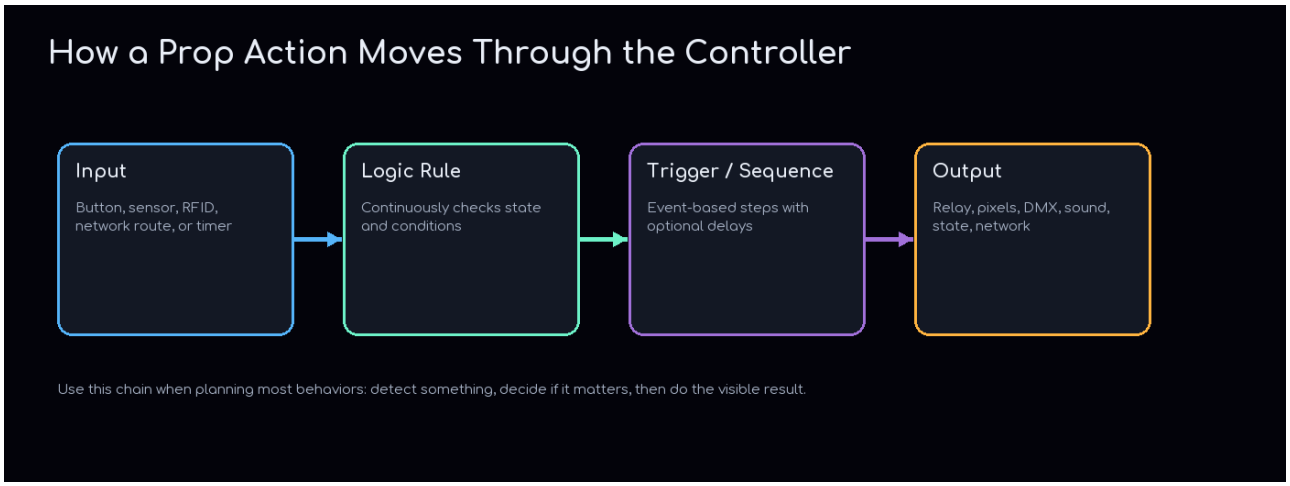


Figure 4: How an input becomes an output.

BREADCRUMB Operator Page > Open Configuration > **Behavior**

Logic Rules

Logic Rules are conditional setups. You define a rule and give it Conditions and when those are met it fires. A Logic Rule doesn't actually have any output past "firing", you will use something else to define the output side of the equation. Logic Rules are however the most complex and stackable of the behaviors on the inputs side.

First you give your Rule ID, this is how you will refer to it later when you use it as part of an Action Trigger or similar to make it actually do something. The Match dropdown has options for All (every condition must be simultaneously true), Any (only one condition will make the whole rule fire), None (every condition must be simultaneously NOT true), and First Match which can act as a router or a Switch if you're familiar with that programming function. There is also a field for Re-fire Every (ms) which allows you to have the trigger fire as long as the conditions are still meeting their setup. If left at 0 it will only fire once until the rule fails and then succeeds again.

You can have a single condition or multiple. Each condition has a Type which is what it is looking at, this list is long and dynamic based on what you have in your setup but you can look at most anything you can think of. Some important ones to call out is looking at an Input looks at its current state (true/false, on/off etc) vs you can't actually look at a Gesture since that is an event and Logic Rules are not event based, they are state based, they want to be able to continuously evaluate "is this all following my rule RIGHT NOW". You could use variables to turn an event into a state by changing the variables value to the last event though.

For most of the Types when you select it you will get a dropdown of IDs to pick which one you want to evaluate for this condition. The Op is the operation that it will check against and the Value is what it's looking for. The Op list will change based on what possible values there are. For example a State can only equal or not equal a value (it either is in that state or it isn't) but an analog sensor could be less than, equal to, greater than, not, etc a specific value. You can also define the debounce here, if left on default it will look to our global setting but you can also choose a specific time delay for this singular condition to wait and make sure the value has settled there for real. If you find yourself doing this often you may want to go edit the global/default debounce to set it there since you can change that and all things following it will go along with your change.

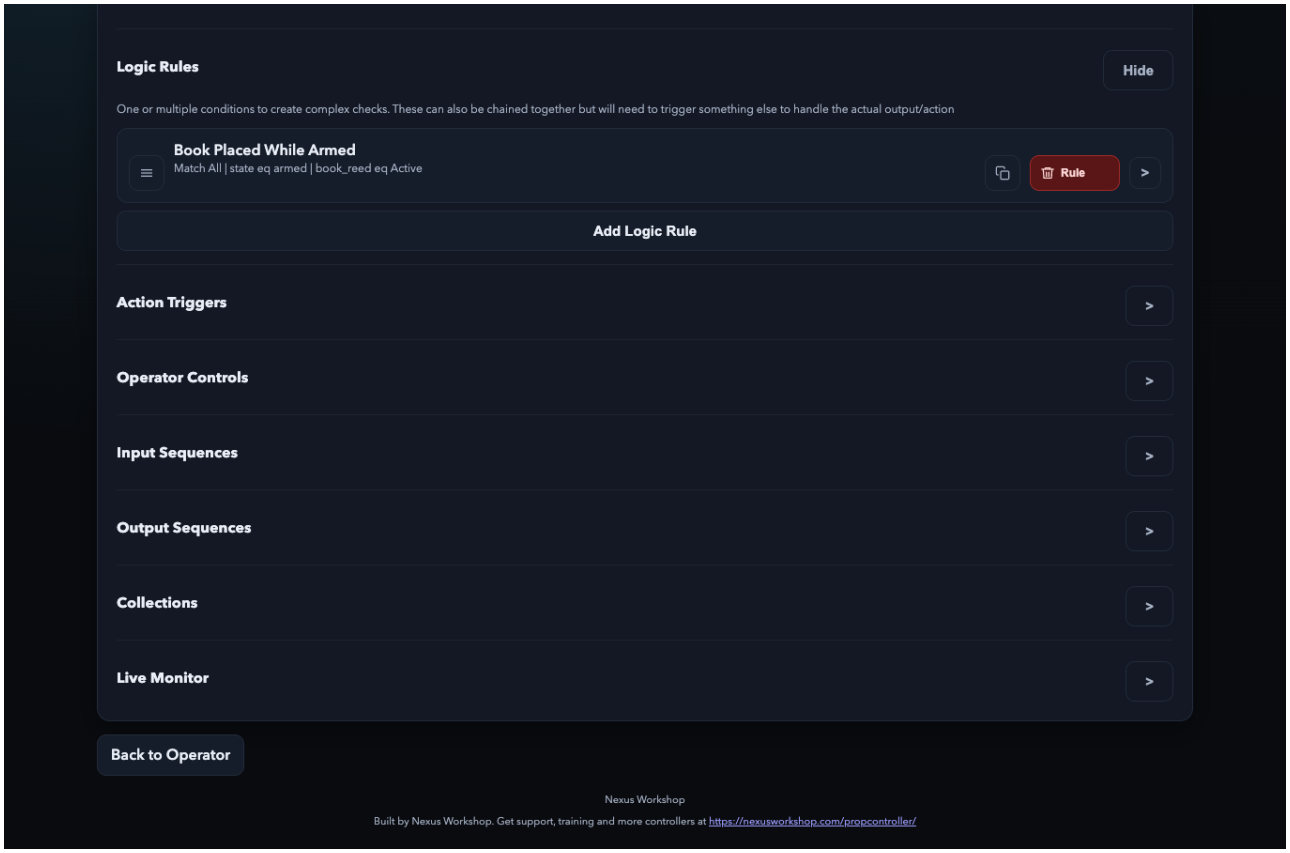


Figure 5: Logic Rules editor.

