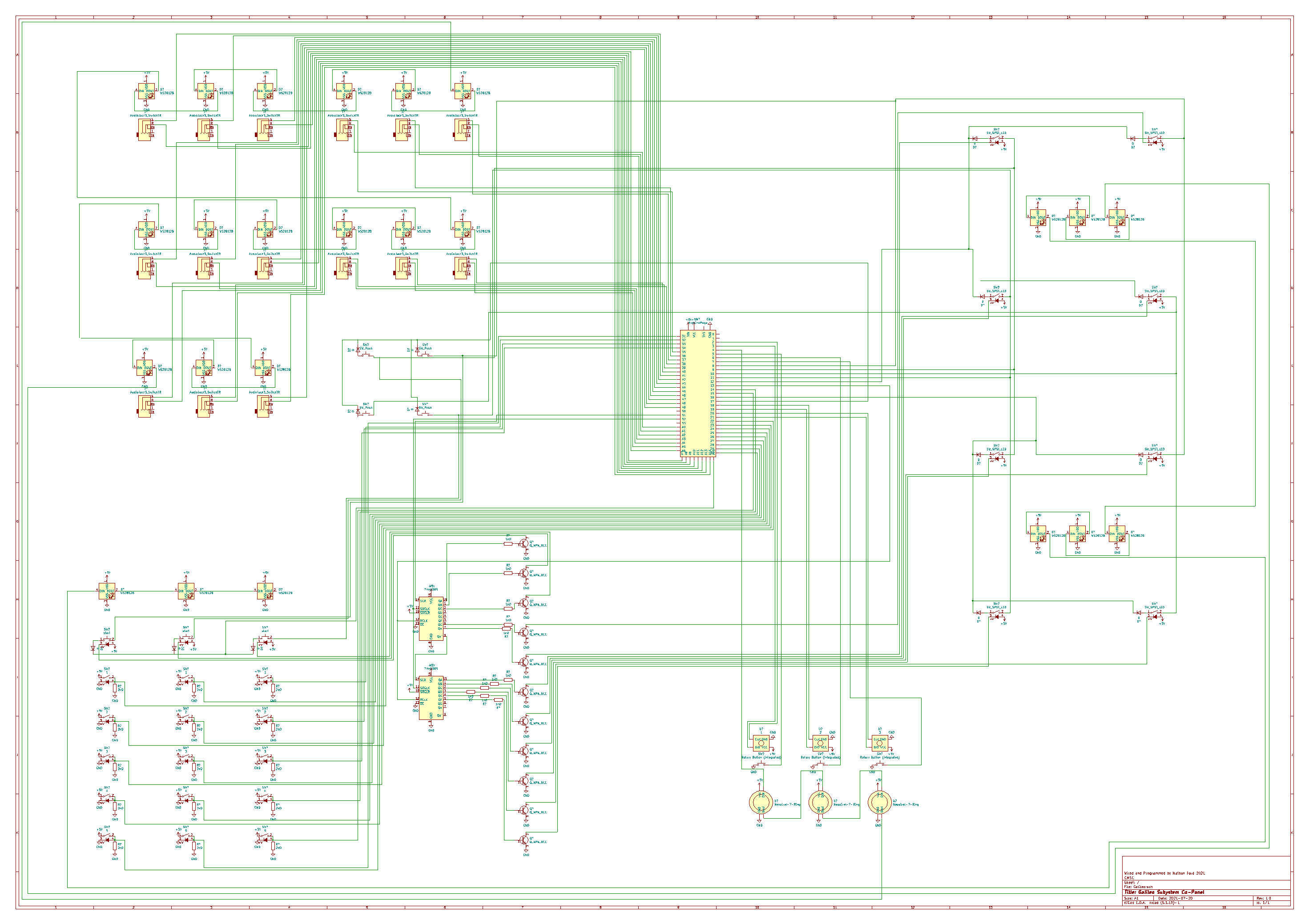
Galileo Subsystem

Co-Panel



# Table of Contents:

Dial Lights – Pg1

Program Lights – Pg2

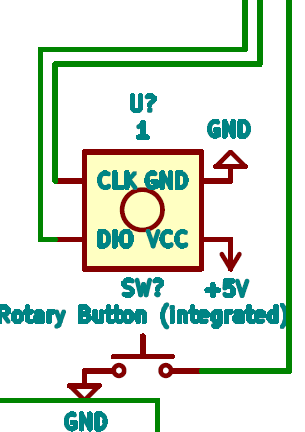
Dice Lights – Pg3

Jacks – Pg4-5

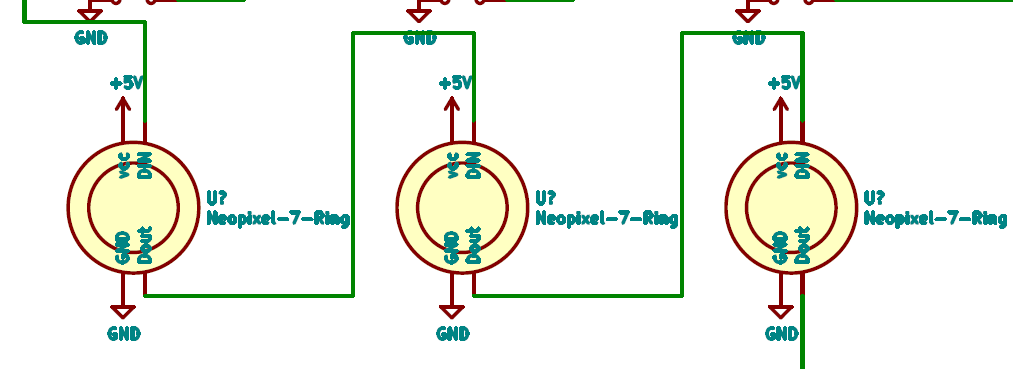
Shift Registers Pg6

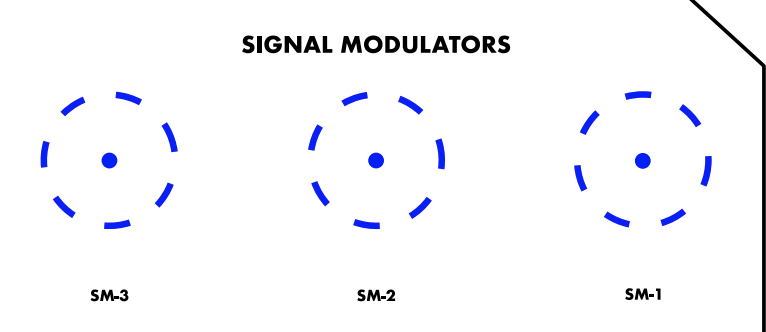
Dial Lights

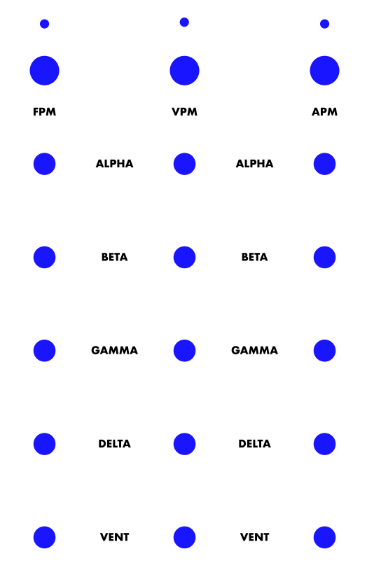
The dial lights (I guess they are officially called signal Modulators) give the user some nice kinesthetic interaction with the panel, as they turn the dial, they see the cursor in the circle move around and if they click the dial, they see where their cursor is lit up with that circle’s color.

