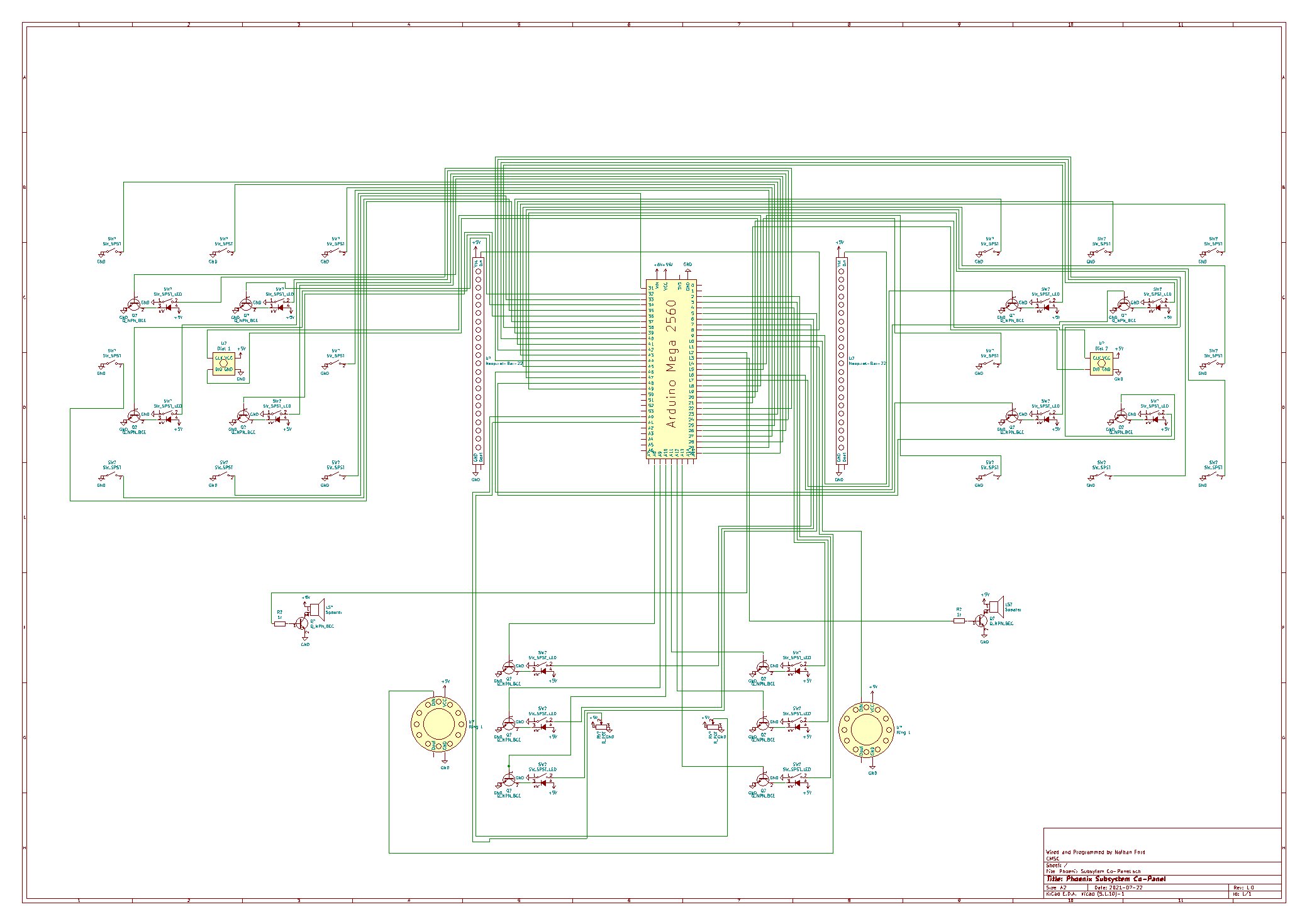
Phoenix Subsystem

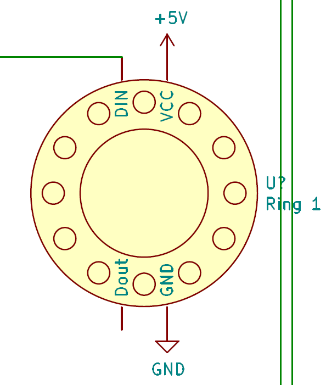
Co-Panel

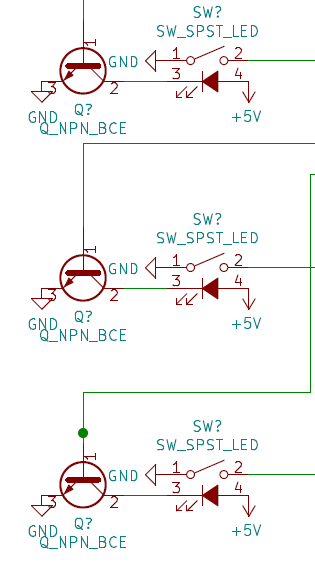


# Table of Contents

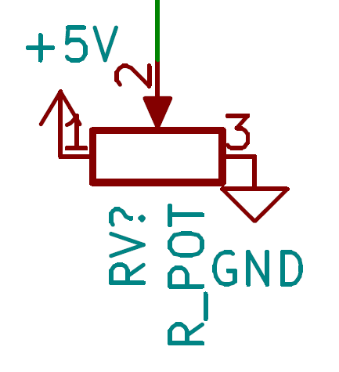
* Color Wheels – Pg 1
* Bars – Pg 2
* Speakers – Pg 4

Color Wheels

 This portion of the panel allows the user to lock in colors into the color wheel, then make them rotate. Once locked in and rotating they cannot be changed, it must be stopped to change the colors. This portion of the panel is also all mirrored for both sides, so keep this in mind, I will give you the pins for both sides.

1. Neo Pixel Rings – these are what the “Wheels” are, these are the feedback for this portion of this panel. They both need 5v and GND, but then side 1 (it doesn’t really matter which side is 1 and 2 if you are consistent) Din goes to pin 10 on the Arduino, and the Din for side 2 goes to pin 11 on the Arduino
2. Lock Switches – These are how the user “Locks” in the colors that they want to use, these are latching SPDT switches (SPST also works fine) with backlights. On the backlight we use an 2N2222 NPN transistor on the negative side of the light controlled by a pin on the Arduino, and 5v to the positive side. The Com port on the switch is hooked to ground, the normally off side is then hooked to the Arduino.