One of the most powerful features of Logic Rules is that they can feed each other. This lets you build extremely complex logic flows with a combination of ands, ors, nots and more to really get in and make sure you have control of what happens when. A lot of props won't need this but it's very satisfying when you do need it.

TIP Logic Rules are state-based. Use them when the question is "is this true right now?" For button taps, RFID reads, sequence successes, network messages, and other momentary events, route that event into an Action Trigger or use it to set a variable/state that a Logic Rule can evaluate.

Action Triggers

Action Triggers can be thought of as the actual "do-ers" of our behavior system. They have a single input that makes them activate but it can be just about anything you can think of including Logic Rules firing. Action Triggers are an event based system so every time that event happens they will do their output(s). They also have the ability to configure multiple Steps, these by default all happen simultaneously when the action trigger fires but they can each have a delay added to build simple sequences. The delays for each step don't stack, they all start counting at the same time when it goes off.

STEPS 1. Choose the input event, such as a Logic Rule firing, an input gesture, a sequence success, or an operator button. 2. Add one or more output steps. 3. Use per-step delays when outputs should happen in an order. 4. Test with simple visible outputs first, such as a relay test or pixel flash. 5. Use Pre-Compile after larger behavior edits.

Operator Controls

Operator Controls are buttons and UI elements that can be placed on the Operator Page to allow interaction directly with the system, this could look like a dropdown that sets the props state, a text field to show on a display or simply a button that does something. They have a feedback system that can look at the output itself or be based on something else external to it (like a variable or state) and they can provide a confirmation before firing if it's something that is important to not mess up. They have the entire multi-step output options system that Action Triggers have and they can get pretty

powerful used in conjunction with variables and logic rules.

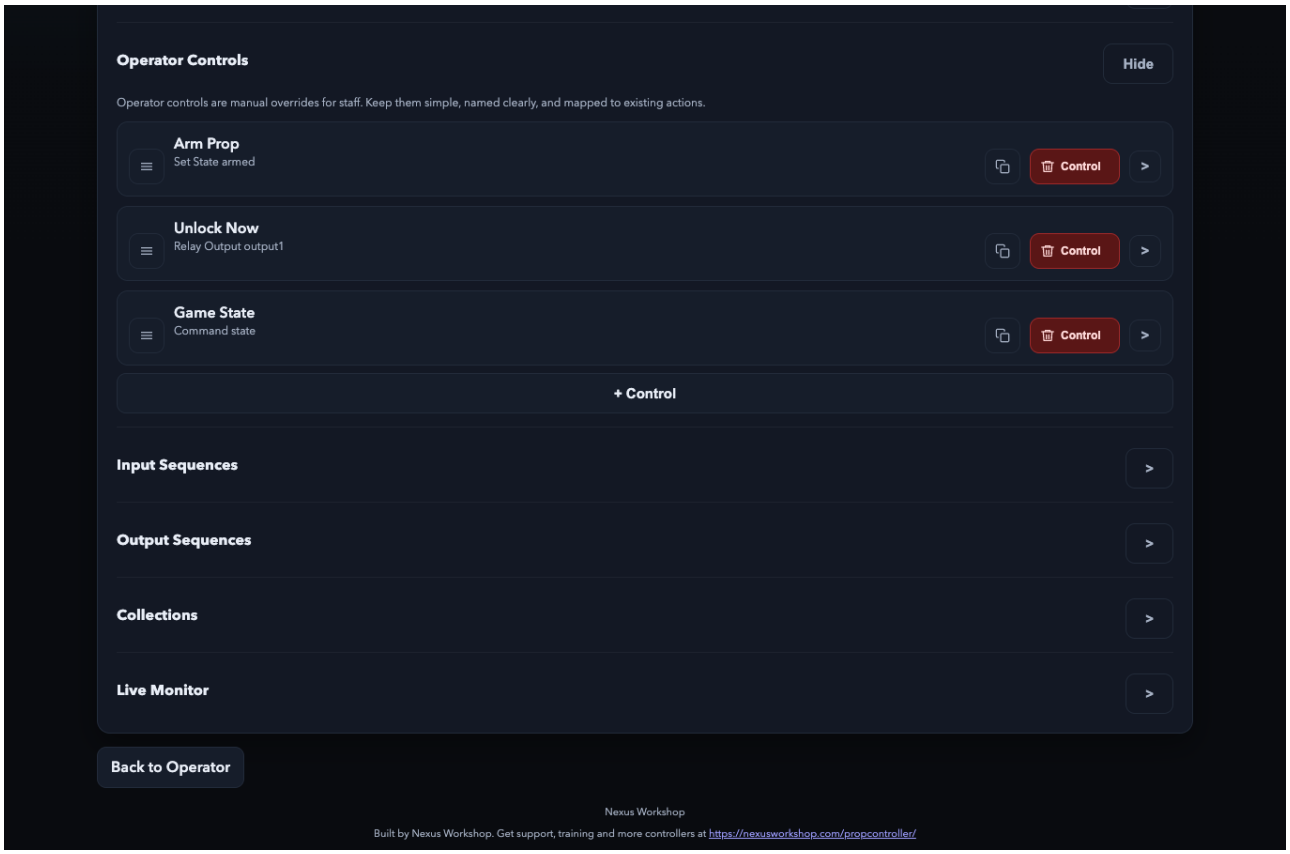


Figure 6: Operator Controls configuration.

TIP Add confirmation prompts to controls that unlock doors, reset games, start loud effects, or otherwise create a real-world consequence that a game master should not trigger accidentally.

Input Sequences

This is the section that most other prop controllers have. Enter a code, do a thing. There are 2 types of Input Sequences, Ordered, and Timed.

BREADCRUMB Configuration > Behavior > **Input Sequences**

Ordered Sequences Ordered Sequences are your traditional code, press the buttons in this order. But in typical Nexus fashion you can take this concept a lot further. The global settings for an Ordered Sequence are the Timeout, Mode, Only In State(s) and Correct Code. Mode has a few different options and they each change what happens with our Attempt and the Timeout.

