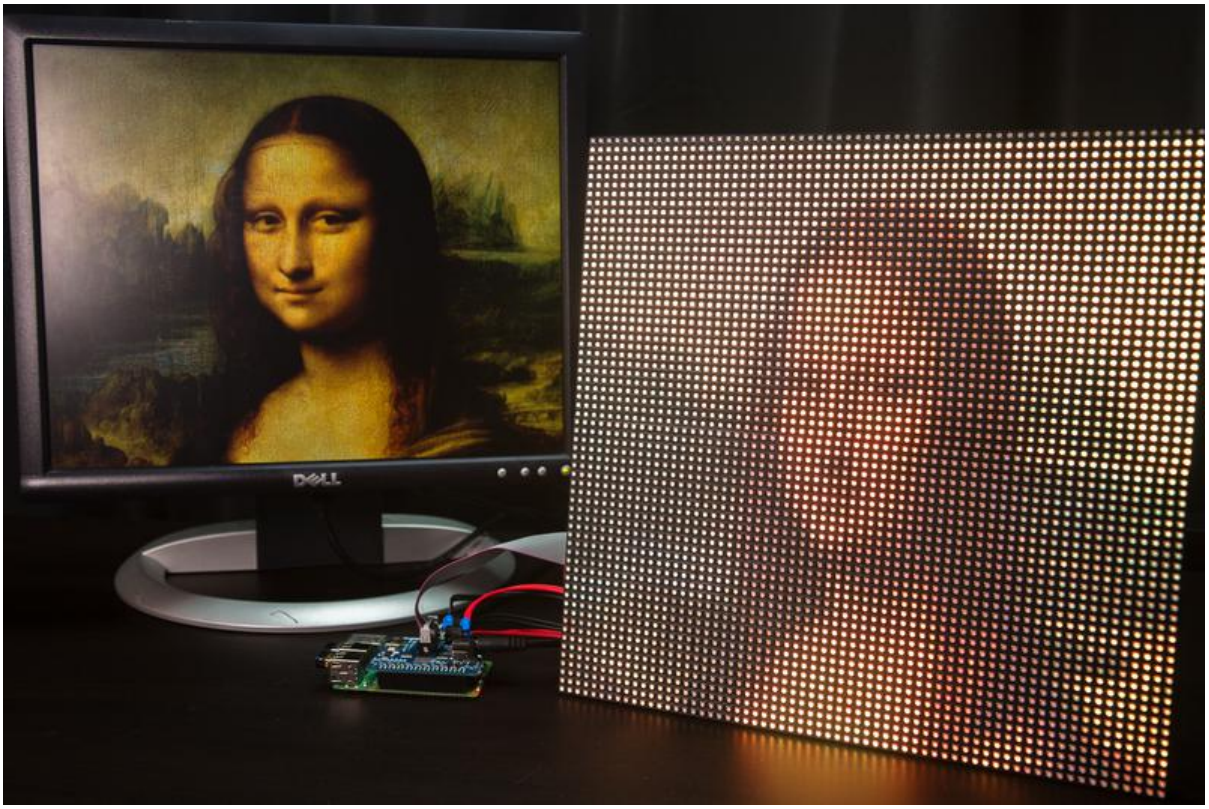
Raspberry Pi LED Matrix Display

Created by Tony DiCola



https://learn.adafruit.com/raspberry-pi-led-matrix-display

Last updated on 2022-12-01 02:39:43 PM EST

Table of Contents

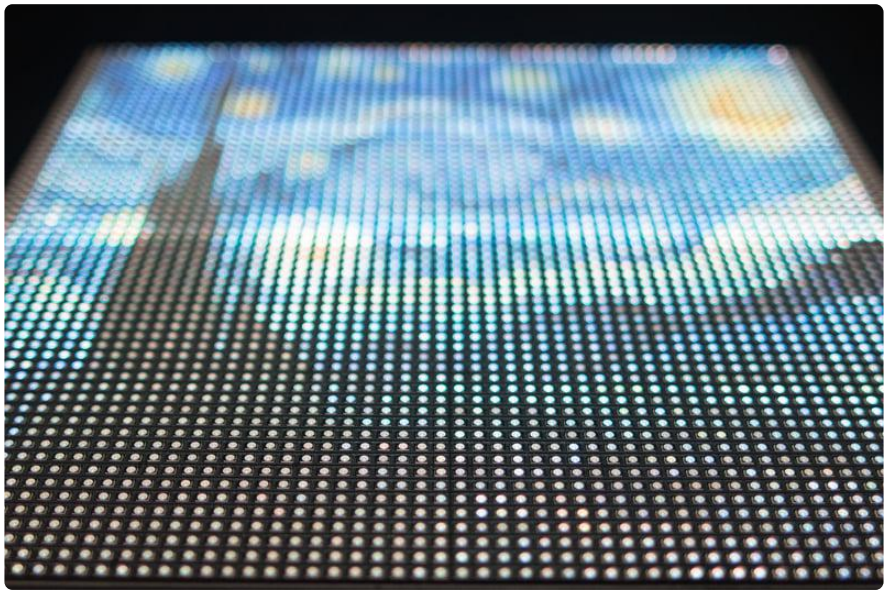
Overview

This tutorial does not work with the Pi 4 - no ETA on when it may be fixed!

This project will show you how to turn RGB LED matrices () into a display for the Raspberry Pi! You can play games, watch movies, display a dashboard of data, and much more on a big and beautiful LED display. Anything the Pi outputs to a monitor can be shrunk down and displayed on LED matrices!

The code for this project uses the excellent rpi-rgb-led-matrix library (). This library allows the Pi to light up and display graphics on LED matrices. Combined with a little bit of extra code to read the Pi's video output you'll have a dedicated LED matrix display setup with ease.

