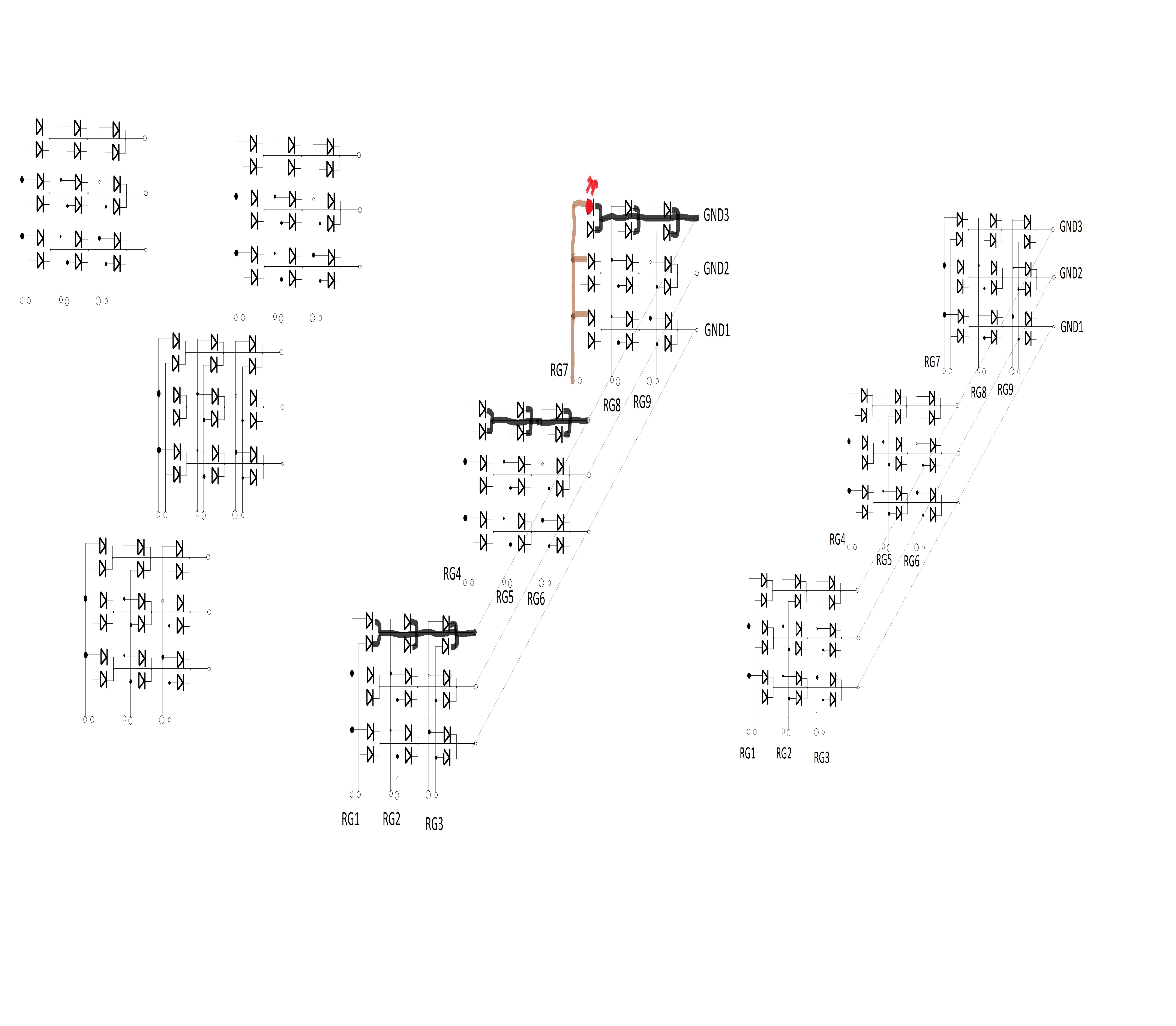
# Information about LED cubes

First off I think we should know how this devices work. They are made of panels of LED diodes connecting them to a common GND or Vcc+. When we connect LEDs in common cathode (-) mode, all shorter legs are soldered together (or just united in some way) and when applying voltage to the other “free leg” it will light up only one LED. To light all the LEDs we would have to give power to each of the anodes one by one or at the same time.

