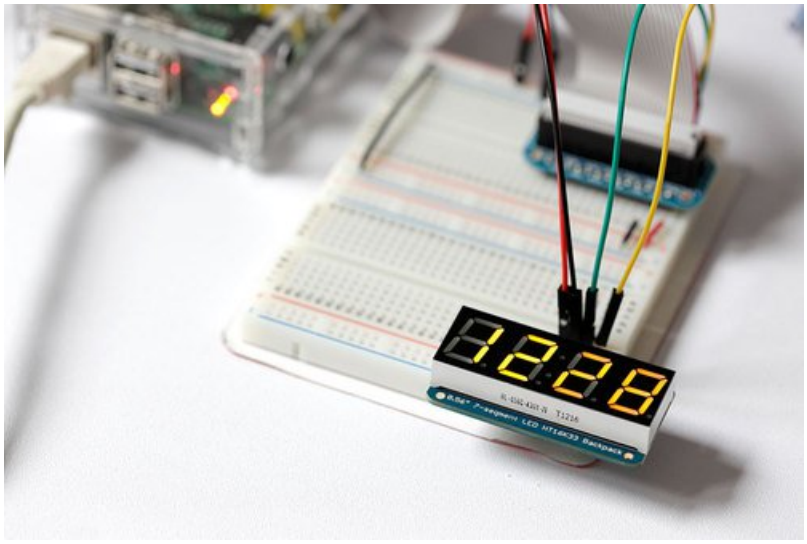
adafruit learning system

Matrix and 7-Segment LED Backpack with the Raspberry Pi

Created by Kevin Townsend



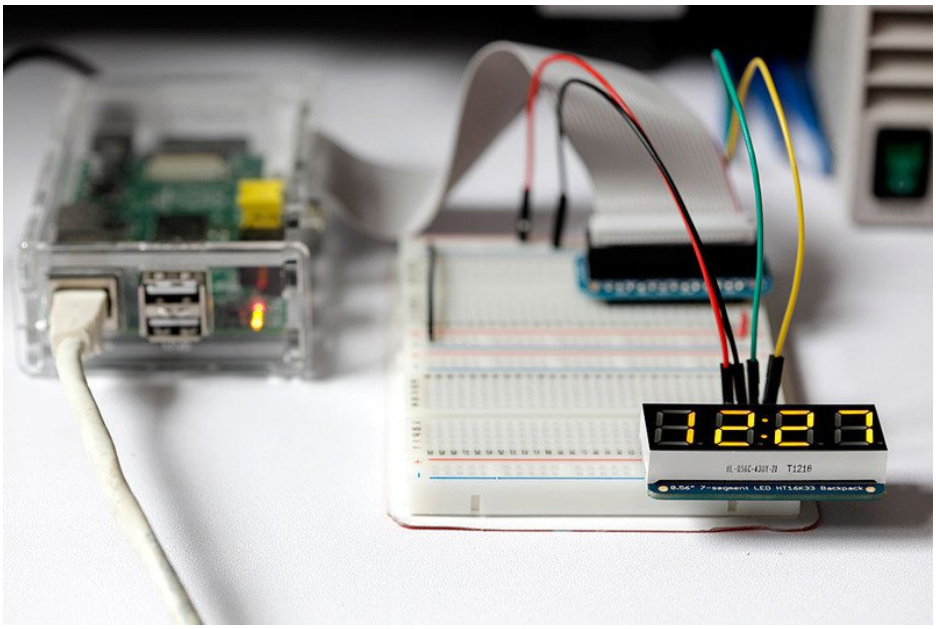
Last updated on 2019-09-07 06:43:43 PM UTC

Overview

This guide works all Raspberry Pi Models that have headers. That is every model except for the compute modules which do not have headers.

LED backpack displays are a great way to add a simple, bright LED display to your project. These displays get their name because of the controller chip attached to the back of them like a 'backpack'. This HT16K33 controller can drive up to 128 multiplexed LEDs in matrix, bar graph, 7-segment numeric, and even 14-segment alpha-numeric configurations. It handles the LEDs with a constant-current driver so the light is bright and consistent even if the power supply varies.

While we already have a great tutorial (https://adafru.it/aOV) and Arduino library (https://adafru.it/aLl) for our handy and easy to use LED backpacks, if you're wondering how you can make use of backpack-enabled LED displays on the Pi, this guide will help you get started!



What You'll Need

1. A Raspberry Pi (https://adafru.it/Bi1)
2. A Pi Cobbler Plus (https://adafru.it/Bsr)
3. One of our many 8x8 (https://adafru.it/aLG) or 4-Digit 7-Segment (https://adafru.it/aLH) backpack-enabled displays

Related Information

If you're looking for help on how to assemble one of our backpack-enabled LED displays, have a look at our earlier guide, Adafruit LED Backpacks (https://adafru.it/aOV). It contains a lot of complimentary information on these displays, including step by step soldering and assembly instructions.