In Continuous mode it looks for the Correct Code and builds up an Attempt as players enter Tokens (button pushes). The attempt keeps updating as tokens are entered and will wrap around until right tokens are pressed in order with no incorrect in between them. The Timeout will reset the attempt to empty it if we go that long without any new button pushes but ultimately previous wrong attempts won't impact us at all.

Instant Fail mode means that the first incorrect button will send a fail command and clear the attempt and timer. Timeout still waits that long since the latest button to clear the attempt and start over.

Allow Misfires looks for the correct buttons pressed in order but it doesn't care what happens in between them. You can think of it more like collecting than an actual order and it doesn't collect anything wrong for that stage of the attempt. In this one the Timeout will clear the attempt if we have that long between the first correct token and the last one.

Mode	Use Case	Wrong Input behavior	Timeout behavior
Continuous	Default behavior, inputs in order create success	No impact	Clears after inactivity
Instant Fail	Strict Code Entry	Sends a fail event	Clears after inactivity
Allow Misfires	Correct inputs must be in order, extras between are allowed	Ignored	The timer is basically a race to complete the sequence

Token Mapping

Now we've been saying "button pushes" but you can actually configure many different types of events/inputs to be a Token. You use the Add Mapping button to map an input to a token. You could think of this in a simple way as having 4 inputs; button 1-4 and they are mapped to tokens 1-4 when they have a Short Press gesture event fire. But we can expand this a lot further if desired into looking for different gestures on the button(s) (perhaps a single button where we need short press, long press, double press in that order), we could also tie a single analog input and map different values on it to tokens low, mid, high and then use the Allow Misfires mode to ignore all the values in between them but collect as we sweep from high to low to mid as our code.

Naming tokens

These Tokens can be called whatever you want. For example, instead of using 1, 2, and 3, you could name them **Cat**, **Dog**, and **Tree** to make it easier for the game master to follow what players are entering.

State Restriction

Looking back at the global settings for an Ordered Sequence you can use Only In State(s) to restrict when a sequence can be activated. This makes it possible to build multi-round puzzles, Simon-style progression, or different code behaviors in different prop states.

Additional Feedback

Note also that you can tie action triggers to these same sequences or to the inputs themselves. This allows you to add feedback such as lights or sounds that are outside of the sequence itself.

Also note that if you pull up some of the Prop Wizard games as a quick start you will see us configuring things like this because we love giving players feedback during puzzles so they know everything is working and they are doing the right thing.

Timed Sequences Timed Sequences are a good way to handle things like a knock puzzle or morse code. When you build one of them you have some familiar options like Timeout and Only In State(s) but you also have some new options like Tolerance and Absolute Timing.

You will start by using Add Step until you have all of your "taps" added. Then you can use the Learn button to capture timing on the real device/input you are using. You can also edit the timing manually but this is a very fast way to capture complicated timings. Absolute Timing when enabled means the code is actually looking for those ms delays between each input, if you disable this checkbox then it means it is only looking for relative timing. If you think of a song it is still the same song at a faster tempo or a slower one (within reason) and that's what the sequence now looks for, it wants the relative delays between steps to be similar even if they all happen faster or slower than how you captured them originally. The Tolerance is what actually makes a game like this work in the real world, this lets you give some wiggle room so players have to know the solution but don't have to be robots to pull it off.

Output Sequences

Output Sequences are similar to the steps and output side of an Action Trigger, they let you build complex chains of things that happen, but they don't have any input conditions. You will trigger Output Sequences from something else like an Action Trigger. The big advantage to Output Sequences is the ability to Add Step and Learn Timing similar to Timed Sequences. You can use the Learn Timing button and then click on Step Timing to capture your delays automatically. This can let you time things to music or build things like morse code output sequences quickly and easily. With Output

Sequences you can use the Test button to play them and you can use the Test Count to choose a number of times to loop while testing (or 0 to loop until stopped).

This is the other big advantage of Output Sequences, when you fire them from something you can choose if you want them to repeat a set number of times, just once, or loop forever. You also get the ability to stop them with another action or do a thing called Devamp which will let it continue its current iteration and then stop so it doesn't bail mid-run. This makes them really powerful for creating effects like a "lighting storm" across sound, motors, relays and lights.

Collections

Collections are a different way of thinking about puzzles. Imagine a statue game where you need to have the 4 statues on the right pedestals. This could be handled by a simple Logic Rule looking for All of those conditions to be true (each statue is placed). A Collection gives you a little additional power by sending events and keeping track of the count at the same time. So you can now tie a meter of addressable pixels and have it fill up as they gather the items. Now of course you don't always want to show that progress (since it could give away the puzzle) but it can also allow you to build feedback where placing any item plays a sound and turns on a glowing light but that doesn't mean it's the right or wrong piece until they are all in place.

Collections can also allow you to have multiple correct solves that have different outputs, or just the same output if there's multiple ways to do it. Lots of possibilities and collections can work with RFID or simple reed switches, by setting Source to be virtual you could even take other things (like an Ordered Input Sequence) and have them collect all of the launch codes. Nothing that isn't possible through the use of Logic Rules and Action Triggers but a more purpose built way to get there.

Live Monitor

Live Monitor allows you to choose which events you want to look at and see them as they happen in real time. This can help quickly debug issues with buttons and input sensors in addition to the live view on the Connections page in each module. You can Clear or Pause the Log and you also have the ability to Download Log which will give you an output of just what you can see onscreen. You also can Download support Log or Send Log from here, this is one of the tools we will ask for when troubleshooting config issues with you since it gives us a lot of visibility into what is happening

Lighting

BREADCRUMB Operator Page > Open Configuration > **Lighting**

Pixels

The biggest part of the Lighting page by far. This section actually has several sub-sections itself and is where you will configure all of the logical elements of the pixels attached together to your actual pin on the board. At the top of this section it shows all of your configured pins that you added on the Connections page. The count here is generated from your config below but this is where you will set the global Color Order and Gamma for that pin.

STEPS 1. Add a WS2812B pixel output in **Connections > Inputs, Outputs, and Modules**. 2. Click **Apply I/O Changes** and wait for the controller to reboot. 3. Open **Lighting > Pixels**. 4. Define the physical element order for that pixel pin. 5. Create groups for the logical targets you want to use in effects and behavior outputs. 6. Use the element test buttons before building effects so wiring mistakes are caught early.

Pixel Elements Order

This is where you start to define what your pixels are. By default it created a string of 20 pixels but you can change the Type, ID, and Size of that as well as add more. The ID is how you will refer and select this Element later. The Size is how many pixels are in it and the Type is whether it is a String, a Ring or a Matrix. If you select a Matrix it will ask you the size of rows and columns then multiply those together as a rectangle to get your actual pixel count for that element. There's actually one more type and that is Null. Null pixels will never light up. Their purpose is to help you build in gaps without having to cut wires and re-splice, they can also be useful to help handle longer runs between controller and pixels or two strings of pixels.

