

Raspberry Pi Weather Station using Adafruit Parts

nashdb@yahoo.com

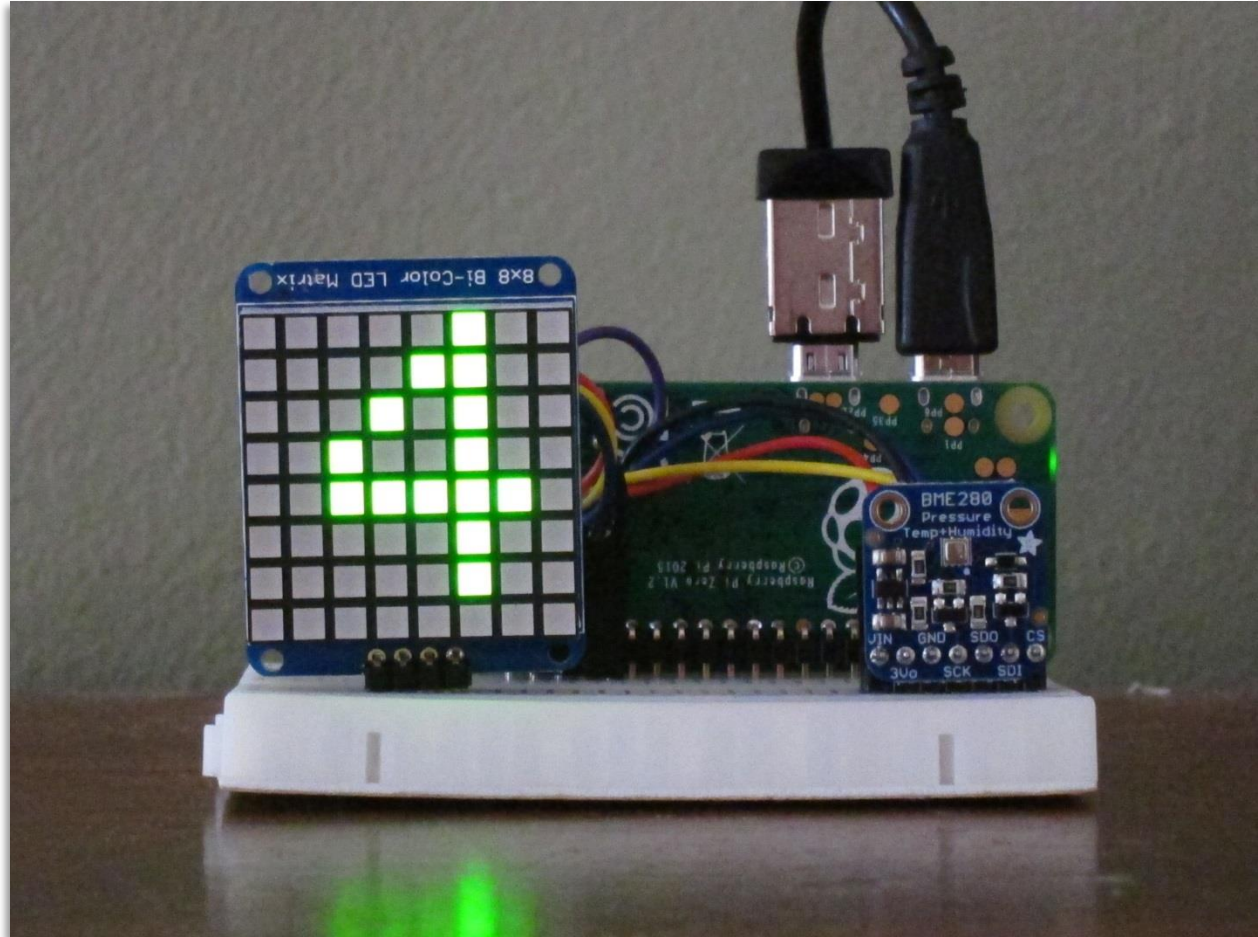


Figure 1. A video of the project in operation is available [here](#).

Abstract

A small, simple device for continuously scrolling the current time, date, temperature, humidity, and barometric trend over the last two hours is built from a [Raspberry Pi Zero](#), an [Adafruit BME280 I2C Temperature Humidity Pressure Sensor](#), and an [Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack](#) for a total cost of less than \$40. Its display is easily visible across a large, brightly lit room and its small size and utility make it an informative and attractive conversation piece for any room.