Hooking Everything Up

This guide works with all versions of Raspberry Pi except the compute modules which do not have a header. Older Raspberry Pi revisions 1 and 2 will require a 26-pin Cobbler and newer versions make use of the Cobbler Plus 40-pin GPIO connector. We provide wiring diagrams for both 40-pin and 26-pin below.

The LED backpack displays are wonderfully easy to connect to the Pi. Each module type is connected the same way. The following types are covered for wiring and example code in this guide.

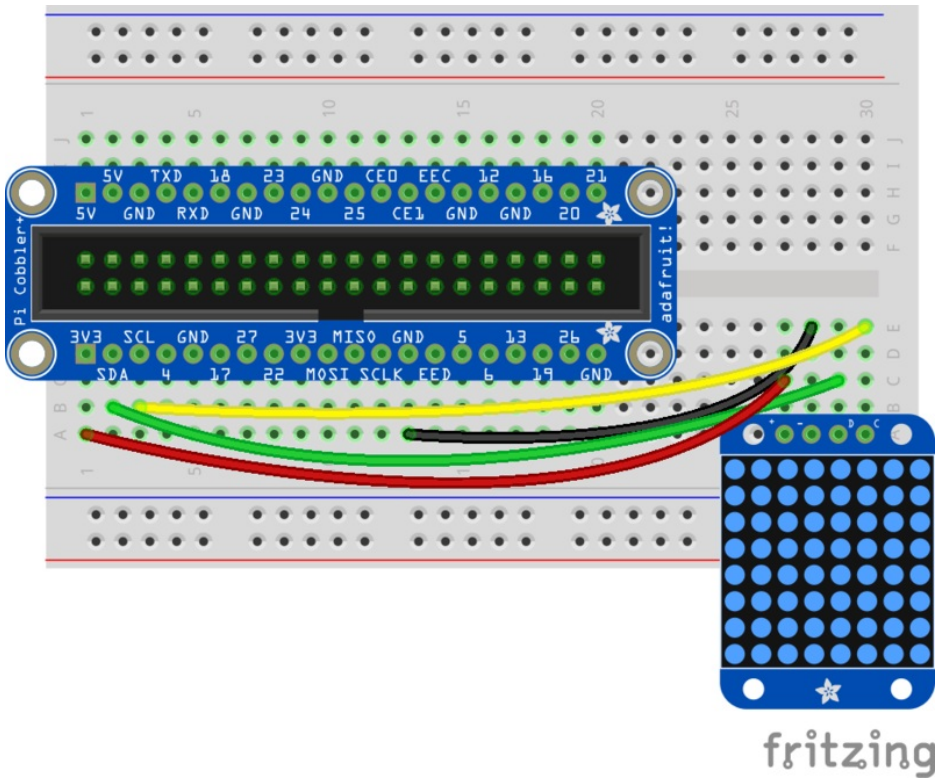
- 8x8 matrix
- 4-character 7-segment
- 4-character 14 segment (alphanumeric)
- bicolor matrix
- bicolor24 bar

Two digital pins for I2C (SDA and SCL) and two power pins (VCC and GND), as follows.

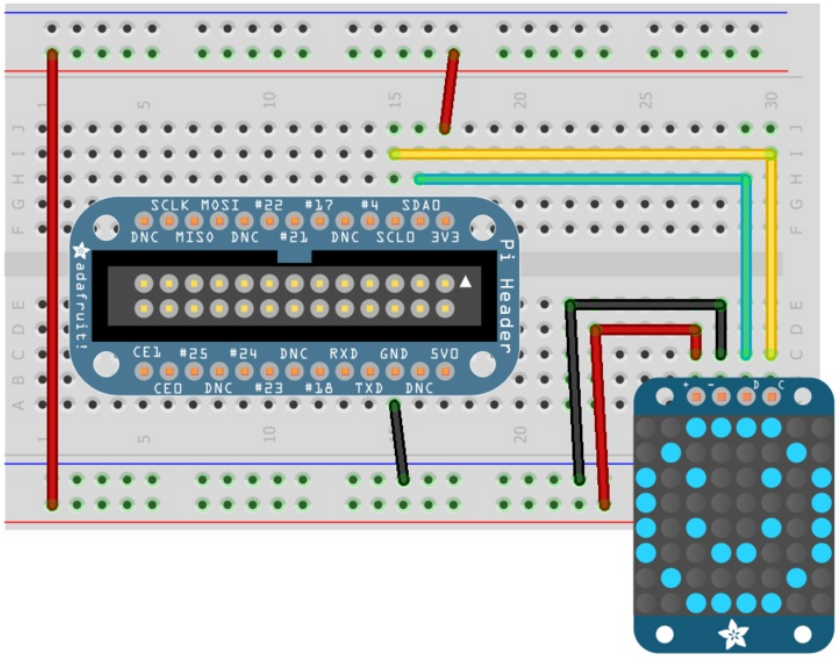
If you're using a red, yellow or green backpack, you can power the backpack from 3.3V which will keep the I2C levels at 3.3V. If you have a blue or white backpack, the LEDs will be dim if powered from 3.3V.