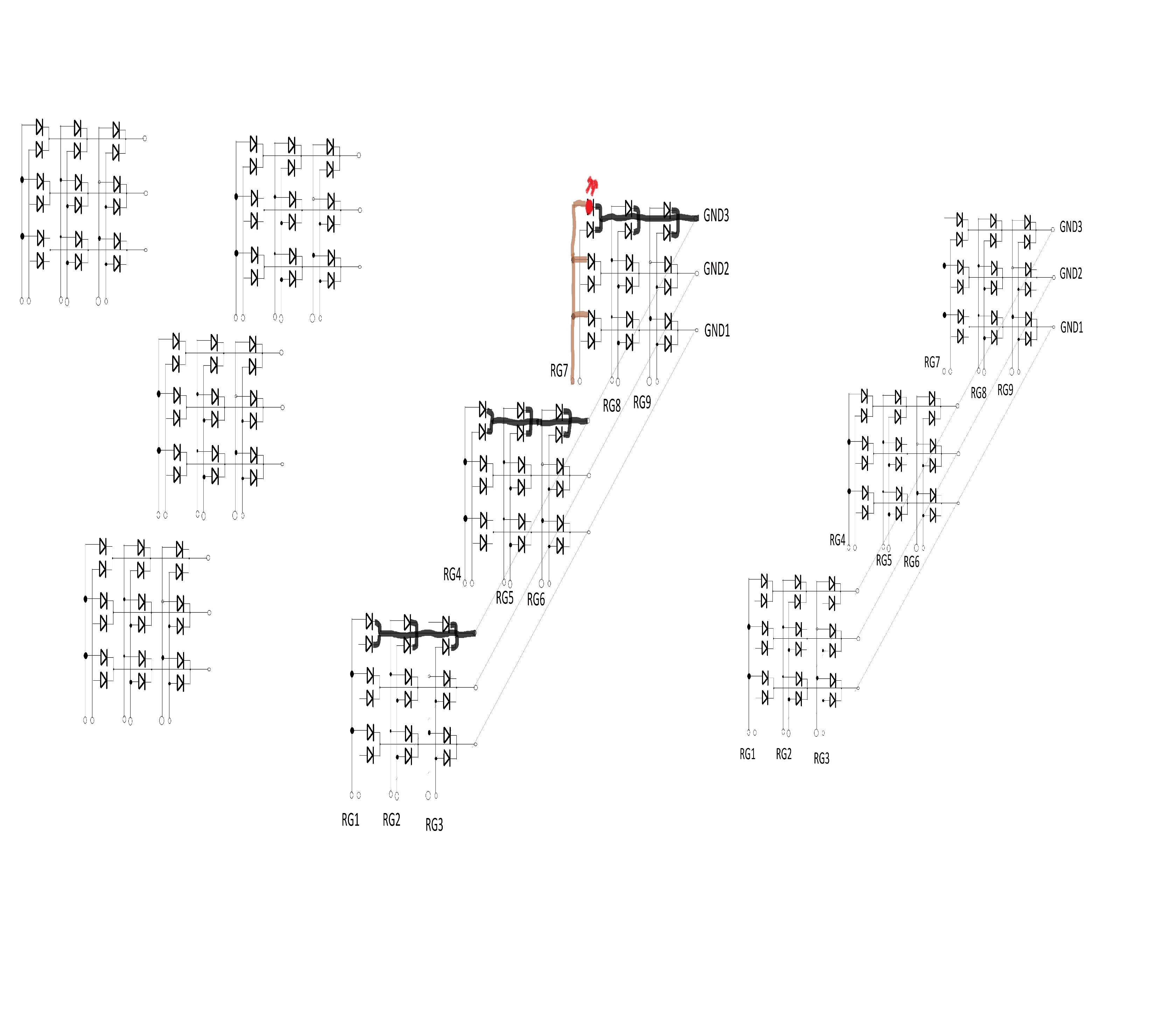
# Dual color LED cube working

We decided to use bi colour LEDs to show different information easily. These LEDs work like if they were two LEDs per each actual LED. Therefore for each column we will have two anodes, one that connects the red LED and the other the green LED. They are common cathode, meaning the gnd is shared.