1.  Rotary Encoders – these are how the user interacts with the circles, twist this, and the cursor moves. If the cursor is moving backwards, just swap CLK and DIO pins, on these rotary encoders with the PCB’s they are not true CLK and DIO, just called that. On these encoders when the dial is being turned it applies power to one pin then the other one or vice versa if you are twisting in the opposite direction. Please also note that the switch is part of the encoder, and you wont have to hook ground directly to it because you already gave the encoder GND, you just hook the SW pin to the proper one on the Arduino.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dial** | **Arduino Pin CLK** | **Arduino Pin DIO** | **Arduino Pin SW** |
| Left | 2 | 3 | 5 |
| Middle | 18 | 19 | 6 |
| Right | 20 | 21 | 7 |

1. Neo Pixel Rings – Now these are going to be the easiest Neo Pixels to wire, because the din on the first one comes from the Arduino, for the rest of these, you will have to connect to the master daisy chain. Just hook them all to 5v and ground, next you wire them from left to right, left DIN to Pin 4 on the arduino, Dout from left to middle DIN, and Dout from middle to right DIN, and then the Dout from the right one will go to the NEOpixels for the jacks. To make the neopixels look nicer we took a clear 3.5mm led, snipped off the leads, superglued it to the metal of the panel, and then hot glued a flat neopixel to the back of it, that way it looks nice like a 3.5mm led, Is cheaper than buying 3.5mm Neo Pixels, and near indestructible with the super glue.

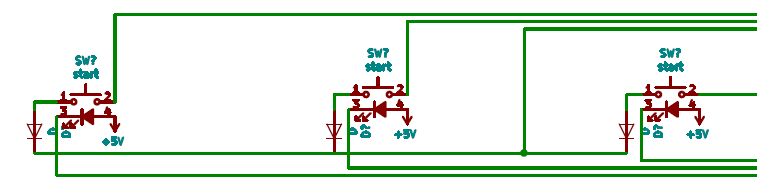


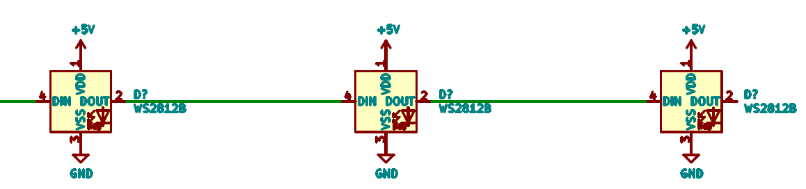
Program Lights

These are lights that you get to “program” yourself, you set the rockers to either on or off, and then hit the “run program” button and watch the Neo Pixel do fun patterns depending on the switch combo. This is my personal favorite part of this panel.

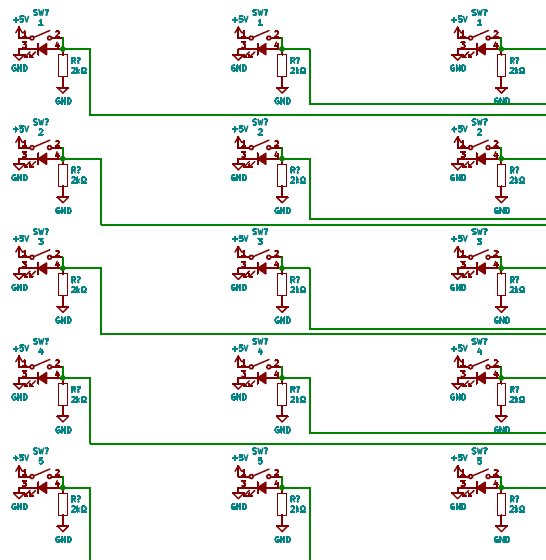
1. Program Run Buttons (FPM VPM APM) – These are non-latching buttons with lights in them. The lights are attached to 5v and an NPN transistor powered by a shift register.

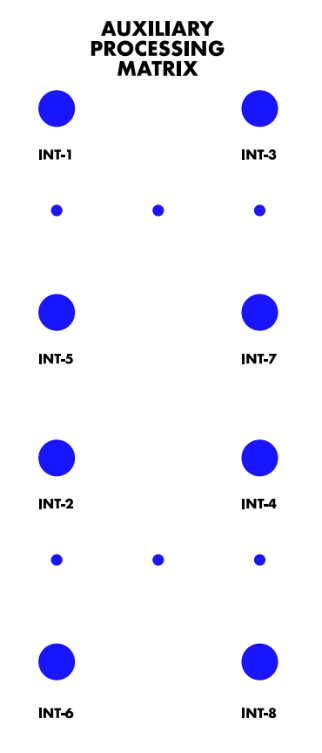
|  |  |  |  |
| --- | --- | --- | --- |
| **Switch** | **Light Pin -** | **Inside pin** | **Outside pin** |
| Left | Shift Register 1 Qa | A15 | 8 |
| Middle | Shift Register 1 Qb | A15 | 9 |
| Right | Shift Register 1 Qc | A15 | 10 |