If you want them a little brighter, connect the VCC pin of the backpack to 5V. There are 10K pullups on the backpack to 5V. **As long as the backpack is the only i2c device on the i2c bus with pullups to 5V this is perfectly safe for the Pi. If you have 2 or more 5V i2c devices, the 5V pullups may 'overpower' the Pi's strong 3.3v pullups, in this case you'll want to use a proper level shifter:** https://www.adafruit.com/products/757 (http://adafru.it/757)

40-Pin (A, B, B+ and Zero) Cobbler Plus Schematic



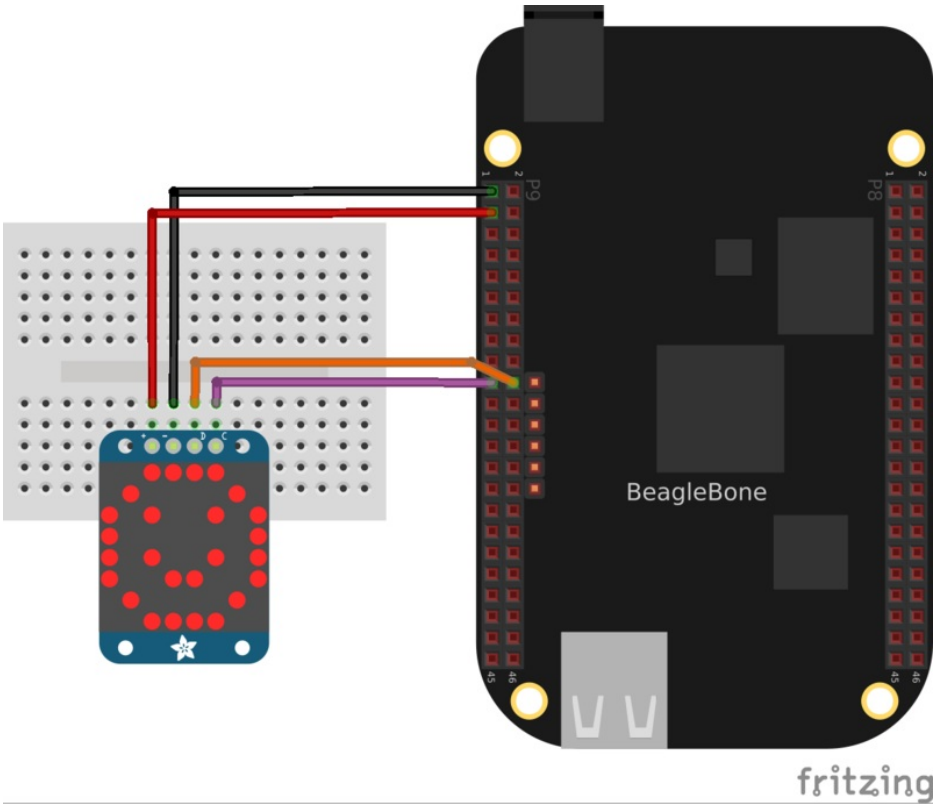
26-Pin (Raspberry Pi Rev 1 and Rev 2) Cobbler Schematic



BeagleBone Black

Wire up the LED backpack display to the BeagleBone Black as follows (note the image below shows a matrix display,

however the wiring is the same for any backpack):



- Connect **display + (power)** to **BeagleBone Black 3.3V or 5V power (red wire). 5V is brighter but if you have other devices on the I2C bus its better to go with 3.3V**
- Connect **display - (ground)** to **BeagleBone Black ground** (black wire).
- Connect **display D (data/SDA)** to **BeagleBone Black I2C2_SDA pin P9_20** (orange wire).
- Connect **display C (clock/SCL)** to **BeagleBone Black I2C2_SCL pin P9_19** (purple wire).
- If there's a **Vi2c** or **IO** pin, connect that to 3.3V as well

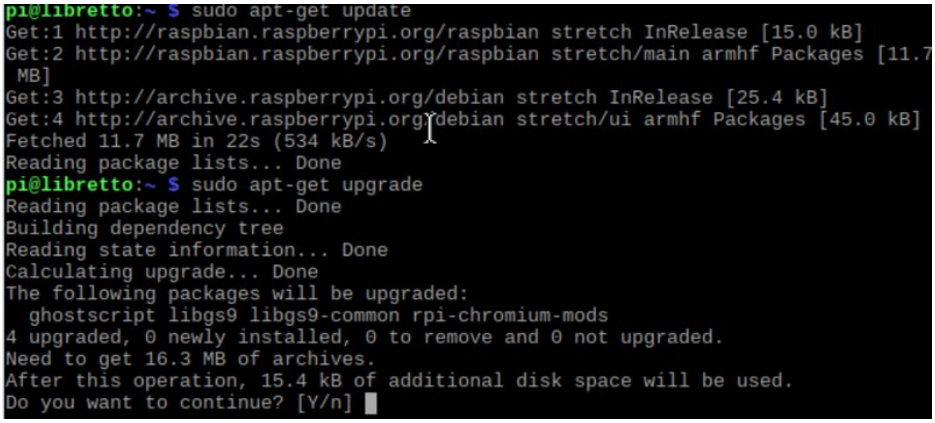
Note that the BeagleBone Black has two I2C interfaces and this wiring will use the **/dev/i2c-1** interface. Make sure there aren't any device tree overlays loaded (https://adafru.it/dp6) which use these I2C pins for other purposes. The default BeagleBone Black device tree configuration with no overlays loaded will expose the necessary I2C interface for the wiring above.