The Adafruit LED matrix HAT (http://adafru.it/2345) also makes it easy to connect RGB LED matrices to the Pi. The HAT takes care of level conversion and power distribution so you can just plug in displays and go.



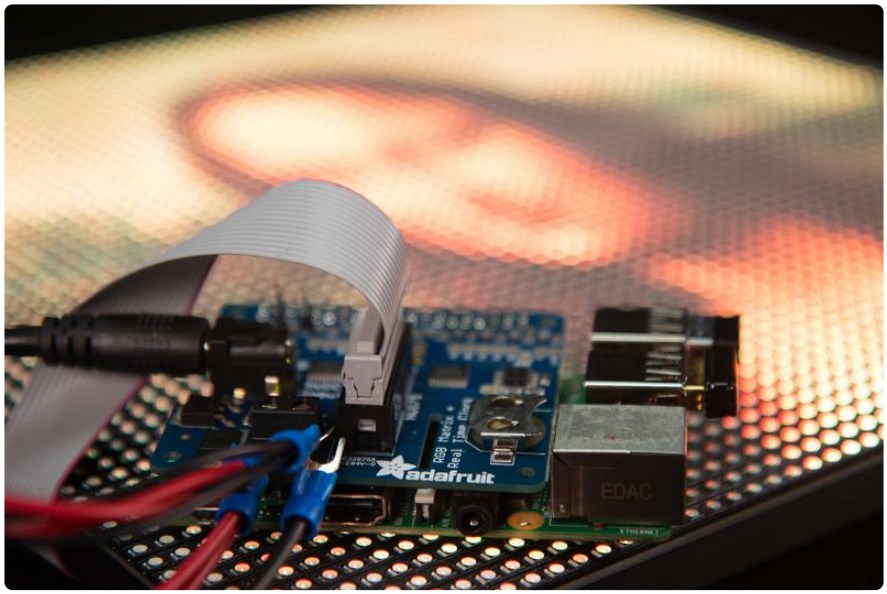
Before you get started you'll want to be familiar with LED matrices and the Raspberry Pi by reading these guides:

- 16x32 and 32x32 RGB LED Matrix Guide ()
- Adafruit RGB Matrix HAT ()

In addition if you're new to the Raspberry Pi check out these Learn Raspberry Pi guides () to learn how to get started by loading an operating system and connecting to the Pi's command line terminal.

Also realize the Raspberry Pi makes a 'best effort' at driving these LED matrices. There's no dedicated fast signal generation on the Raspberry Pi so you're sometimes at the mercy of the Linux kernel when talking to hardware like these panels. This means you might see some slight flickering or other small display artifacts. Using a Raspberry Pi 2 and keeping the load on the processor low will help reduce graphical glitches.