1. Neo Pixels – these are daisy chained from left to right, and these are the last thing in the big, long strand! And so, the Dout on the right most one goes nowhere! The Din for these comes from the dout on the Dice. Again, remember to give them 5v and GND.

|  |  |  |  |
| --- | --- | --- | --- |
| **Row** | **Left Column Pins** | **Middle Column Pins** | **Right Column Pins** |
| **1** | 22 | 27 | 32 |
| **2** | 23 | 28 | 33 |
| **3** | 24 | 29 | 34 |
| **4** | 25 | 30 | 14 |
| **5** | 26 | 31 | 15 |

1.  Toggle Switches – these are all the different functions the program can do. The switches that were used on the original were actually switches that you applied 5v to, gnd for the built in LED and then you could take out a signal of 5v off the third conneciton, so you have to get some pulldown resistors for these because the arduiono only has integrated pullup resistors, 2-10k Ohms should work nicely

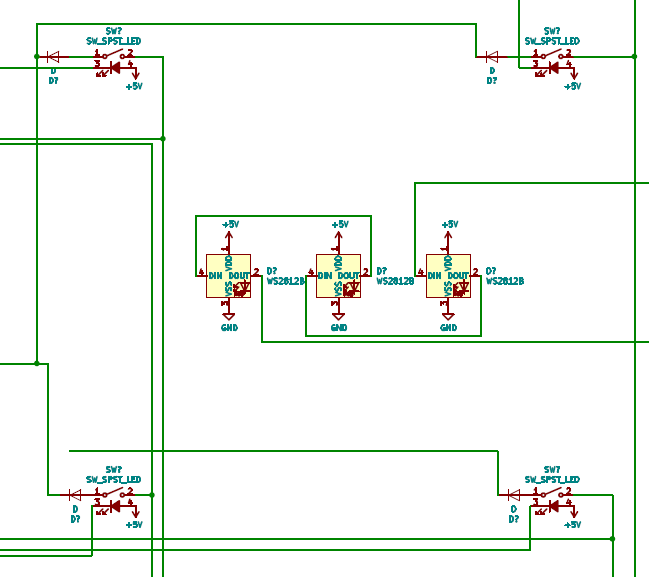
Dice Lights

These are the lights with 4 latching SPDT light up lights plus 3 Neo Pixels per die. Depending on what switches are pressed (16 possible combos) it will change the colors of the Neo Pixels and what lights are on and off in each light and Neo Pixel.

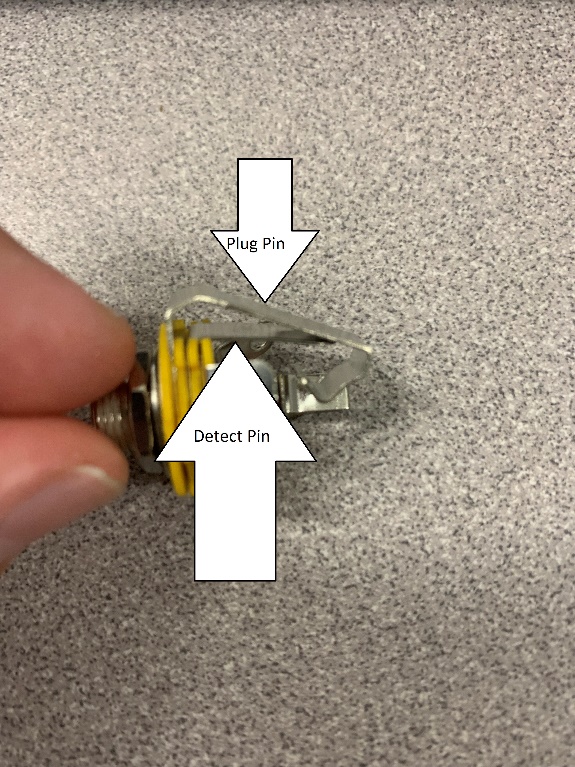
1. The Buttons – These do not necessarily have to be SPDT (single pull double throw), SPST (single pull single throw) would also work simply fine, just the ones that Mr. Porter ordered were SPDT. For the switch both sides go to a pin on the Arduino, for the light, just attach one pin to 5v, and the other to the shift register.