Pi Prep

Update Your Pi to the Latest Raspbian

Your Pi will need to be running the latest version of Raspbian. This tutorial was written using Raspbian Buster (July 2019). Checkout our guide for Preparing an SD Card for your Raspberry Pi (https://adafru.it/dDL) if you have not done so already. After the installation is complete be sure and run the following commands to make confirm your installation packages are up to date.

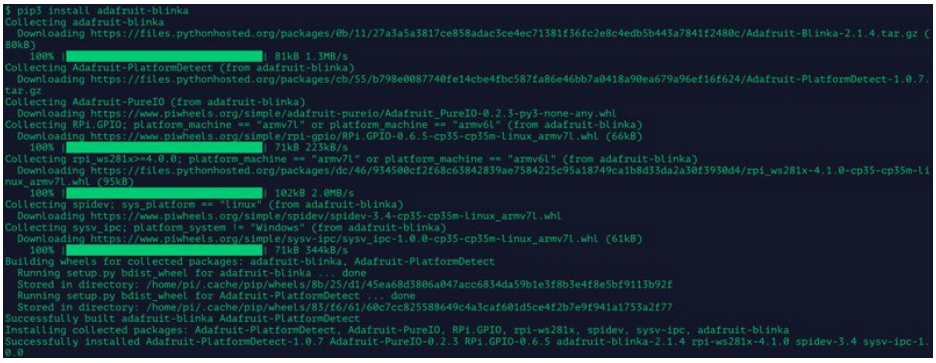
```
sudo apt-get update -y
sudo apt-get upgrade -y
```



Install adafruit-blinka

Run the following command to install the adafruit_blinka CircuitPython Libraries.

```
pip3 install adafruit-blinka
```

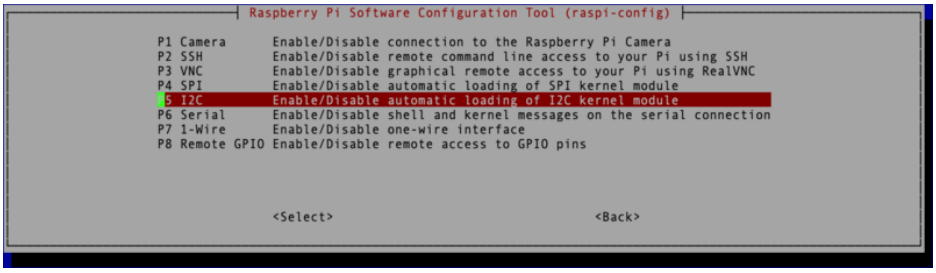


Enable the I2C Interface

The Holtek HT16K33 (https://adafru.it/aMy) chip used in all of our backpacks communicates using the common I2C bus. The I2C bus allows multiple devices to be connected to your Raspberry Pi, each with a unique address, that can often be set by changing jumper settings on the module. Linux and the Pi both have native support for I2C, but you'll need to run through a couple quick steps from the console before you can use it in Python.

To see a detailed screen by screen tutorial about how to setup I2C with Raspbian, take a minor diversion to this Adafruit Tutorial: Raspberry Pi Lesson 4. GPIO Setup (https://adafru.it/aTI). We will provide a brief summary below for those somewhat familiar with the **raspi-config** utility.

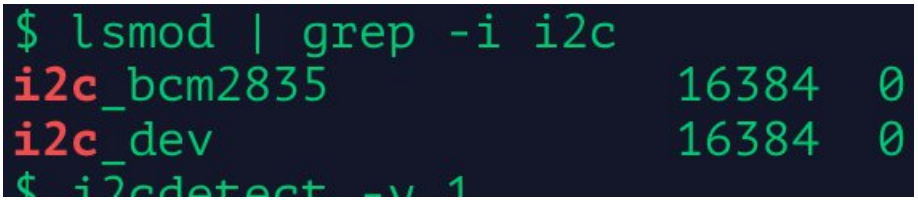
```
sudo raspi-config
<Interfacing Options>
<I2C> Enable/Disable
<Enable>
<Back>
<Finish>
```



Test the I2C Bus

Now that our Pi has rebooted we can confirm that the correct I2C modules have loaded with the following command.