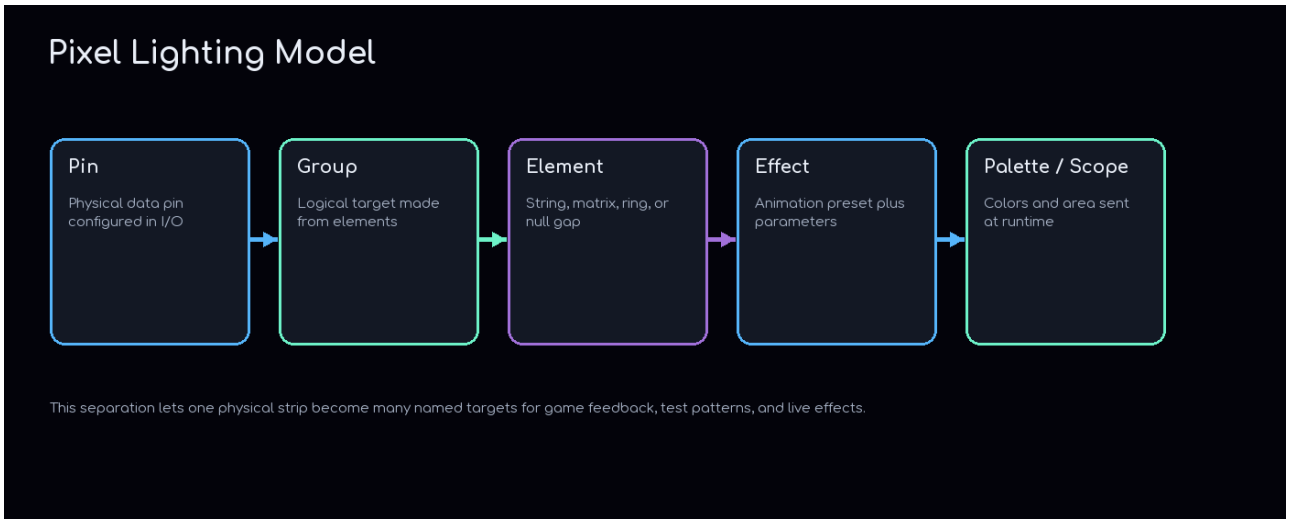


Figure 7: Pixel lighting model.

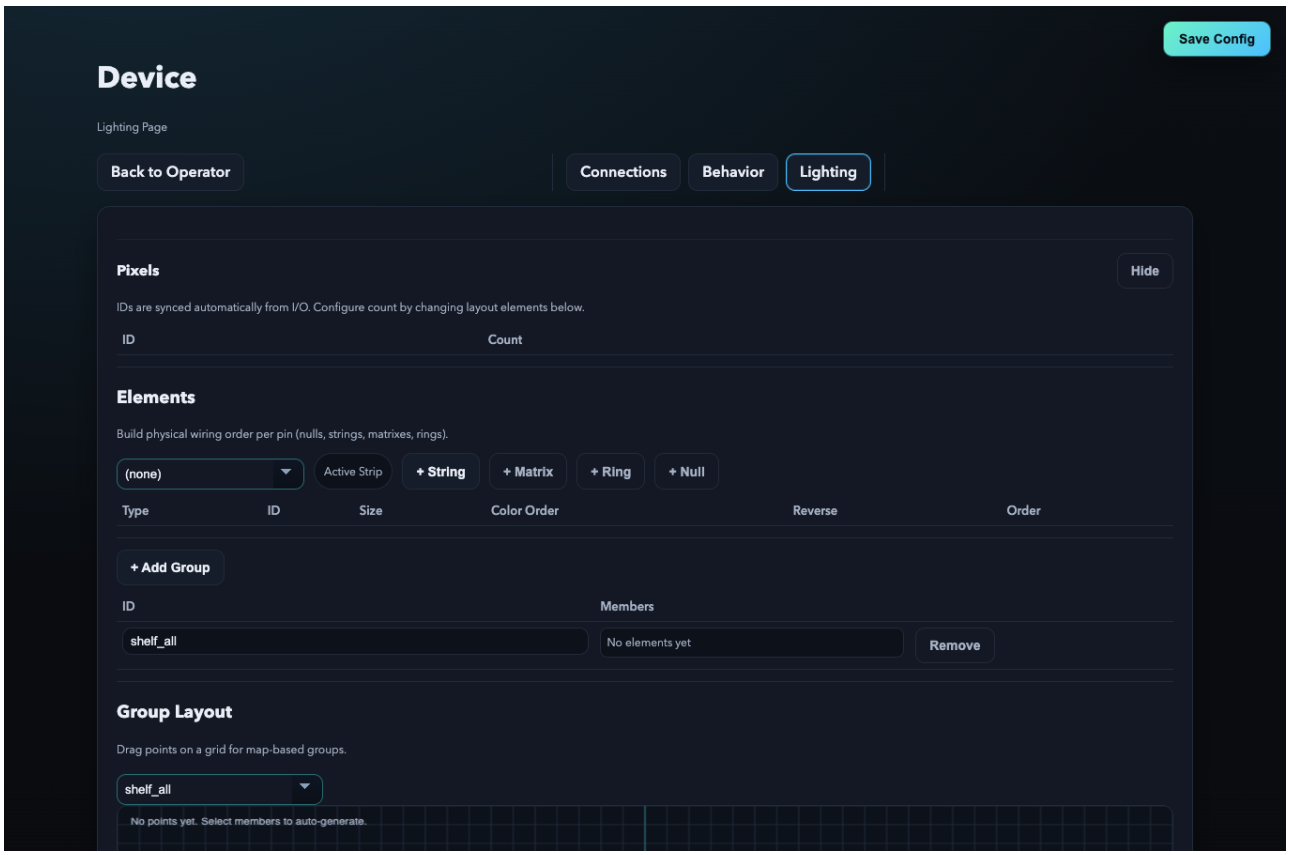


Figure 8: Pixel lighting page.

For each of these Elements you can choose whether they follow default or have their own Color order and Gamma. If you leave it as default it will show you the effective setting that it is inheriting. You can also choose to reverse an element and you can duplicate or remove them from here. There is a play icon for the test button. This will flash a pattern on that single element and is extremely useful for confirming your setup really is how you think it is. There is also a grab handle on the left side of each element that lets you re-order them within the physical chain of devices.

Now it's been awhile since we had something that didn't auto save but this section is one of those. There is a manual save button and it will glow green if you have unsaved changes. It will also go ahead and apply itself automatically when you navigate away and it doesn't cause a reboot when saving so it isn't as intrusive as the others but not auto-saving makes this section perform a bit better when you are renaming and moving around a whole lot of pixel elements quickly.