Hardware



Parts

You'll need the following parts to build this project:

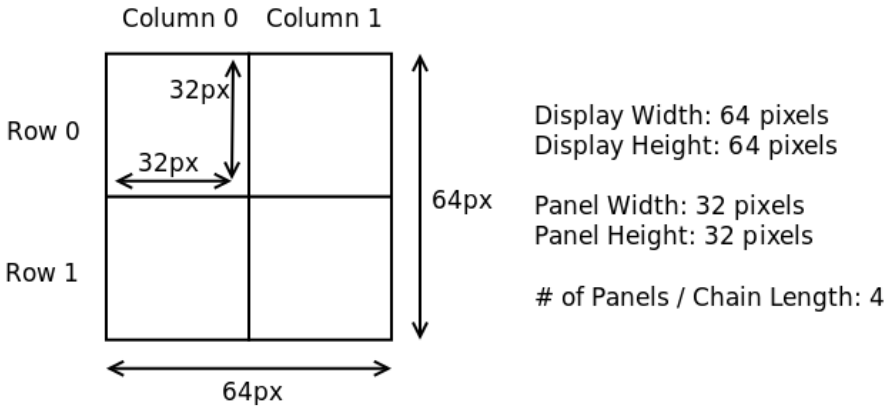
- Raspberry Pi 2 () - You can in theory use a less powerful Raspberry Pi like the Raspberry Pi B+ or even the new Raspberry Pi Zero, however this project greatly benefits from the multiple cores of the Pi 2. If you're using a less powerful Pi you'll see more glitching and issues with rendering the display.
- LED Matrix HAT (http://adafru.it/2345) - The Pi LED Matrix HAT makes it easy to connect LED matrices to the Pi. Alternatively you can wire panels directly to GPIO on the Raspberry Pi, see the rpi-rgb-led-matrix library () for more information on manually wiring displays.

- RGB LED Matrix Panels () - These RGB LED panels come in sizes of 32x32 or 64x32 pixels and can be chained together to build a large display. For the best results with the Raspberry Pi 2 only about 12 panels can be chained together at once. To use more displays you'll need to manually wire displays up to parallel matrix channels, see the rpi-rgb-matrix () library for more information. Remember the longer 64x32 pixel panels actually count as two 32x32 pixel panels.
    - Note: You want the HUB75 RGB LED matrix panels, not the DotStar or NeoPixel panels! See the column on the right for links to all the appropriate panels.
- 5V Power Supply () - These LED panels can take a lot of current, up to 2 amps alone per panel, so make sure to get a power supply that can power all the panels. I recommend at least the 5 volt 10 amp supply for driving 4-5 panels.
- Soldering Tools () - You'll need tools to solder a header onto the LED matrix HAT. If you're new to soldering don't worry this is an easy soldering project, see the guide to excellent soldering ().
- Optional: 16-pin 0.1" spaced keyed female IDC connectors & 16-wire ribbon cable - If you're chaining multiple displays together and need a longer signal cable you can make one. Check local electronics stores or large distributors like DigiKey (), Mouser (), etc. for these parts.

Plan Your Display

If you're chaining together more than one display it will help to take a moment to plan out how the panels will be connected. The software for this project works by dividing the Pi's video output into a rectangular grid and assigning each grid square to a LED matrix panel.

For example a configuration of 32x32 pixel panels might look like the following:



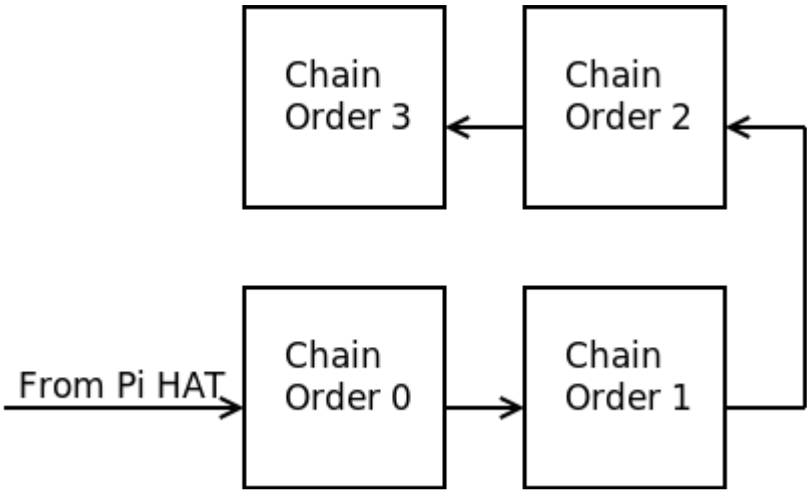
You can see four panels are combined to create a 64x64 pixel display. When the software runs it will take the Pi's video output and shrink it down to fit inside the 64x64 pixel display.