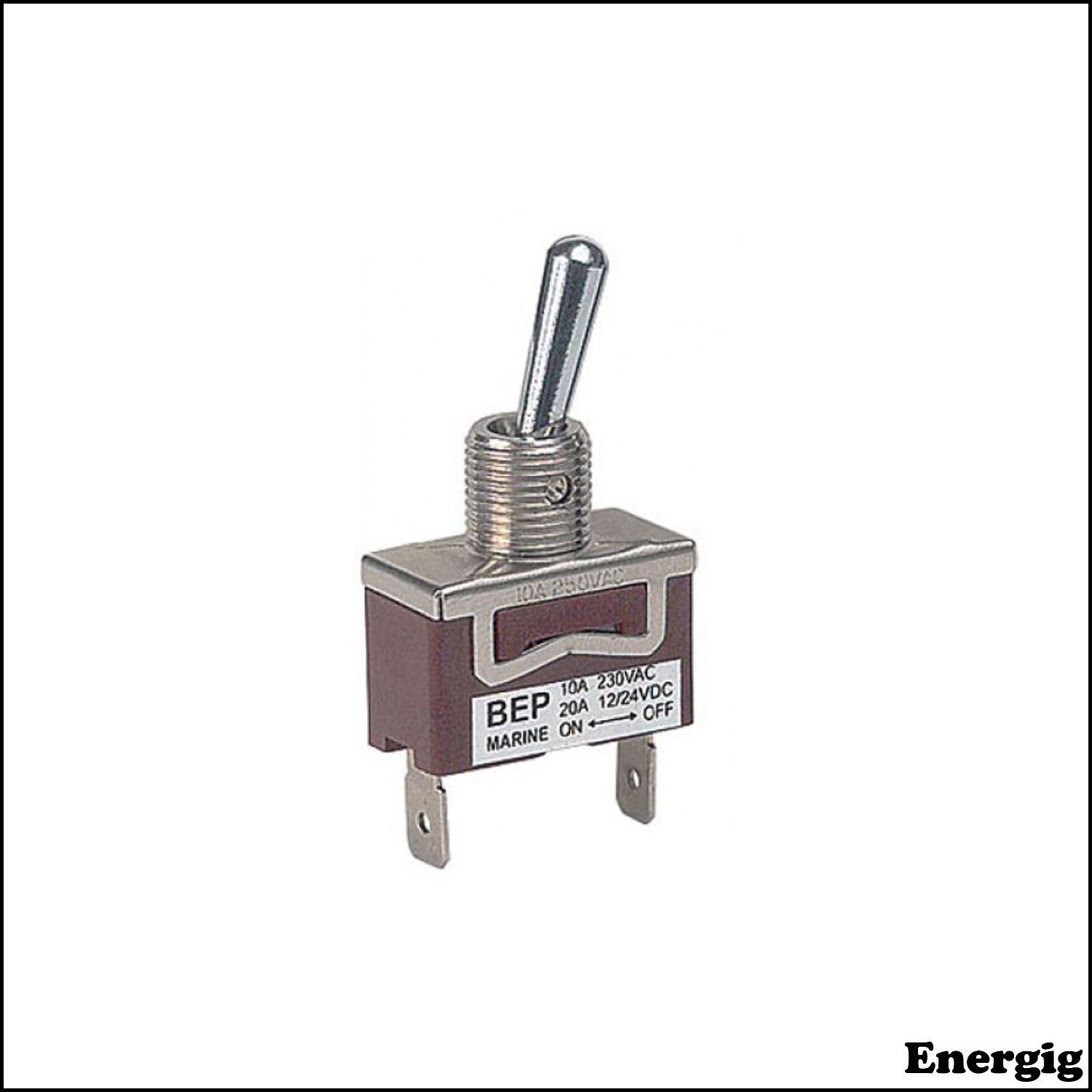
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Lock Switch** | **Switch 1 Arduino Pin** | **Switch 1 Light Pin** | **Switch 2 Arduino Pin** | **Switch 2 Light Pin** |
| Top | 7 | A8 | 4 | A11 |
| Middle | 6 | A9 | 3 | A12 |
| bottom | 5 | A10 | 2 | A13 |



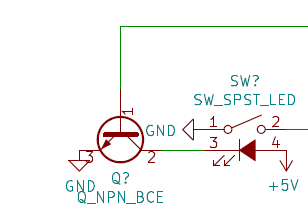
1. Slider Potentiometers – This is the device that the user uses to select their desired color, they slide it up and down, and it uses a color line algorithm to calculate the different colors. The com ports go to the Arduino, while the other sides go to either 5v or GND. Slider 1 is hooked to A0 on the Arduino, while Slider 2 is hooked to A1 on the Arduino.

Bars

I apologize for the name of this section, but I have no idea what they are named on the panel, because I do most my programming and documentation without the panel physically next to me because they are flying 99% of the time and I must come up with some name for them so the code is readable, and I couldn’t think of a better description XD. Also, for the switches, don’t tell anyone, but the order doesn’t really matter if you keep the toggles separate from the latching buttons, isolating each side, and wire the lights to the correct latching button for each input. The reason I have them wired like this is because random wiring just is lame and confusing, so I wanted there to be some type of order.