Groups This isn't technically a section but you can now make logical groups of pixel Elements. They each get an ID so you can refer to them later and you can select which Elements are members of them. Elements can be a member of multiple groups and groups can even be members of other ones so you can chain things together for applying effects to different hierarchies.

Group Layout

This is one of the biggest magic parts of the Nexus Ultimate Prop Controller. For each group you can physically map out how the pixels are spaced with each other. There are view navigation buttons at the top left and you can also use Shift+Drag with your mouse and the scroll wheel to move around the view quickly. You can click on individual nodes, click again or click on the Element's name to select all nodes in the Element and you can move, rotate and scale each of them. You can also Shift+Click to make multiple selections or drag a bounding box to grab and move things around. When you have multiple things selected you can rotate and scale them as a grouping.

When you apply 2 dimensional pixel effects to this group it will reference this Group Layout to map those effects and this can be incredibly powerful when used well.

> **CHECKPOINT** If a 2D effect looks rotated, mirrored, or stretched, fix the Group Layout instead of compensating in every individual effect. The layout is the shared map all map-based effects will reuse.

Color Palettes

There are built in colors that can be used throughout the system but sometimes you just need to define a specific one and here is where you will do that. With single colors this is pretty straight forward but its real power comes from the ability to add multiple colors into a palette. Once added you can re-order them and there are effects that will use multiple colors for different parts of them. So the Color Palettes have the ability to be entire color profiles or swatches.

Pixel Effects

There are several built in pixel effects that can be applied without doing anything else, but you can also add new ones here where you start with a base and then configure parameters that are specific to that effect. From this section you can choose a Test Target, Scope and Test Color then fire off the effects you have created as you play with settings.

Also as a note when you trigger Pixel Effects from an output item (like Action Triggers, Operator Controls or Output Sequences) you have the ability to send Color Palette and Scope so you could run an effect across a group of pixels and choose if it is fading within the entire group or within each element, etc. You can also apply transitions to running effects and do some layering that make for a very powerful pixel engine

DMX Fixtures, Profiles & Patch

This section is where you will define what you have connected via DMX. The first thing in this section is a list of fixture profiles within your project. Each one shows an ID, a description/name, how many channels it takes up and some tags of specific control channels it has. You can manually edit these profiles by adding channels of different types. Some of these channels can have "Wheel Slots" which are essentially mapped regions that have a function. That could be useful for something like a color or gobo wheel as well as channels with macro functions on them. At the moment the system isn't

able to handle variable ranges within a wheel so you would need to treat that channel as a generic channel.

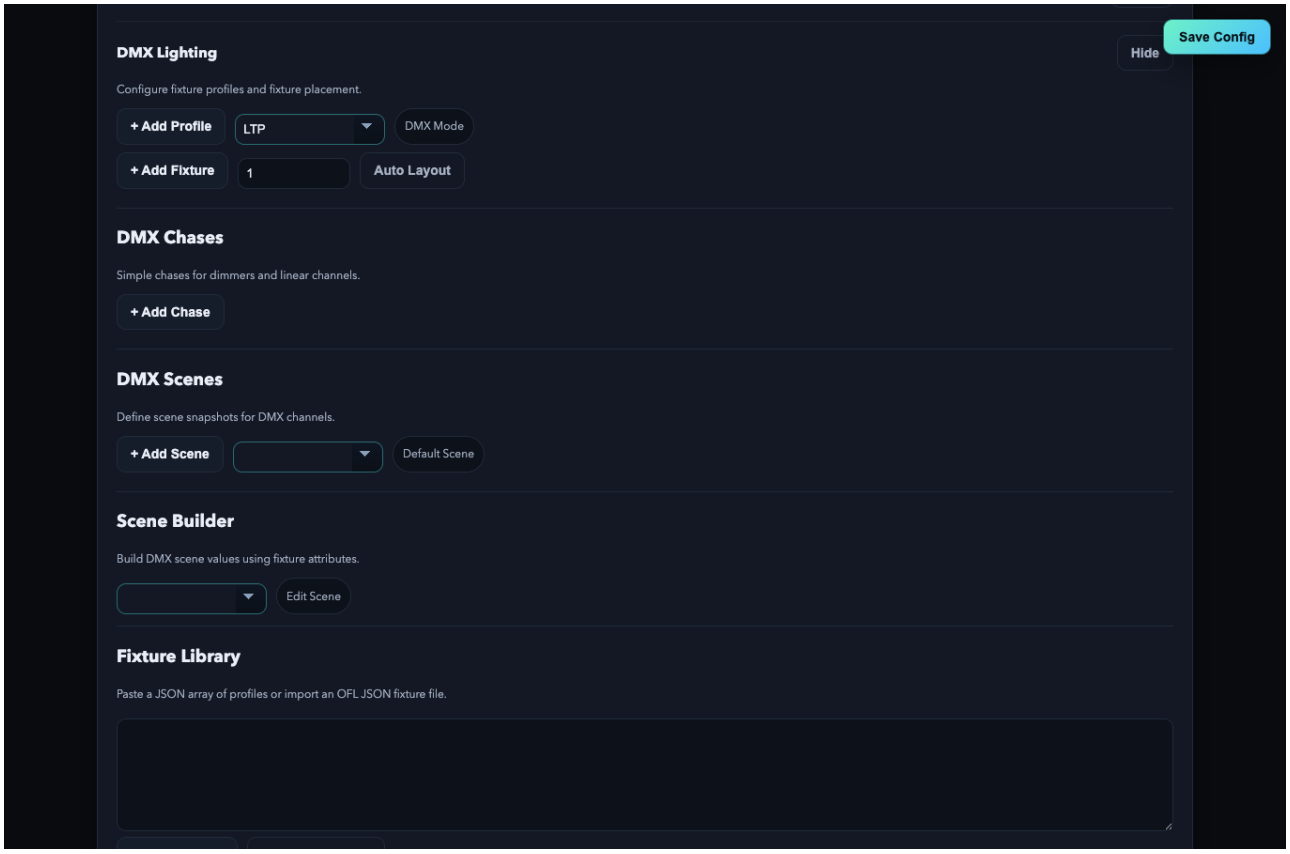


Figure 9: DMX scene builder.

You can also use the Fixture Library to import JSON or fixture profiles that you have either created or found from a system like the Open Fixture Library project.

Once you have your profiles configured you can add the actual light fixtures themselves, each one will get a fixture #, an ID and select one of your profiles. As you add them they will automatically add to the end of the available channels but you can edit their starting DMX channel as well as their fixture number to reference easier later. The centered checkbox pertains to moving lights to tell them where they should home themselves, at 0 or at the midpoint.

After adding fixtures you can combine them into groups. These groups can share the same fixtures in different layouts so you can control things with the right hierarchy. The groups also get a # that can be used later in the console for quicker programming but not that it always gets a G added to the beginning of it to make sure there are no conflicts with fixture numbers.