Note as an alternative to shrinking down the display you can instead configure the software to grab a tiny crop of the Pi's output. This is useful if there's only a very tiny portion of the screen you want to display, or if you're writing a program to display on the matrix and want pixel-perfect accuracy. With the configuration above a 64x64 pixel crop from anywhere on the Pi's video output could be displayed on the panels.

Keep in mind these constraints when planning your display:

- All panels must be the same LED/pixel width and height. Remember the wide 64x32 pixel panels are actually two 32x32 panels chained together so they're fine to mix with single 32x32 panels.
- The panels must be put together into a rectangle. You can make a square display, super wide, or even tall display but it must be a rectangular shape.
- Keep the total number of panels to 12 or less. On a single chain a Pi 2 can only drive about 12 panels. It is possible to drive more panels on parallel chains, but you'll need to consult the rpi-rgb-led-matrix library () and manually wire those chains.
- Match the aspect ratio of your display to the aspect ratio of the Pi's video output as close as possible. If you're displaying a program with a wide 16:9 aspect ratio (like a 1280x720 resolution) you will want to construct a similarly wide display of LED panels. This will reduce distortion from shrinking the Pi display down to the panels.

You'll also want to plan out how the display panels are wired. Each panel has an input and output connector. By connecting the output of one panel to the input of another panel you can construct a long chain of panels with minimal wiring. For the configuration above one wiring might look like:



The output of the Raspberry Pi is connected to the bottom left panel, then snakes around through all the panels and ends in the top left. The very first panel connected to the Pi is order/position 0 in the chain, the next panel is order/position 1, and so forth to the end of the chain. Keep in mind these chain order values as they'll be used later in the software configuration.

Assembly

To assemble the parts for this project just follow the RGB LED matrix HAT guide (). The guide will walk you through how to solder the header on the HAT and connect it to the Raspberry Pi and LED matrix panels. Be sure to follow the driving matrices () page to test each panel to confirm they're working before you move on.

Software

This tutorial does not work with the Pi 4 - no ETA on when it may be fixed!

To install the software for this project you'll first want to make sure your Raspberry Pi is running the latest version of the Raspbian operating system (). Using the Raspbian Jessie version is recommended.

You will also want to make sure your Raspberry Pi is connected to the internet through a wired or wireless connection as you follow the steps below. Software and dependencies will be downloaded from the internet and installed on the Pi.

Installation

Connect to a terminal on the Raspberry Pi using SSH and execute the following commands to download dependencies:

```
sudo apt-get update
sudo apt-get install -y build-essential git libconfig++-dev
```

Next download the code for this project from its home on GitHub () by running:

```
cd ~
git clone --recursive https://github.com/adafruit/rpi-fb-matrix.git
cd rpi-fb-matrix
```

Make sure to use the --recursive so that the required git submodule is installed too!

After the code has been downloaded you're ready to compile it. If you're using a Raspberry Pi 2 and the RGB LED matrix HAT you're ready to go and can skip down to the compile command. However if you're using an original Pi or manually wired your Pi to the matrices you might need to adjust some compilation options.

In the Makefile this line at the top sets compilation options for the rpi-rgb-led-matrix library:

```
# Configure the rpi-rgb-led-matrix library here:
# The -DADAFRUIT_RGBMATRIX_HAT value configures the library to use the Adafruit
# LED matrix HAT wiring, and the -DRGB_SLOWDOWN_GPIO=1 value configures the
# library to work with a Raspberry Pi 2.  For a Pi 1 (or perhaps even on a Pi 2,
# but I found it necessary in my testing) you can remove the -DRGB_SLOWDOWN_GPIO=1
# option.  You can also add any other rpi-rgb-led-matrix library defines here
# to configure the library for more special needs.  See the library's docs for
# details on options:
#   https://github.com/hzeller/rpi-rgb-led-matrix/blob/master/lib/Makefile
export DEFINES = -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1
```

Like the comments mention adjust the options as necessary for your setup. See the rpi-rgb-led-matrix library Makefile () for a description of available options.

