LED MATRIX

What you need:

1. 2 breadboards
2. 1 Breakout board kit
3. 2 Raspberry Pi port expanders kits
4. 8 150 Ohm resistors
5. 8 330 Ohm resistors
6. 25 jumper wires
7. 1 GPIO cable
8. 1 LED Matrix
9. 1 Capacitor (optional)

Instructions:

1. Leave the breadboard aside for now and take a look at the packages with the breakout board and the port expanders - the breakout board package is the one without the chip
2. Time for soldering - A very good guide through the process could be found here: https://www.modmypi.com/blog/rpi-mcp23017-port-expander-board-kit-assembly-guide
    a. Now follow the same process for the second port expander
    b. To connect the two port expanders take the 5 pin right angled male header pins and use it to connect them the same way we did the breakout board and the 1st port expander - make sure SDA is connected to SDA,SCL to SCL, 0V - 0V, 3V - 3V, 5V - 5V

3. Place the three connected boards on the breadboard
4. Use the GPIO cable to connect the breakout board to the Raspberry Pi
5. Use a wire to connect #0 and 3V on the second port expander - that gives the two port expander different addresses
6. Turn on your Raspberry Pi - we need to configure the I2C bus
    a. Enter from a shell sudo nano /etc/modprobe.d/raspi-blacklist.conf and put a hash mark '#' at the beginning of line 'blacklist i2c-bcm2708'
    b. Edit the modules - enter sudo nano /etc/modules and add the following line 'i2c-dev' (without the ''!)
    c. Next we need to install some python libraries and tools - type:
        i. sudo apt-get update
        ii. sudo apt-get install python smbus
        iii. sudo apt-get install i2c-tools
        iv. add a new user to the i2c group - sudo adduser pi i2c
7. Restart the Pi and then type in a shell 'sudo i2cdetect -y 1' (for older Raspberry Pies use 0) - you should see something like this
    a. In case you skipped step 6 and you see only this ... connect #0 and 3V on the second port expander now and try 'sudo i2cdetect -y 1' again you should see something like this
    b. To better our I2C bus performance we could add some pull down resistors - keep in mind that there are some pull down resistors onto the Raspberry Pi so most of our