DMX Chases, Scenes & Scene Builder

DMX Chases

This section is under construction and will be coming soon in a firmware update.

> **NOTE** The firmware UI already exposes DMX Chases in the lighting page, but this manual should treat them as an evolving feature until the final chase workflow is locked for the release you are documenting.

DMX Scenes

Here you can add scenes as well as define their scope and values. The scope can determine whether a scene changes all lights or only the ones that it “touches“ , meaning ones that are explicitly set in its memory. At the top right of this section is a drop down that can select which scene is live though

in runtime you will typically be doing this with something like an action trigger.

Scene Builder

This is the main section for DMX, here you can set the levels and parameters of each fixture. This includes their brightness/intensity/dimmer but also things like color and gobo wheels or movement on fixtures that have those controls. If a fixture has been touched by a scene it's Release Fixture button will glow and can be used to remove it from the scene scope.

At the top of this section is a selection dropdown to choose which scene you are working on, there are also buttons to preview this scene live, clear preview and release all. You can choose whether you want to view parameters as percentage or as a range of 0-255. The last thing in this top area is the console, this is an area where you can type commands to program scenes quickly. These commands can include fixture or group names or numbers as well as ranges and IDs. The console will be further expanded later but right now is a simple way to bring lights up to levels.

> **STEPS** 1. Create or import a fixture profile. 2. Patch each physical fixture with a fixture number, ID, profile, and DMX start channel. 3. Create fixture groups when multiple lights should move together. 4. Add a scene, select it in Scene Builder, then set the fixture values. 5. Preview the scene live, then clear preview before returning to normal runtime behavior.

DMX Live Output

This section just gives you a quick view of what is happening on the DMX output in realtime, this can be very helpful for troubleshooting DMX fixtures and giving a quick sanity check.

The Operator Page

Now that the controller has been configured, you can view it through the **Operator Page**. Like the configuration pages, this page is generated dynamically based on your setup. It shows live information from the controller, including things like firmware version, current state, IP address, Inputs, Variables, connected modules, and any Operator Controls you have created.

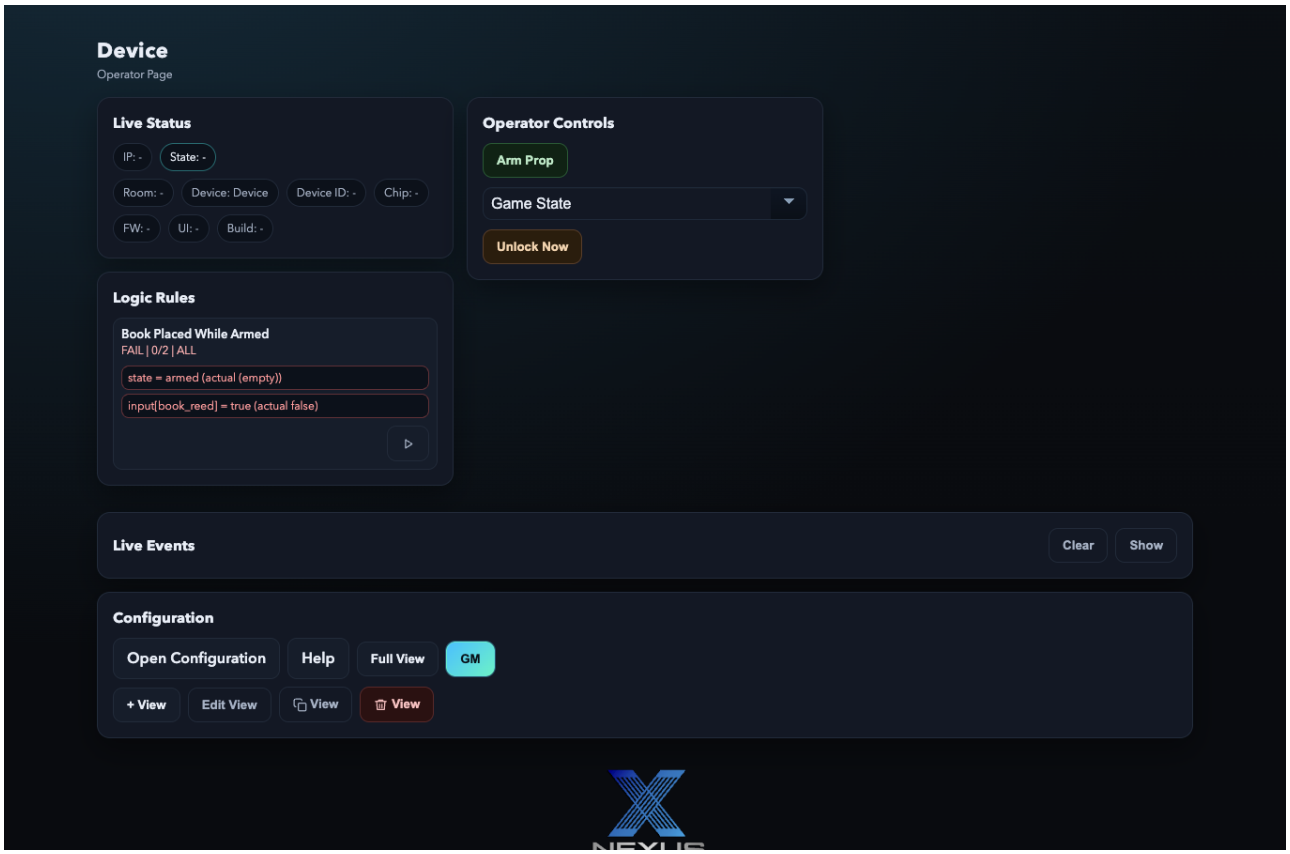


Figure 10: Operator Page with live status and controls.

Designed to give you a live view of what the controller is doing and provide direct access to the controls that you want available during operation. Depending on the configuration of the controller, this view can become crowded quickly. To help with that you have the ability to define Views.

Full View always exists and always has everything on it in the default order. Below that you can add a view, and edit, duplicate, or delete the currently active view (you can't edit or delete the Full View). When you add a view, you can give it a name and choose which sections and items within those sections you want it to include. You also have the ability to drag the handles on the left side of each item to move the order around.

When editing a view, the left side shows the sections or groups. The right side has a drop down to select which section you are editing then select which items you want within it. If a section is disabled you won't be able to have any of the items within it.

Up at the top of the Edit View window is a checkbox to make that view the default view, this means anytime someone connects to the device they will go to that view first. They can always switch to a different view at any time.

In practice, you may have one default view for all Game Masters but then have additional views for the individuals who prefer more, or less, information. You also could have a view with more diagnostic information only when needed and leave it streamlined the majority of the time.