To compile the code run the following command:

```
make clean all
```

The code should compile without error and finish with output similar to:

```
g++ -c -o rpi-fb-matrix.o rpi-fb-matrix.cpp -Wall -std=c++11 -O3 -I. -I./rpi-rgb-
led-matrix/include -I/opt/vc/include -I/opt/vc/include/interface/vcos/pthreads -I/
opt/vc/include/interface/vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -L./
rpi-rgb-led-matrix/lib -L/opt/vc/lib
g++ -c -o GridTransformer.o GridTransformer.cpp -Wall -std=c++11 -O3 -I. -I./rpi-
rgb-led-matrix/include -I/opt/vc/include -I/opt/vc/include/interface/vcos/pthreads -
I/opt/vc/include/interface/vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -
L./rpi-rgb-led-matrix/lib -L/opt/vc/lib
g++ -c -o Config.o Config.cpp -Wall -std=c++11 -O3 -I. -I./rpi-rgb-led-matrix/
include -I/opt/vc/include -I/opt/vc/include/interface/vcos/pthreads -I/opt/vc/
include/interface/vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -L./rpi-rgb-
led-matrix/lib -L/opt/vc/lib
make -C ./rpi-rgb-led-matrix/lib
make[1]: Entering directory '/home/pi/rpi-fb-matrix/rpi-rgb-led-matrix/lib'
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o gpio.o gpio.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o led-matrix.o led-matrix.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o framebuffer.o framebuffer.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o thread.o thread.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o bdf-font.o bdf-font.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o graphics.o graphics.cc
g++ -I../include -Wall -O3 -g -fPIC -DADAFRUIT_RGBMATRIX_HAT -DRGB_SLOWDOWN_GPIO=1 -
DRGB_SLOWDOWN_GPIO=1 -c -o transformer.o transformer.cc
ar rcs librgbmatrix.a gpio.o led-matrix.o framebuffer.o thread.o bdf-font.o
graphics.o transformer.o
make[1]: Leaving directory '/home/pi/rpi-fb-matrix/rpi-rgb-led-matrix/lib'
g++ -o rpi-fb-matrix rpi-fb-matrix.o GridTransformer.o Config.o rpi-rgb-led-matrix/
lib/librgbmatrix.a -Wall -std=c++11 -O3 -I. -I./rpi-rgb-led-matrix/include -I/opt/
vc/include -I/opt/vc/include/interface/vcos/pthreads -I/opt/vc/include/interface/
vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -L./rpi-rgb-led-matrix/lib -L/
opt/vc/lib -lrgbmatrix -lrt -lm -lpthread -lbcm_host -lconfig++
g++ -c -o display-test.o display-test.cpp -Wall -std=c++11 -O3 -I. -I./rpi-rgb-led-
matrix/include -I/opt/vc/include -I/opt/vc/include/interface/vcos/pthreads -I/opt/
vc/include/interface/vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -L./rpi-
rgb-led-matrix/lib -L/opt/vc/lib
cc    -c -o glcdfont.o glcdfont.c
g++ -o display-test display-test.o GridTransformer.o Config.o glcdfont.o rpi-rgb-
led-matrix/lib/librgbmatrix.a -Wall -std=c++11 -O3 -I. -I./rpi-rgb-led-matrix/
include -I/opt/vc/include -I/opt/vc/include/interface/vcos/pthreads -I/opt/vc/
include/interface/vmcs_host -I/opt/vc/include/interface/vmcs_host/linux -L./rpi-rgb-
led-matrix/lib -L/opt/vc/lib -lrgbmatrix -lrt -lm -lpthread -lbcm_host -lconfig++
```

That's all there is to compiling the project code! Once the code has compiled you're ready to start configuring and testing the display.

If the compilation fails with an error go back and check that all of the previous steps have been completed, and that you're using the Raspbian Jessie operating system.

Configuration

The software for this project is configured with a simple text configuration file. An example configuration is provided in the matrix.cfg file (). You can use this file as a base and edit it for your specific display.

Open the file with the nano text editor by running:

```
nano matrix.cfg
```

You can scroll through the file and read a description of each configuration value in the comments (comments are lines that start with a // and won't be interpreted as configuration).