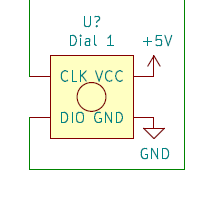
1. Toggle Switches – These are the primary HID part of this portion of the panel, in that they take up the most space and there are more of them than any other part, they are super simple to wire though, so that is a plus! One side of the SPST toggle switch goes to ground, and the other side goes to a pin on the Arduino.

|  |  |  |
| --- | --- | --- |
| **Toggle Location** | **Side 1 Arduino Pin** | **Side 2 Arduino Pin** |
| Top Left | 23 | 39 |
| Top Middle | 25 | 41 |
| Top Right | 27 | 43 |
| Middle Right | 29 | 45 |
| Bottom Right | 31 | 47 |
| Bottom Middle | 33 | 49 |
| Bottom Left | 35 | 14 |
| Middle Left | 37 | 15 |

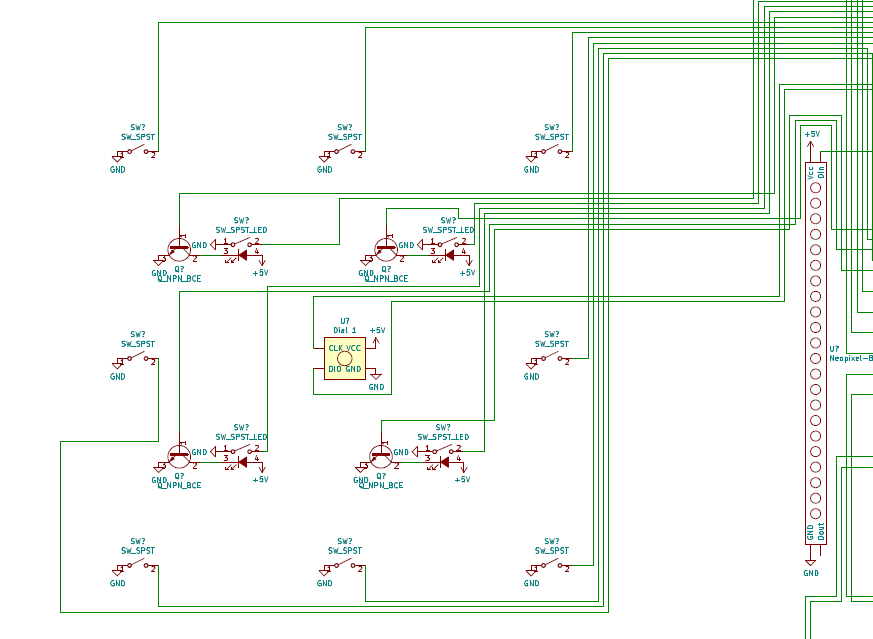
1. Glow Switches – These are the switches that form the inner square inside the square of toggle switches. These are SPST latching Light up switches, we chose blue and orange, they sadly were rated too high for the orange ones to light up with 5v, and the blue ones have been burning out… suffice it to say I would recommend getting different ones if you are building a new panel. They need GND on the com port of the switch, and need 5v in for the light, on the light you will also use a 2N222 NPN transistor on the other side.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Switch Location** | **Side 1 Switch Pin** | **Side 1 Transistor Pin** | **Side 2 Switch Pin** | **Side 2 Transistor Pin** |
| Top Left | 22 | 30 | 38 | 17 |
| Top Right | 24 | 32 | 40 | 16 |
| Bottom Left | 26 | 34 | 42 | 46 |
| Bottom Right | 28 | 36 | 44 | 48 |