STEPS 1. Build Operator Controls in **Configuration > Behavior > Operator Controls**. 2. Open the Operator Page and confirm each control appears with the expected label and style. 3. Use **Edit View** to create a game-master view with only the sections needed during normal operation. 4. Make the streamlined view the default, then keep Full View available for troubleshooting.

TIP Operator Views do not delete the underlying controls or data. They only decide what is visible in that view, which makes them useful for show mode versus troubleshooting mode.

We want to give enough power for the Operator page to be a useful tool for most users but it's worth noting that we don't think this replaces an actual room control system/dashboard for more complex rooms. There are ways to make it provide information from other props and controllers but it gets too big pretty quickly. If you have a more integrated or complex experience that would benefit from a single, complete, full-featured control dashboard that is something we can absolutely help with, just let us know. Those need to be more tuned to the specifics of the experience so they are handled at a custom level, the tools within our prop controller system make them very easy to develop and integrate though.

Factory Reset & Recovery

You can factory reset from the System Setup menu in the Configuration pages. If you can't get to the configuration page but the device is on your network then the Nexus Controller Manager app can let you "Neuralyze" the board which is rarely plan A but can help recover. If all else fails and you desperately need the config on the board and don't have backups, reach out to support. If all else has failed, you can't connect to the board, you can't get in then you can connect a wire from GPIO17 to GND and power cycle the board. This will do a complete factory reset and unlock everything but your data will be lost in the process.

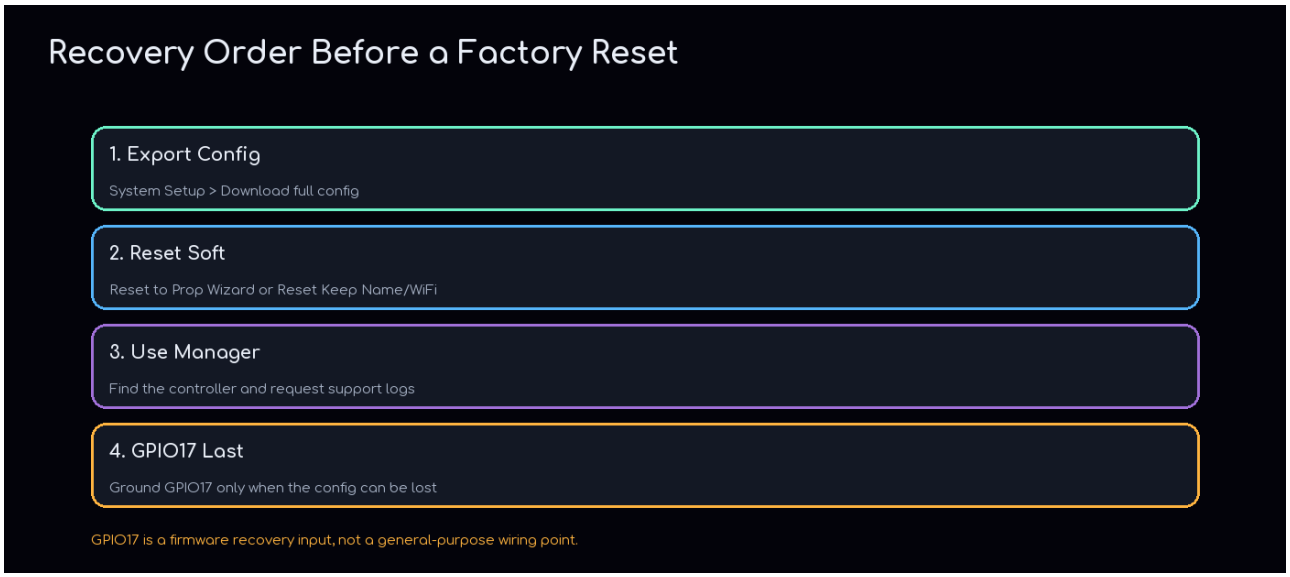


Figure 11: Recovery order before factory reset.

WARNING A GPIO17 reset is the last resort. It is meant for a locked-out controller where other recovery paths have failed, and it will wipe the configuration.

BREADCRUMB Configuration > System Setup > **Download**, **Prop Wizard**, **Reset (Keep Name/WiFi)**, or **Factory Reset**

Where do we go from here?

Like we said earlier this system is constantly evolving and growing. If there are modules or features you think would be helpful we'd love to hear from you. We also are always interested in seeing the problem solving behind configuring these controllers so would love to see your interesting solutions in our Facebook community. We'd even love to help create and promote content on those use cases as we expand out our training and video demos of the system.

We hope the system will prove as useful for you as it has already been for us as we build our own experiences and we look forward to seeing the entire industry continue to grow!

Glossary & Module Reference

Glossary

Access Point (AP)
A mode where the controller creates its own WiFi network so a device can connect directly to it for setup or recovery.

Action Trigger
An event-based behavior that performs one or more outputs when its assigned input or event occurs.

Admin Password
The password that protects access to the full configuration pages of the controller.

Analog input
An input that reads a changing value rather than a simple on or off state, such as from a potentiometer or analog sensor.

Analog mux expander
A hardware device that increases the number of analog signals the controller can read by switching multiple channels through fewer controller pins.

AP SSID
The network name broadcast by the controller when it is operating in Access Point mode, allowing other devices to find and connect to it directly.

Audio Files
The detected sound files available on a connected sound module, which can be given friendly names for use elsewhere in the system.

Bus
A shared communication path used by multiple devices to communicate with the controller.

Collection
A behavior tool used to track gathered items, matched inputs, or grouped progress toward one or more puzzle outcomes.

Color Order
The arrangement of color channels used by an addressable LED device, such as RGB or GRB, determines how color data is interpreted.

Color Palette
A saved color or group of colors that can be reused by lighting effects throughout the system.

Debounce
A timing filter that requires an input to remain stable for a short period before the controller treats it as a real action.

DHCP
Dynamic Host Configuration Protocol. A network service that automatically assigns a device its IP address and other network settings when it connects to a network.

Digital input
An input that reads only two states, such as on and off or pressed and released.

Digital output
An output that switches between two states, such as turning a relay or simple indicator on and off.

DNS name
A human-readable network name used to reach a device instead of entering its numeric IP address.

ESP-NOW
A wireless communication method used by ESP-based devices that allows controllers to share data such as network credentials directly.

Ethernet
A wired network connection used to connect the controller to a local network.

Expander
An external module that adds more usable inputs, outputs, or communication capacity than the controller has on its own.

Firmware
The software that runs on the controller hardware and defines how the device operates.

Gamma
A brightness correction method used to make changes in LED intensity appear more natural to the human eye.

GPIO
General Purpose Input/Output. A hardware pin on the controller that can be assigned to different supported functions.

Group
A logical set of pixel elements that can be controlled together for lighting effects.

Group Layout
The spatial arrangement of pixels within a group, used so two-dimensional effects can map correctly across them.

Hostname
A network name assigned to the controller that can be used to identify and access it on the local network.