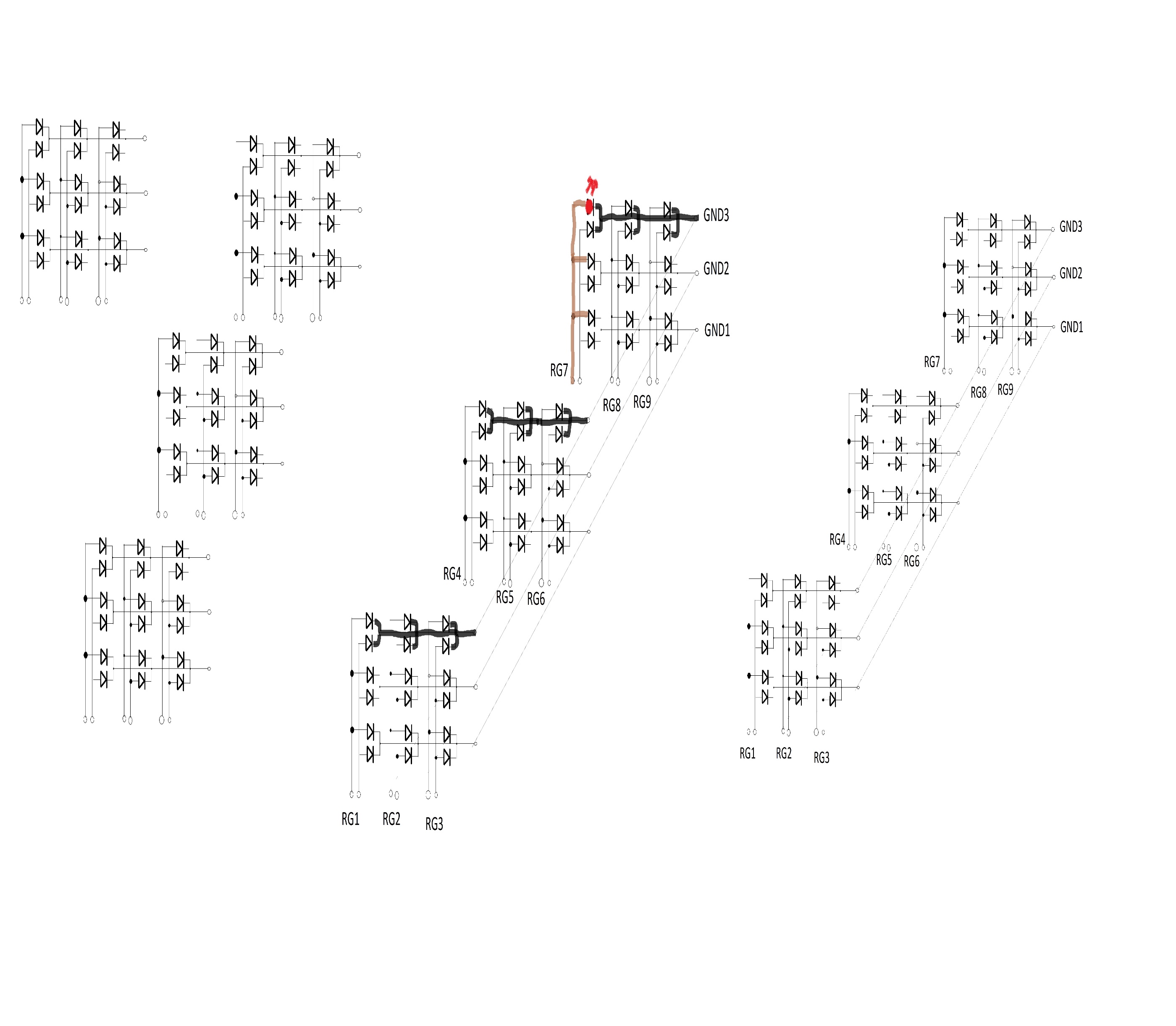
A 3x3x3 cube can be separated in 3 vertical slices by 3 horizontal layers. To simplify the explanation, let’s take a look into one of the vertical slices.

In the picture above, the vertical lines (columns) are the two anodes we get and the horizontal lines (rows) are the common GND. In this way, a particular LED will light up only when it’s row is connected to GND and the column set to HIGH (or connected to the power supply). To light up two different rows we would have to connect each row to GND, but by doing so we would light up the LED corresponding to that row at the column we light up before and we might not want that.

To avoid this, we use multiplexing. We send the data for what LED we want to turn on in the first row and connect the first row to GND. After some miliseconds we change the row connected to GND and send new information, now about which LEDs we want to turn on from the next row. And so on, repeating again and again depending on the code we had uploaded.

Then, we would need 2 control pin for each column to control the LED turned on and the color, and 1 control pin for each row. For only this one slice that would make 6 control pin (columns) + 3 control pins (rows) = 9 pins from out microcontroller. That seems feasible. However we are not trying to make a 3x1x3 LED cube (if we would call that a cube) so two more slices have to go in.

Now we have something that starts to look like a cube… sort of. I hope the picture above is clear enough to understand, we simply added to more vertical slices to the previous one. Also, a connection between the GND from each of the rows has been made, creating a layer (GND1, GND2, GND3). Now, by connecting each layer to ground we get the same result that we got for the row in 2D explanation but in 3D. So to light up the red LED of the upper left corner of the last vertical slice, we would have to set HIGH R7 and LOW GND3.

By doing so we are also connecting the corresponding layer (upper layer) entirely connected to GND (as the picture can hopefully show) and the 7th column to HIGH. However only our red LED on the upper left corner is turned on, since it is the only one completing the circuit. If we would want to connect the middle green LED from the same slice, the one above it and the one to it’s left would light up too. Again, as in 2D we would have to multiplex between the layers.

What about the number of pins needed now? Has it changed? Well for the layer/row control it remains the same: 3 control pin one for each layer. However, we have 6 more columns, with two new pins each and those need an individual connection for themselves, meaning 12 new control pins for LED and color. Now we need 18 pins for LED control (column control) and 3 pins for layer control (GND). We intend on adding a sensor to serve as an input of data for the cube to change its color and animation, therefore we need 2 pins for I2C control. In total 23 pins of a microcontroller. The Arduino Uno (microcontroller to be used) has 14 digital I/O pins and 6 analog I/O, a total of 20 pins which is 3 less than what we need. Then we would have to add another microcontroller for the last 3 pins which, let’s say, control the GND multiplexing. OR we could add shift registers that can take 3 pins of the Arduino and control 8, 16 or more outputs (depending on the shift register). This would get the number of used pins low enough to be handled by one single Arduino UNO, which is what we have in hand (also, buying a shift register is cheaper than a new microcontroller).

A new question arises then, what is a shift register, what does it do and how does it work with the cube?