Introduction

Although any model of Raspberry Pi may be used for this project, the low cost, and small size of the recently released Raspberry Pi Zero make it ideal for use in permanent installations. The project presented here can be assembled by anyone with an intermediate-level of competency with a Raspberry Pi and some familiarity with the Python programming language. Wiring is quite simple because the display, an [Adafruit Bicolor LED Square Pixel Matrix](#), and the sensor, an [Adafruit BME280](#) both use the I^2C two-wire serial protocol. In all pictures and wiring diagrams a black wire is used for ground, a red wire is used for +3.2V, a blue wire is used for the I^2C clock, and a maize wire is used for the I^2C data line.

Communicating with the Raspberry Pi

It is assumed here that the user has already setup his or her Raspberry Pi and has successfully installed a WiFi USB dongle for communicating with their Raspberry Pi. A [USB to Micro USB Male OTG Adapter](#) is particularly recommended if using a Raspberry Pi Zero (**Figure 1**). With the Raspberry Pi successfully connected to the local WiFi network, installation of the software may be done with an RDP (Remote Desktop Protocol) program from a Window, Mac, Android, or iPhone. I am most familiar with the Windows programs *XLaunch* (see [here](#) for setup instructions) and Microsoft's *Remote Desktop Connection* (see [here](#) for setup instructions). Both programs have advantages and disadvantages. I prefer Microsoft's *Remote Desktop Connection* although it does require installation of xrdp on the Raspberry Pi:

```
sudo apt-get install xrdp
```

Parts

The project can be assembled on a solderless prototyping board (breadboard), perfboard or other base. Only two parts are used along with jumper wires; all are available from Adafruit. Although I like [Adafruit's Premium Male/Male Jumper Wire package](#), it makes consistent color use for particular types of connections difficult because of the relatively small number of any particular wire color. It is not strictly necessary to use a given wire color for a particular purpose but it does make things simpler and safer to be consistent in wire color selection. I always use red wire for positive (+3.2V) power, black wire for ground connections, and maize and blue wires for I^2C data and clock connections respectively. I also prefer to make my own jumper wires because the pins (male dupont servo pins) can be assembled into multi-pin plugs which makes connections easier and less prone to error.

Both of the parts used communicate with the Raspberry Pi using the two-wire, I^2C serial communication protocol (another two wires are necessary to power each device). I^2C makes wiring simple but does require a one-time, initial setup. Adafruit has an [excellent tutorial](#) on setting up I^2C on the Raspberry Pi although if you are using a recent release of [Raspbian Jessie](#) some of the steps toward the end tutorial are unnecessary. Do not proceed until you have successfully setup I^2C and are able to successfully execute

```
sudo i2cdetect -y 1
```

Next, install the [Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack](#). First visit [here](#) and *read the installation requirements very carefully*. It is essential to install other supporting software before the device will work so make sure you execute

```
sudo apt-get install build-essential python-dev
```

and

```
sudo apt-get install python-imaging
```

Once this is done,

```
git clone https://github.com/adafruit/Adafruit\_Python\_LED\_Backpack.git
```

and setup the software (detailed instructions [here](#)). Do not proceed until the software is successfully installed and

```
sudo python bicolor_matrix8x8_test.py
```

can be successfully executed in the Adafruit_Python_LED_Backpack/examples directory.

The [Adafruit BME280 I2C or SPI Temperature Humidity Pressure Sensor](#) can perform serial communication with the Raspberry Pi via I^2C or SPI protocol but I^2C is used here. The software library necessary for this sensor is available [here](#). *Read the installation requirements very carefully*. It is ***essential*** to run

```
Sudo python Adafruit_BME280_Example.py
```

in the Adafruit_BME280 directory; not only will this verify that the sensor is working but it will create a compiled python subroutine library, *Adafruit_BME280.pyc*, necessary to operate the sensor in python. *Adafruit_BME280.pyc* or a symbolic link¹ must be in the same directory as any python program using *import Adafruit_BME280*². Once again, do not proceed until the software is installed and *Adafruit_BME280_Example.py* has been successfully executed.

Wiring

The fact that each I^2C device has (or should have) a unique address permits several of them to be strung together on the same two wires. Note, however, if you are using a Raspberry Pi Zero that pin headers are only installed in the inner row (closest to the CPU and furthest from the edge) of pin holes otherwise each pin in one row will be shorted to the pin in the other row when plugged into the prototyping board. The jumper GPIO4 (purple wire) is connected to ground to display temperature in degrees Fahrenheit and to +3.2V to display temperature in degrees centigrade³.

¹ In /home/pi/git/Adafruit_Python_BME280/Adafruit_BME280.pyc Adafruit_BME280.pyc

² It is not necessary to do this for the [Adafruit Bicolor LED Square Pixel Matrix with I2C Backpack](#) because setup.py puts the subroutine into the python search path.

³ Degrees centigrade is *always* used in data stored in the output file whatever units are specified for the display.