Bars Continued…

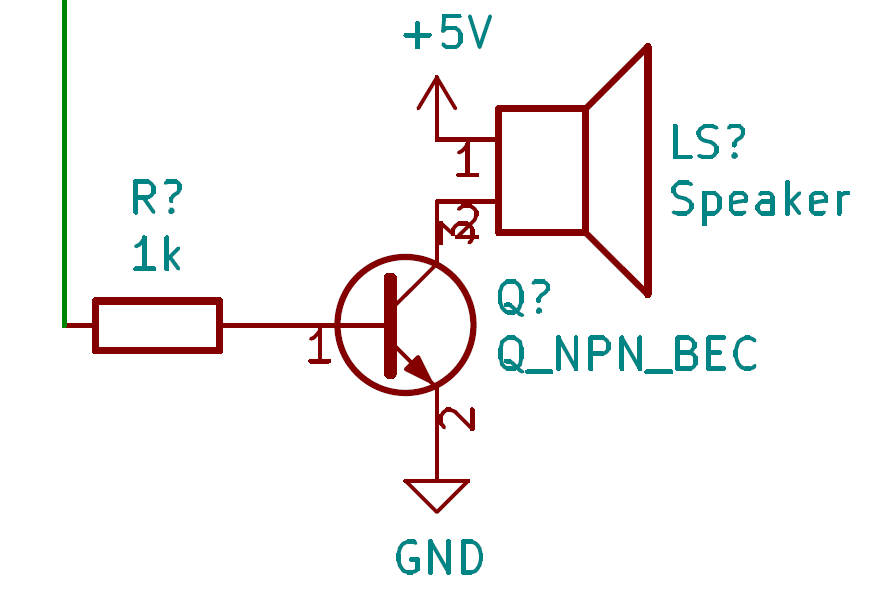
1. Rotary Encoder Dials – These control the speed of the animaiton on the bars, they need 5v GND and two pins on the arduino, their order doesn’t specifically matter, if the direction is backwards, then that means you put them in backwards.

|  |  |  |
| --- | --- | --- |
| **Dial** | **CLK** | **DIO** |
| Side 1 Dial | 18 | 19 |
| Side 2 Dial | 20 | 21 |

1. Neo Pixels – Finally, the part you have all been waiting for, the reason this portion of the panel is called the Bars! It is because we have these two Neo Pixel bars in the middle…. A bit of a let-down I know. These are what gives the user feedback based on the switches flipped, it also is the pretty part of the panel, looking visually interesting and drawing the user to the panel. The strands need 5v, GND and Din from the Arduino. Side 1 uses pin 8, and Side 2 uses pin 9 as Din.

Speakers

This is the portion of the panel that will generate cool sounds based on what switches are switched, unfortunately we have a problem though… Whenever the Arduino needs to write to the neo pixels, it garbles the sound generated, I can only assume it is because the neo pixel’s uses the clock lines, used for timing the neo pixel signals correctly, which are needed for making the tones, making them incorrectly, or it could possibly be that interrupts temporarily don’t work while it is writing to the pixels, causing delays you can hear. Either way I have not found a way around it yet… I have toyed with the idea of having an Arduino Pro Mini control the speakers, using serial to communicate with the mega, but in my experience, serial is very unreliable, perhaps it would be better if I could use I2C or SPI, but I haven’t used those to facilitate communication between 2 Arduino’s before. I also thought of using the Serial programming chip to run the speakers, but I don’t know if it has enough integrated clock lines. There is a possibility if I switched the controller to an Arduino Due, it could have enough brute strength to fix this, or to a Pi Pico with two cores could solve this, but that would require major reworks of everything first. I have been unable to implement a working solution as of 7-23-21, hopefully I will get one fixed soon, but here are how the speakers are wired as of right now:

 We give the speaker 5v power on one side (place a potentiometer here if you want to have volume control) and then on the other side connect it to a 2N222 Transistor to ground. The signal for the transistor comes through a resistor from pins 12 (side 1) and pin 13 (side 2)