HTTP
Hypertext Transfer Protocol. A network communication method that can be used by the controller to exchange commands or data with other systems.

I2C address
The numeric address used to identify a specific device on an I2C bus.

I2C bus pins
The pins used for I2C communication, typically SDA for data and SCL for clock, allowing multiple devices to share the same bus.

I2C multiplexer
A device that separates I2C devices onto isolated channels so multiple devices with the same address can be used together.

Input Gesture
A button-related event such as short press, long press, double tap, or hold that can be used as a trigger in the system.

Input Sequence
A behavior that looks for a defined series of inputs, such as a code, pattern, or timed rhythm.

IP address
The numeric network address used to identify the controller or another device on a network.

JSON configuration
The controller's saved configuration data in JSON format, used for backup, sharing, import, and export.

Logic Rule
A condition-based behavior that evaluates one or more conditions and fires when those conditions are met.

Live Monitor
A diagnostic view that shows selected controller events in real time for testing and troubleshooting.

Manual Route
A user-defined path, topic, or address used to receive or organize communication with an external system.

MQTT
Message Queuing Telemetry Transport. A lightweight network messaging protocol commonly used

for communication between controllers, devices, and servers in automation systems.

MQTT broker
The server or service that receives and distributes MQTT messages between connected devices.

Null pixel
A placeholder pixel that is counted in the wiring chain but never lights, often used to represent spacing or gaps.

Optocoupler
An electrical isolation component that allows one circuit to control or sense another using light, helping protect low-voltage electronics from higher-voltage or noisy signals.

Operator Control
A button or interface element placed on the Operator Page so a user can directly interact with the controller.

Operator Page
The main user-facing page that shows live system information and provides access to configured operator controls.

OSC
Open Sound Control. A network messaging protocol used to send control data between devices and software, commonly in audio, lighting, and interactive systems.

Output Sequence
A reusable chain of output steps that can be triggered by another behavior and optionally timed, repeated, or stopped.

PoE / Power over Ethernet
A method of delivering both data and electrical power over a single Ethernet cable.

Pull-up
A configuration that holds an input at a default high state until it is actively changed by a switch or device.

PWM
Pulse Width Modulation. A method of controlling output power by rapidly switching a signal on and off, commonly used for LED dimming, motor control, and servo control.

Relay
An electrically controlled switch used to turn higher-power circuits or devices on and off from the controller.

Room Prefix
A configurable text prefix used as part of default integration addresses, especially for MQTT and OSC communication.

Serial
A communication method that sends data one bit after another over dedicated transmit and receive lines.

Simple UI
A restricted configuration view that exposes only selected settings and controls instead of the full admin interface.

SPI bus pins
The pins used for SPI communication, typically including SCK, MOSI, MISO, and CS, which provide fast communication between the controller and connected devices.

State
A named operating mode of the controller, such as Ready, Armed, Solved, or Resetting, that can be used to control behavior.

Static IP
A fixed IP address manually assigned to a device instead of being automatically assigned by DHCP.

Token

A named input value used in sequences, allowing events or inputs to be treated as code elements such as numbers, words, or symbols.

Variable
A stored value the controller can read, update, and use later in logic or behavior.

WiFi SSID
The name of a WiFi network as it appears when users search for available wireless networks.

WS2812B / NeoPixel
A type of addressable RGB LED pixel commonly used for lighting effects, where each pixel can be individually controlled.

Module Reference

General Module Types

Simple Inputs
Basic on or off input devices such as buttons, toggle switches, reed switches, and similar digital sensors.

Simple Outputs
Basic on or off output devices, most commonly relays, but also useful for simple indicator LEDs and similar loads.

Expander Modules

PCF8575
An I2C input and output expander that adds 16 digital pins. These added pins can be used for supported input and output functions and include internal pull-ups.

TCA9548A I2C Multiplexer
An I2C bus multiplexer used to separate devices onto isolated I2C channels. This is especially useful when connecting multiple I2C devices that do not support selectable addresses and would otherwise conflict on the same bus.

CD74HC4067
An analog multiplexer used to create additional analog inputs for the controller.

Lighting Modules

WS2812B / NeoPixel
An addressable RGB lighting module where each pixel can be controlled individually. The controller can assign many pixels to one pin, and multiple pixel outputs can be used when needed.

Output Modules

Linear Actuator Module
A module used to control a linear actuator through two control pins. It supports timed motion and can also use limit switches to detect fully extended and fully retracted positions.

Servo Controller Module
A PWM-based module used to control standard positional servos. It includes settings such as minimum pulse, maximum pulse, and startup angle.

Motor Controller Module
A module used to control motor direction and speed through three configurable pins. It supports multiple motor driver models.

Stepper Controller Module
A module used to control supported stepper drivers. It includes settings for speed, pulse timing, direction inversion, and optional enable-pin control.

Analog PWM Output Module
A general-purpose PWM output module used for dimmable LEDs, analog-style meters, or transistor-controlled higher-power loads.

Input Modules

RFID Reader RC522
A common SPI-based RFID reader used widely in props and escape room builds. In this system it is configured primarily through the CS and RST pins, while the SPI bus pins are handled through the bus setup.

RFID Reader PN5180
A longer-range RFID or proximity-style reader that uses more pins than the RC522 and supports different operating modes depending on the tag type being used.

Color Sensor TCS3472
An I2C color sensor that reports RGB values and a generated color number. Because these sensors typically do not have a changeable I2C address, multiple units usually require an I2C multiplexer.

Matrix Keypad
A keypad-style input module that uses multiple row and column pins. It is configured by defining the number of rows and columns and assigning pins to each.

Patch Cables
A puzzle-oriented module that detects connections between ports and can also identify which cable was used based on its internal wiring. It supports simple patch-panel behavior as well as more advanced cable-identification logic.

Display and Sound Modules

I2S + SD Sound
A I2S audio output to be connected with a I2S DAC to convert to analog audio and feed speakers/amps. This module is combined with the built-in SD card module to playback audio files.

7-Segment I2C Display
A compact numeric display module, typically with four characters and clock-style punctuation such as decimals and a colon. It is configured using its I2C address and route.

I2C LCD
A character LCD display module, commonly in 16x2 or 20x4 sizes, controlled through an I2C backpack. It is configured using its row count, column count, I2C address, and route.

Tools We Like:

Electric Screwdriver.

> If you are wiring to these boards these are just really nice to have.
> Fanttik E1 Electric Screwdriver (Paid Link)
> UnMela Electric Screwdriver (Paid Link)

Solder.
This is not inherently required for the controller itself. But may be used on modules in conjunction with the controller.

Station.
Hakko Digital Soldering Station (Paid Link)

Battery powered options.

> Having a battery powered soldering iron is really useful for making repairs/changes in field.

Fanttik Soldering Iron (Paid Link)

If you have a battery power tools a lot of times there will be options that work off of those batteries.