The first two settings define the total pixel width and height of the display:

```
// Define the entire width and height of the display in pixels.
// This is the _total_ width and height of the rectangle defined by all the
// chained panels.  The width should be a multiple of the panel pixel width (32),
// and the height should be a multiple of the panel pixel height (8, 16, or 32).
display_width = 64;
display_height = 64;
```

The next settings define the pixel width and height of an individual panel:

```
// Define the width of each panel in pixels.  This should always be 32 (but can
// in theory be changed).
panel_width = 32;

// Define the height of each panel in pixels.  This is typically 8, 16, or 32.
// NOTE: Each panel in the display _must_ be the same height!  You cannot mix
// 16 and 32 pixel high panels for example.
panel_height = 32;
```

The chain length setting configures the number of panels that are connected together into a single chain:

```
// Define the total number of panels in each chain.  Count up however many
// panels are connected together and put that value here.  If you're using
// multiple parallel chains count each one up separately and pick the largest
// value for this configuration.
chain_length = 4;
```

The parallel count defines how many chains of panels are connected to the Pi in parallel. Remember the Adafruit HAT only supports one parallel chain.

```
// Define the total number of parallel chains.  If using the Adafruit HAT you
// can only have one chain so stick with the value 1.  The Pi 2 can support up
// to 3 parallel chains, see the rpi-rgb-led-matrix library for more information:
//   https://github.com/hzeller/rpi-rgb-led-matrix#chaining-parallel-chains-and-
coordinate-system
parallel_count = 1;
```

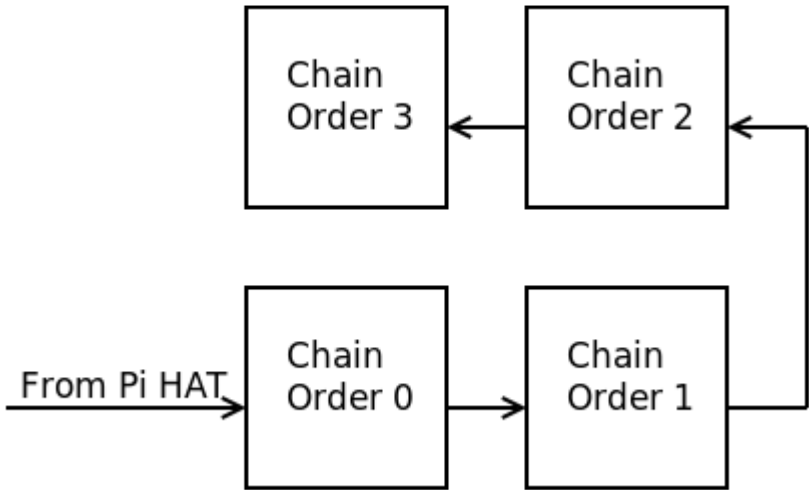
Next the individual panels of the display are configured by the panels setting. This setting is a two-dimensional list of panel configurations. Each panel configuration

defines which panel will display that portion of the Pi's screen. For example the configuration in the first row and first column will define which LED panel displays the top left portion of the screen.

Panels are identified by their order along the chain of panels. The first panel connected to the Pi is order 0, the next panel is order 1, etc.

In addition to the order each panel configuration can have a unique rotation applied to it. This allows you to compensate for panels which are rotated relative to each other as they snake around to form a display. The rotation is a value in degrees and can only be 0, 90, 180, or 270. You'll see in a moment a tool that helps check and determine which rotation values are needed for the panels.

For example a 4 panel square display might be wired as follows:



This display would have a panels configuration value that looks like:

```
panels = (
  ( { order = 3; rotate = 180; }, { order = 2; rotate = 180; } ),
  ( { order = 0; rotate =   0; }, { order = 1; rotate =   0; } )
)
```

Notice that there are two rows and two columns of panel settings (the panel settings are the values inside the curly braces { }, and the parenthesis indicate the panels in a row).

The order for each panel is set exactly as shown in the diagram, and a rotation of 180 degrees is applied to the top row of panels. These top panels need to be rotated because they are in a different orientation compared to the bottom panels. If no rotation was applied then the top row would be flipped and the display wouldn't look right.

If you aren't sure how to set the rotation don't worry you will see how to use a tool in the next section to check the rotations. Just give each panel a rotation of zero and move on to adjust it later.

Also it's not shown here but each panel entry can set a 'parallel = x;' value where x defines which parallel channel (0, 1, or 2) the panel is associated with. Remember each parallel channel will have a unique ordering of panels so be sure to set the panel order based on where it is along that parallel chain. Parallel chains aren't supported on the LED matrix HAT, only if you're manually wiring matrices to GPIO.