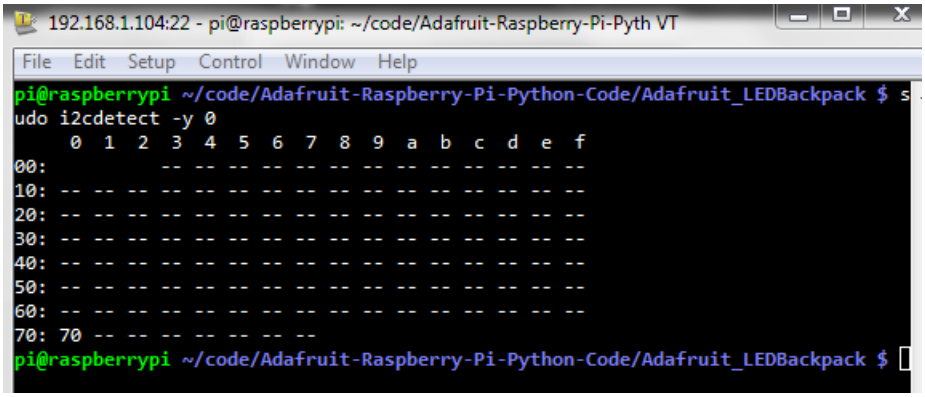
```
lsmod | grep -i i2c
```




```
sudo i2cdetect -y 1
```

This will search I2C for all address, and if a Holtek HT16K33 breakout is properly connected and it's set to it's default address it should show up as follows (the exact address will vary depending on whether or not you have any of the address solder jumpers set).

If you happen to have one of the original first batch of Raspberry Pis, you will need to change the 1 to a 0 in the command above.



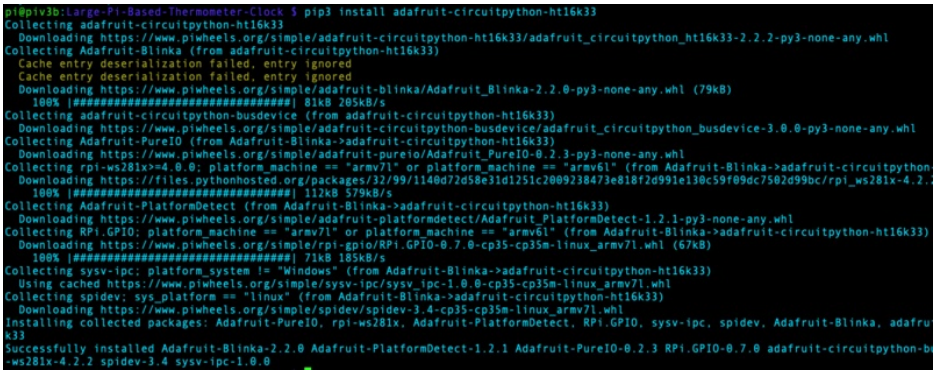
Recent Raspbian releases includes the i2cdetect command, but if it is not found on your system please use the following syntax to install the necessary packages.

```
sudo apt-get install -y python-smbus i2c-tools
```

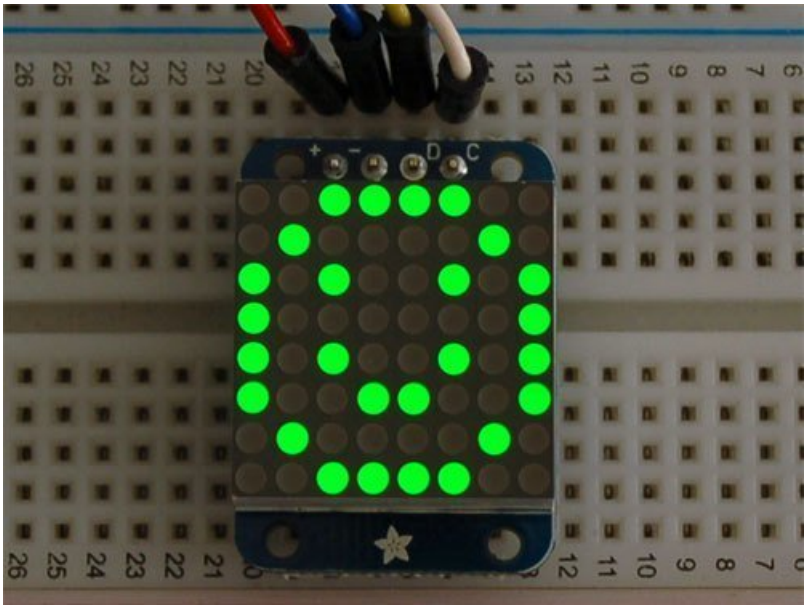
Install the HT16K33 Library

The software for this project uses the Adafruit code for driving the 7-segment and matrix displays.

```
pip3 install adafruit-circuitpython-ht16k33
```



Matrix 8x8 Pixel



CircuitPython Code

The following example will illuminate the matrix rows and columns one pixel at a time.

```
# Import all board pins.
import time
import board
import busio
from adafruit_ht16k33 import matrix

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# creates a 8x8 matrix:
matrix = matrix.Matrix8x8(i2c)

# edges of an 8x8 matrix
col_max = 8
row_max = 8

# Clear the matrix.
matrix.fill(0)
col = 0
row = 0

while True:

    # illuminate a column one LED at a time
    while col < col_max:
        matrix[row, col] = 2
        col += 1
        time.sleep(.2)

    # next row when previous column is full
    if row < row_max:
        row += 1
        col = 0

    # clear matrix, start over
    else:
        row = col = 0
        matrix.fill(0)
```