|  |  |  |  |
| --- | --- | --- | --- |
| **Button** | **Switch Pin 1** | **Switch Pin 2** | **Light Pin** |
| Int-1 | 12 | 9 | SR 1 Qg |
| Int-3 | 12 | 8 | SR 1 Qh |
| Int-5 | 12 | 11 | SR 2 Qa |
| Int-7 | 12 | 10 | SR 2 Qb |
| Int-2 | 16 | 9 | SR 2 Qc |
| Int-4 | 16 | 8 | SR 2 Qd |
| Int-6 | 16 | 11 | SR 2 Qe |
| Int-8 | 16 | 10 | SR 2 Qf |



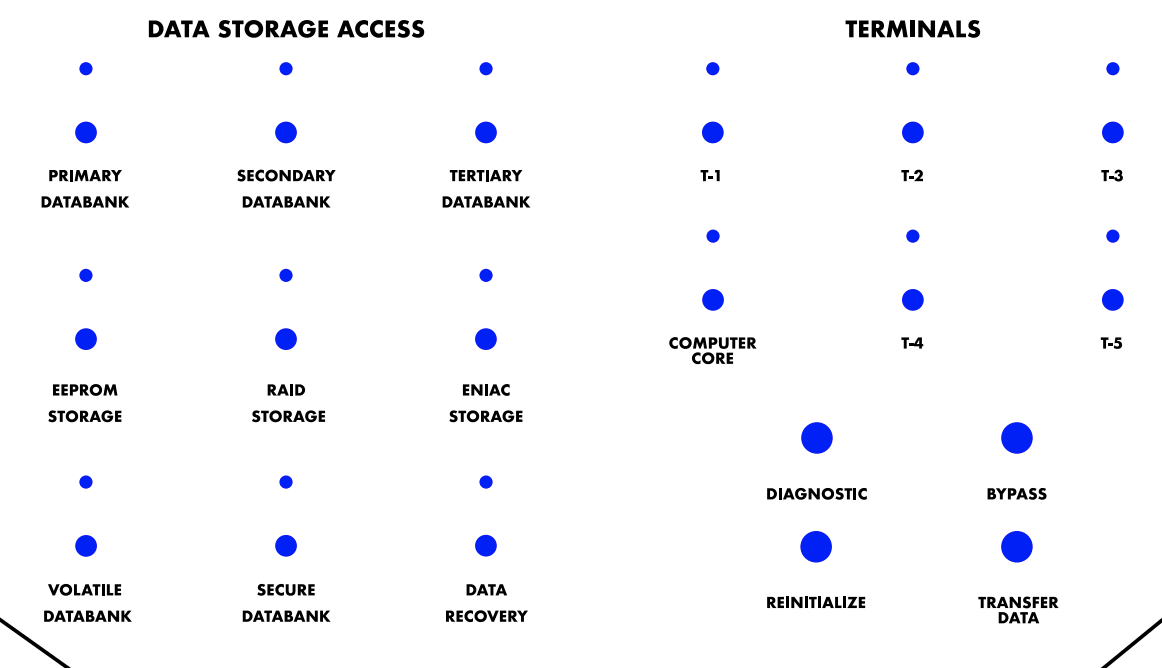
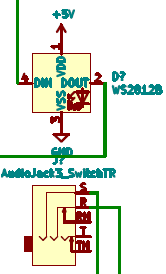
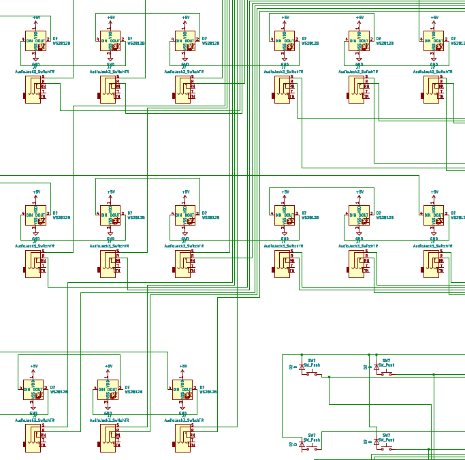
1. The Neo Pixels – These are not to bad to wire, they go from Right to Left (if looking from front of panel, from left to right if looking from behind) top to bottom. They all also need 5v and GND (big surprise, I know) The Din for the first one comes from the Dout of the last Jack Light, and the Dout of the last die light goes to the Din of the program lights. We also use the 3.5 mm clear led hack here.