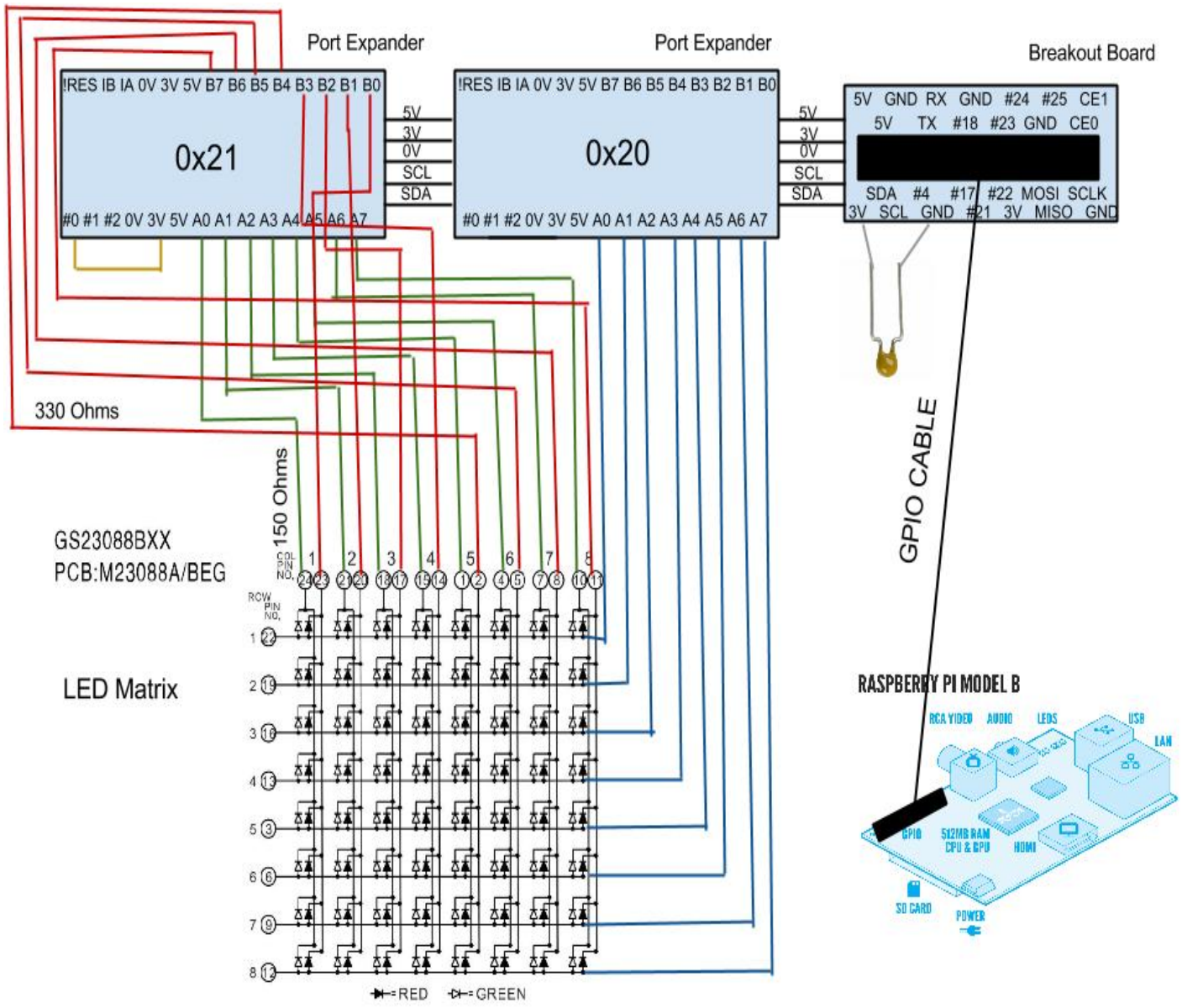
job is done if you want you can add some like this

c. You can also add a capacitor between GND and 3V to smooth the current.
d. Now the LED matrix - Depending on which king of matrix you order the connections may be different - this is one sample matrix pins - 24 in total - 12 on each side. It is advisable to download the datasheet and examine as to what each pin corresponds. For example:
e. In this example case the matrix is bi-colour, so it has both green and red LEDs. Now we need some resistors to limit the current and keep our matrix safe. The reason they are different for red and green LEDs is the amount of current the red one needs in comparison to the green one. You can check here for more details about the voltage drop on the LEDs depending on their colour. Playing with different resistors will get you differently bright LEDs.

!!!

Keep in mind you might blow them up if you supply to much current! Do not forget Ohm's law!
The resistors on on the red LEDs influence the green as well - they are connected in parallel.

A picture of the connection:



Now comes the software part - Finally!!! (I know some of you thought we would never get to this part, but here we are!)
The modules you need to install are

- pygame - it comes pre-installed on the Pi
- numpy - sudo apt-get install python-numpy
- smbus - you have installed that already(see step 7)
- Bicolour_Interface - my own library for easy manipulation of the matrix - download it from here - https://github.com/haralav2/LEDMatrix/blob/master/Bicolour_Interface.py

You can download a GUI to interface - https://github.com/haralav2/LEDMatrix/blob/master/CombinedMatrixNewTesting.py the display as well as some libraries to do simple matrix interfacing as well.
What about some text displaying - I am glad you asked - look here - https://github.com/haralav2/LEDMatrix/blob/master/TalkToPiLight.py. Now I am not going to tell you everything you can use it for - good thing is you can keep developing as much as you want! There is some guide to the code as well!