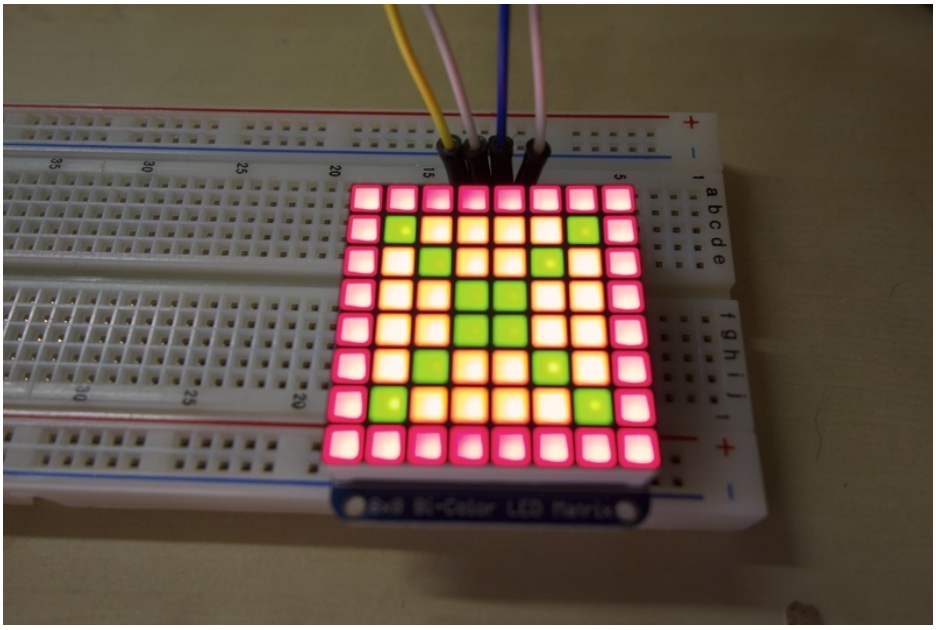
Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/Matrix_7-Segment_LED_Back
python3 ./matrix8x8_test.py
```

Which should result in something like the following:

Matrix Bicolor



The bicolor matrix display is demonstrated in the **matrix_bicolor_test.py** example below. This code demonstrates using the **Matrix8x8x2** class in a very similar way to the **Matrix8x8** class. The key difference is that are assigning color value and are not limited to a 1 or 0 value like with the **Matrix8x8** class.

CircuitPython Code

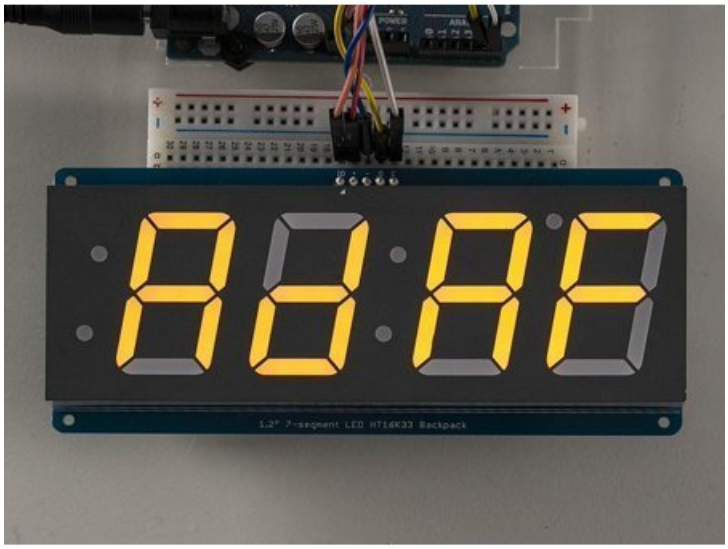
```
# Import all pins
import time
import board
import busio
from adafruit_ht16k33 import matrix

# Create the I2C interface
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED bargraph class.
bicolor = matrix.Matrix8x8x2(i2c)

# color mapping shortcut
OFF = 0
GREEN = 1
RED = 2
YELLOW = 3

# Set individual segments of the bicolor matrix
# Illuminate the first three pixels
bicolor[0,0] = GREEN
bicolor[0,1] = RED
bicolor[0,2] = YELLOW

time.sleep(2)

# Edges of an 8x8 matrix
```

```
col_max = 8
row_max = 8

# Turn all pixels off
bicolor.fill(OFF)
col = 0
row = 0

# Illuminate each pixel with each color
while row < row_max:

    # Turn them on in a loop
    while col < col_max:
        bicolor[row, col] = RED
        time.sleep(.25)
        bicolor[row, col] = GREEN
        time.sleep(.25)
        bicolor[row, col] = YELLOW
        time.sleep(.25)
        col += 1