Jacks

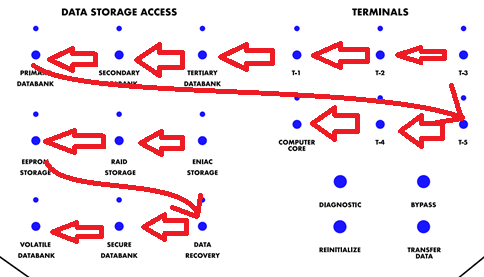
 This is the hardest part of this entire panel to wire, because of the shear number of wires needed to do so, if you notice, all the pins on the Arduino are being used on this panel, besides MISO and pin 53, and the reason these are not in use at this moment is that they are reserved for the future SPI Ethernet Module for integration for thorium.

1. The Jack Definition – This is an especially important part of the wiring, that is knowing what pins to wire to. The Schematic symbol for the audio jack is like the real one, but I could not find an exact match, so as you can see on the picture on the right, you have your jack detect and your plug pin parts of it, you should then pull the outer part apart, so they are not touching (or plug a cable in) and use a multimeter to find the correct terminals for those connections. **If you get this backwards the panel will not work!**

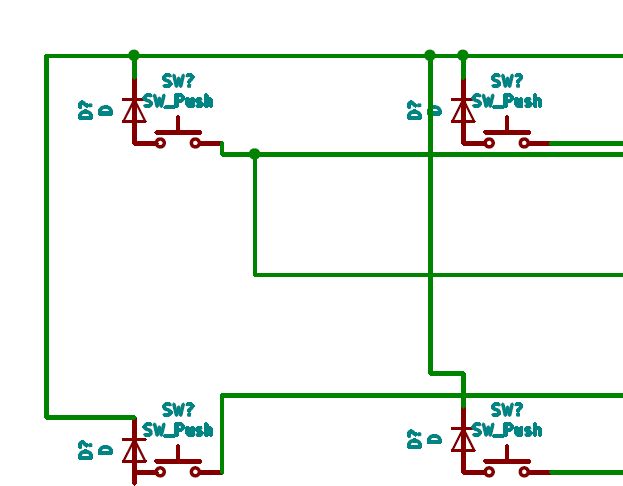
|  |  |  |
| --- | --- | --- |
| Jack | Plug Pin | Detect Pin |
| T1 | 35 | A0 |
| T2 | 36 | A1 |
| T3 | 37 | A2 |
| Computer Core | 38 | A3 |
| T-4 | 39 | A4 |
| T-5 | 40 | A5 |
| Primary Databank | 41 | A6 |
| Secondary Databank | 42 | A7 |
| Tertiary Databank | 43 | A8 |
| EEPROM Storage | 44 | A9 |
| RAID Storage | 45 | A10 |
| ENIAC Storage | 46 | A11 |
| Volatile Databank | 47 | A12 |
| Secure Databank | 48 | A13 |
| Data Recovery | 49 | A14 |