The final setting is an optional one to enable the display of a cropped portion of the Pi's video output. This allows you to see a pixel-perfect part of the Pi's video output instead of shrinking down the entire image.

```
// By default the rpi-fb-matrix tool will resize and scale down the screen
// to fit the resolution of the display panels.  However you can instead grab
// a specific pixel-perfect copy of a region of the screen by setting the x, y
// screen pixel coordinates below.  A rectangle of the exact size of the display
// (i.e. display_width x display_height pixels) will be copied from the screen
// starting at the provided x, y coordinates.  Comment this out to disable
// this crop behavior and instead resize the screen down to the matrix display.
//crop_origin = (0, 0)
```

To enable the crop behavior just remove the // comment from the line and set the X and Y coordinate that you'd like to define as the upper left corner of the crop rectangle on the screen. The width and height of the crop rectangle will be set as the display width and height.

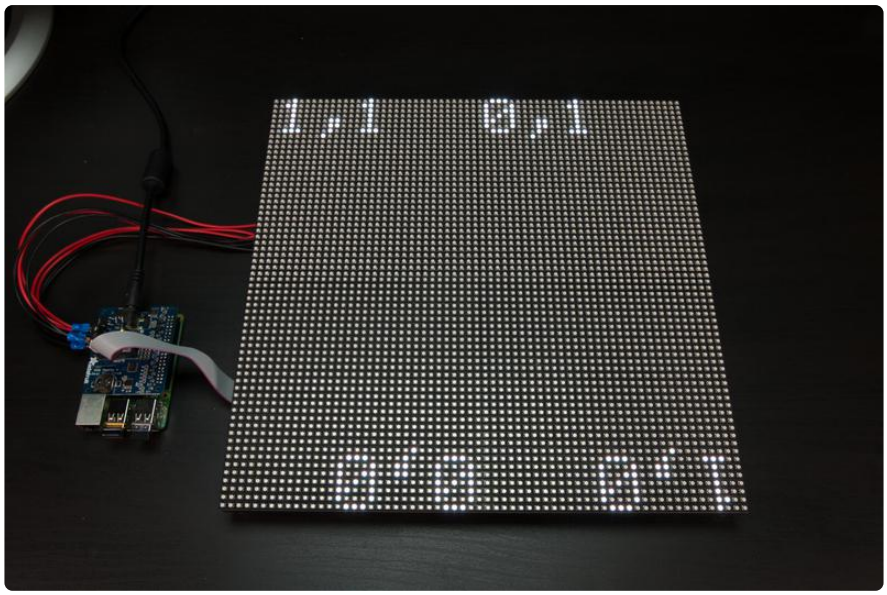
Once you're done editing the file you can save it and exit nano by pressing Ctrl-s then enter, and then Ctrl-x.

Display Test

A useful part of the software for this project is the display-test tool which will print out the row and column of the Pi video display region associated with each panel. You can run the tool by pointing it at a matrix.cfg file (being careful to run the tool as root with sudo):

```
sudo ./display-test matrix.cfg
```

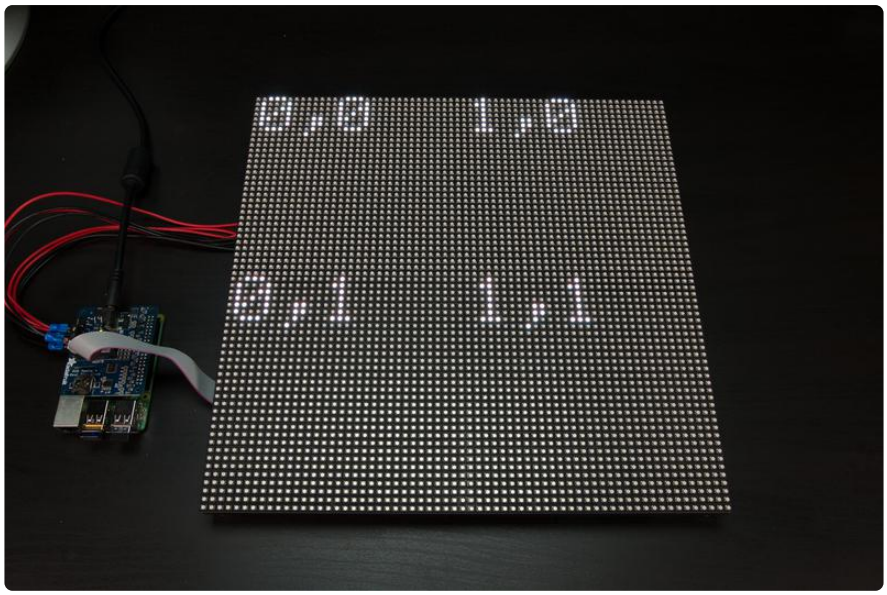
Once running the LED matrix panels should light up with text on each panel. For example you might see something like:



The text on a panel should be printed in its upper left corner. If the text is not in that corner then the rotation for that panel needs to be adjusted. You can see in the photo above the bottom row of panels do not have the right rotation. To fix this problem the matrix.cfg should be edited and a rotation of 180 degrees should be applied to those panels.

On each panel the column, row of the display region associated with that panel will be printed. For example the picture above shows the top left panel is displaying the second column and second row (the numbers are zero-based) of the display. However this is not correct, instead we should expect to see the top left panel displaying the first column and first row, or 0,0 value. Again the matrix.cfg should be edited and the order of the panels adjusted so they follow the chain from the Raspberry Pi.

With both those problems fixed the tool now displays what you want to see:



All of the panels have the correct orientation with text in the upper left. In addition the values on each panel are displaying the right order of columns and rows. Make adjustments to your matrix.cfg panel settings until the display-test tool gives output just like the above.

Running The Program

Once you've configured and tested your display you're ready to run the program that will copy the Pi's video output to the panels.

First make sure a TV or monitor is connected to the HDMI port on the Pi and that you see video output like a terminal or X windows (don't worry you'll learn how to make the panel work without a TV/monitor being connected in the next section).

Now run the rpi-fb-matrix program and point it at your configuration (again running as root with sudo):

```
sudo ./rpi-fb-matrix matrix.cfg
```

Woo hoo, you should see the video from the Pi shrunken down and displayed on the matrices! Try running a game or other graphical application to see how it looks on the display.

You might see some flickering from the LEDs. This is normal and a side-effect of the Raspberry Pi's Linux operating system trying to drive matrices that need constant refreshing. Big commercial LED signs use fast FPGA or other dedicated drivers, but the Raspberry Pi is much simpler and can only give a 'best effort' at driving the

matrices. If you're watching a movie, playing a game, or displaying things that have some motion the flickering is generally not noticeable. The amount of load on the Raspberry Pi CPU will also affect the LED flickering, so try to keep the load low (and the multiple cores of the Pi 2 will greatly help!).

If you notice portions of the screen that seem to be jumbled or rotated the wrong way you might need to adjust your configuration. Go back to the display test section above and carefully check the configuration using that tool.

When you're finished press Ctrl-C to end the program. That's really all there is to the Raspberry Pi LED matrix display! Simply run the program in the background (perhaps even configuring it to run automatically on boot ()) and it will mirror whatever is displayed by other programs running on the Pi.

Force HDMI Output & Resolution

If you don't plan to have a monitor or TV connected to the Pi you can adjust some configuration to turn on the Pi's HDMI output even with no display connected. This is necessary to make sure a primary display is available for the rpi-fb-matrix program to copy from.

The configuration you want to change is described in the 5" HDMI monitor display guide (). Specifically edit the /boot/config.txt file on the Pi and adjust it so the following sections are uncommented and added:

```
# uncomment if hdmi display is not detected and composite is being output
hdmi_force_hotplug=1

# uncomment to force a specific HDMI mode (here we are forcing 800x480!)
hdmi_group=2
hdmi_mode=1
hdmi_mode=87
hdmi_cvt 1024 768 60 6 0 0 0
```

The hdmi_cvt line in particular is important because it defines the resolution that will be forced on the HDMI display, like 1024x768 pixels as shown above. You can even change this to a square value like 1024x1024 pixels if you're using a square display.

After modifying the /boot/config.txt reboot the Pi for the change to take effect.

I recommend keeping the display at a large value like 1024 pixels instead of trying to set it to the exact dimensions of your LED matrix display. This will help prevent issues with programs that might not handle such a tiny resolution, but will still scale down

well to the matrices. Experiment with differerent sizes to see what works best for your configuration.

That's all there is to the Raspberry Pi LED matrix display project!