    # next row when previous column is full
    if row < row_max:
        row += 1
        col = 0

    # clear matrix, start over
    else:
        row = col = 0
        bicolor.fill(OFF)

time.sleep(1)

# Fill the entrire display, with each color
bicolor.fill(GREEN)
time.sleep(1)
bicolor.fill(RED)
time.sleep(1)
bicolor.fill(YELLOW)
time.sleep(1)
bicolor.fill(OFF)
```

Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Matrix_7-Segment_L
python3 matrix_bicolor_test.py
```



CircuitPython Code

The following code illustrates how to display integers, characters, floating point, hex values and toggle the colon on a 7-segment display.

```
import time
import board
import busio
from adafruit_ht16k33 import segments

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED segment class.
# This creates a 7 segment 4 character display:
display = segments.Seg7x4(i2c)

# Clear the display.
display.fill(0)

# Can just print a number
display.print(42)
time.sleep(1)

# Set the first character to '1':
display[0] = '1'
# Set the second character to '2':
display[1] = '2'
# Set the third character to 'A':
display[2] = 'A'
# Set the forth character to 'B':
display[3] = 'B'
time.sleep(1)

numbers = [0.0, 1.0, -1.0, 0.55, -0.55, 10.23, -10.2, 100.5, -100.5]

# print negative and positive floating point numbers
for i in numbers:
    display.print(i)
    time.sleep(0.5)

# print hex values, enable colon
for i in range(0xFF):
    display.fill(0)
    display.print(':')
    display.print(hex(i))
    time.sleep(0.25)
```

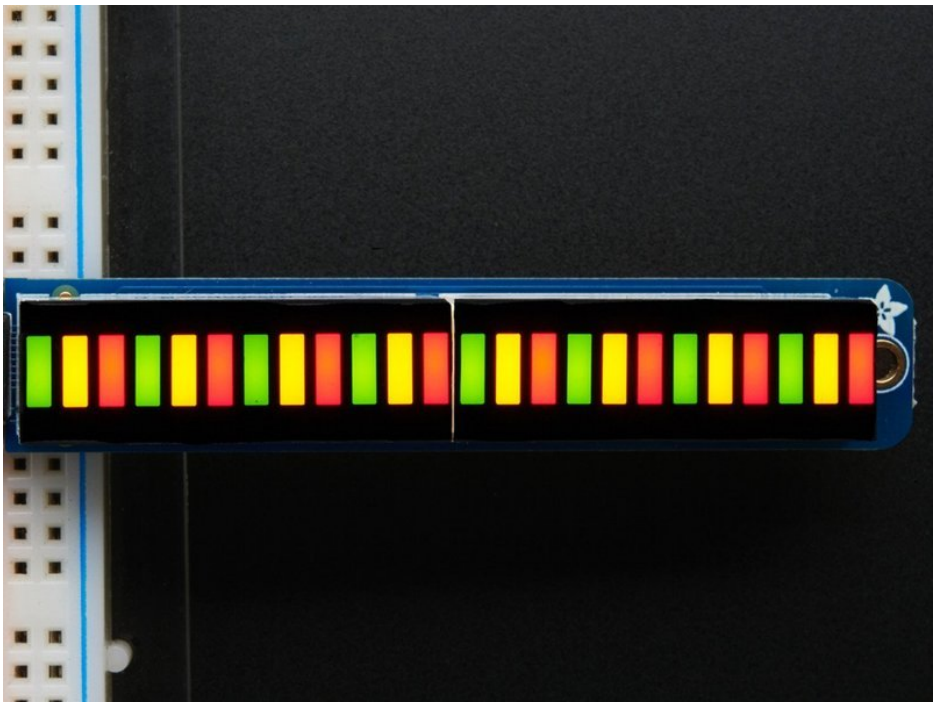
Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Matrix_7-Segment_L
python3 ./sevensegment_test.py
```

This will briefly scroll through printing integers, floating point values, some text characters and hex values on a seven segment display.

Bicolor Bar Graph 24



CircuitPython Code

The bicolor bar graph display is demonstrated in the **bicolor24_test.py** script. This code will light up 3 bars in different colors and then loop through full color all on and all off modes.

```
# Basic example of using the Bi-color 24 segment bargraph display.
# This example and library is meant to work with Adafruit CircuitPython API.
# Author: Carter Nelson
# License: Public Domain

import time
import board
import busio

# Import the Bicolor24 driver from the HT16K33 module
from adafruit_ht16k33.bargraph import Bicolor24

# Create the I2C interface
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED bargraph class.
bc24 = Bicolor24(i2c)

# Set individual segments of bargraph
bc24[0] = bc24.LED_RED
bc24[1] = bc24.LED_GREEN
bc24[2] = bc24.LED_YELLOW

time.sleep(2)

# Turn them all off
bc24.fill(bc24.LED_OFF)

# Turn them on in a loop
for i in range(24):
    bc24[i] = bc24.LED_RED
    time.sleep(0.1)
    bc24[i] = bc24.LED_OFF

time.sleep(1)