Jacks Pt2

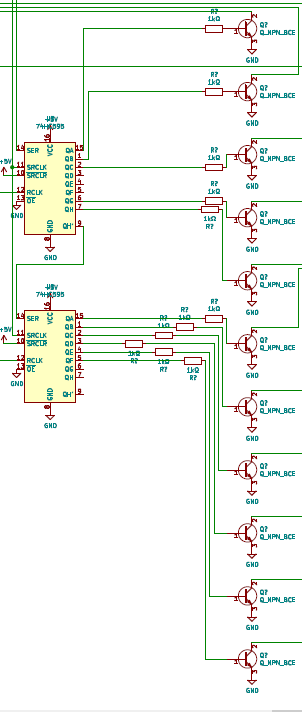


1. Neo Pixels – These are wired simply, left to right top to bottom from the back, or right to left from the front (refer to picture). The Din on the first one comes from the Dial lights, and the Dout on the last one goes to the Dice lights. Again, they all need 5v and GND.
2. Function Buttons – These are the buttons that tell what the computer should do with the connection that the user made. They have no lights, so they are super simple to hook up.



|  |  |  |
| --- | --- | --- |
| **Button** | **Button Side 1 Pin** | **Button Side 2 Pin** |
| Bypass | 17 | 8 |
| Diagnostic | 17 | 9 |
| Transfer Data | 17 | 10 |
| Reinitialize | 17 | 11 |

Shift Registers

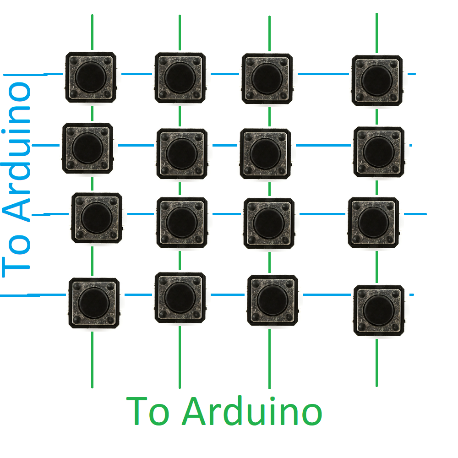


|  |  |
| --- | --- |
| **Pin on Chip 1** | **Connection to** |
| 1 | Qb |
| 2 | Qc |
| 3 | N/C |
| 4 | N/C |
| 5 | N/C |
| 6 | Qg |
| 7 | Qh |
| 8 | GND |
| 9 | SER on Chip 2 |
| 10 | 5v |
| 11 | Pin 52 Arduino |
| 12 | Pin 13 Arduino |
| 13 | GND |
| 14 | Pin 51 Arduino |
| 15 | Qa |
| 16 | 5v |

Because of our lack of free pins (as I said above in the jack section) we had to do some compressing to make it all fit. One of these solutions was to use 2 shift registers to give us 16 extra outputs, I attached resistors with NPN transistors to each of these to make it so that we do not put too much strain on the shift registers I used the resistors to make sure that it stayed under the rated amps of the shift registers I was using. (In theory 250 ohms should work, but 1k does not hurt). I decided to save pins even further when wiring these up, so I used MOSI for the shift register and SPI CLK for the register clock line so effectivly I only sacrifice one pin (Chip select) to gain 11 more outputs.

|  |  |
| --- | --- |
| **Pin on Chip 2** | **Connection to** |
| 1 | Qb |
| 2 | Qc |
| 3 | Qd |
| 4 | Qe |
| 5 | Qf |
| 6 | N/C |
| 7 | N/C |
| 8 | GND |
| 9 | SER on Chip 2 |
| 10 | 5v |
| 11 | Pin 52 Arduino |
| 12 | Pin 9 on Chip 1 |
| 13 | GND |
| 14 | Pin 51 Arduino |
| 15 | Qa |
| 16 | 5v |

## Button Grid

 As you may have noticed, for the buttons on the die, the jacks and the program section have neither side connected to ground, this is because to conserve pins on the Arduino, I created button grids so instead of needing 15 pins, you only need 8. The easiest way to explain this is by comparing it to a grid you have horizontal and vertical pins, you set one row at a time low, while keeping the rest high, and read which columns also turn low, indicating a connection, you store this and move on. You should also put diodes on one side of each switch too, to keep it from ghosting, this is an especially bad problem when you have locking switches like we have, so technically they aren’t needed anywhere else, in fact in the first version I made, they are only on the locking switches, but they are there in the schematic in case anybody wants to try that out.