# Fill the entrire bargraph
bc24.fill(bc24.LED_GREEN)
```

Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://github.com/adafruit/Adafruit_Learning_System_Guides/blob/master/Matrix_7-Segment_LED_Backpac
python3 bicolor24_test.py
```

7-Segment Clock

CircuitPython Code

Display the system time on a four digit seven segment display.

```
import time
import datetime
from adafruit_ht16k33 import segments
import board
import busio

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED segment class.
# This creates a 7 segment 4 character display:
display = segments.Seg7x4(i2c)

# clear display
display.fill(0)

while True:
    # get system time
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second

    # setup HH:MM for display and print it
    clock = '%02d%02d' % (hour,minute)          # concat hour + minute, add leading zeros
    display.print(clock)

    # Toggle colon when displaying time
    if second % 2:
        display.print(':')                      # Enable colon every other second
    else:
        display.print(';')                      # Turn off colon

    time.sleep(0.5)
```

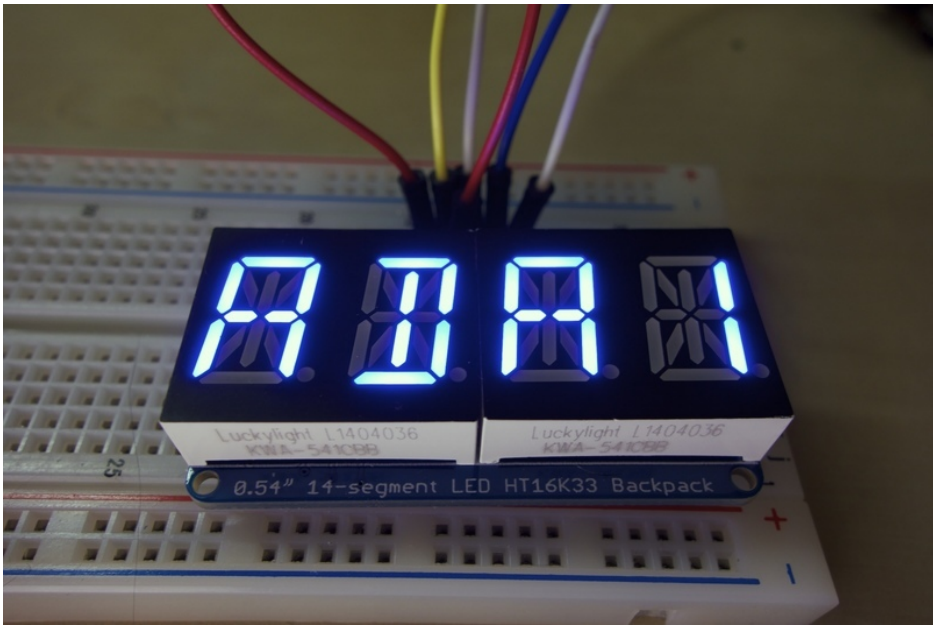
Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Matrix_7-Segment_L
python3 ./sevensegment_clock.py
```

You should see the time appear on your 7-segment display in 24-hour format with a slow blinking colon. It should look like this:

14-Segment Alphanumeric Display



CircuitPython Code

The 14 segment alphanumeric display is demonstrated in the **alphanum4_test.py** script in the examples folder. If you run this example it will scroll a text message across the display. The message if followed by a run down of various integer, decimal, hex and character values. This display is great for showing complete text messages because the 14 segment displays are very flexible.

With the 14 segment display the usage is very similar to the 7 segment display above.

```
import time
import board
import busio
from adafruit_ht16k33 import segments

# Create the I2C interface.
i2c = busio.I2C(board.SCL, board.SDA)

# Create the LED segment class.
# This creates a 14 segment 4 character display:
display = segments.Seg14x4(i2c)

# Clear the display.
display.fill(0)

# set brightness, range 1-15, 15 max brightness
display.brightness = 15

# show phrase on alphanumeric display
message = "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
count = 0

# print one character at time with short sleep
# creates smooth scrolling effect
while count < len(message):
    display.print(message[count])
    count += 1
    time.sleep(0.3)

# Can just print a number
display.print(42)
time.sleep(1)

# Set the first character to '1':
display[0] = '1'
# Set the second character to '2':
display[1] = '2'
# Set the third character to 'A':
display[2] = 'A'
# Set the forth character to 'B':
display[3] = 'B'
time.sleep(1)

numbers = [0.0, 1.0, -1.0, 0.55, -0.55, 10.23, -10.2, 100.5, -100.5]

# print negative and positive floating point numbers
for i in numbers:
    display.print(i)
    time.sleep(0.5)

# print hex values, enable colon
for i in range(0xFF):
    display.fill(0)
    display.print(':')
    display.print(hex(i))
    time.sleep(0.25)
```

Download and Run the Code

We can easily copy this code onto our Pi's home directory using the 'wget' command and then run it using the following commands.

```
cd
wget https://raw.githubusercontent.com/adafruit/Adafruit_Learning_System_Guides/master/Matrix_7-Segment_L
python3 alphanum4_test.py
```