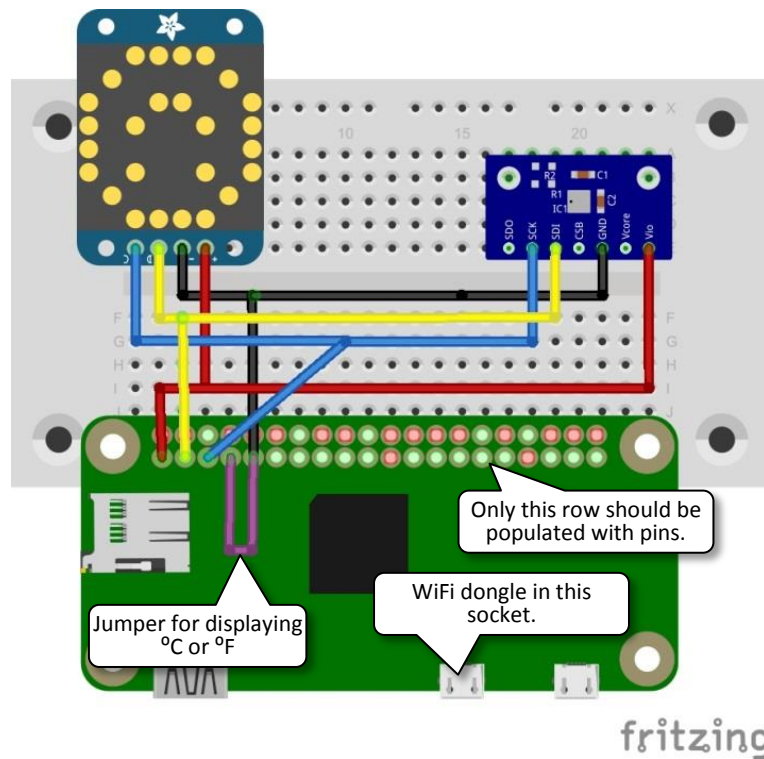


Figure 2. Simple wiring for desktop weather station.

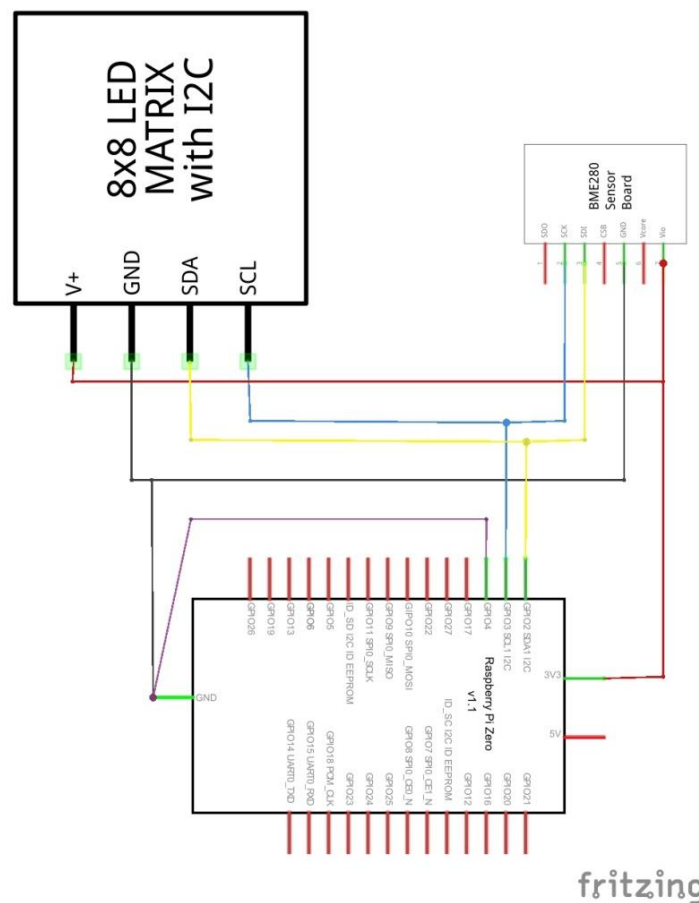


Figure 3. Schematic for **Figure 2** breadboard.

Display and Output

When the program is started, a time-date stamped output file is created, *weather_output_yymmddHHMMSS.csv* (line 178⁴ in the program listing, be sure to replace */[path to your directory]/* with the appropriate path for your Raspberry Pi installation). The date-time, temperature (always in °C), relative humidity (%), and barometric pressure (Pa) are saved as comma-separated values that can be read directly into spreadsheet programs such as *LibreOffice Calc* and *Microsoft Excel* (**Figure 1**). Output values are averaged and output at 900 second intervals (may be changed by entering a different value for *record_time*, line 167 in the program listing).

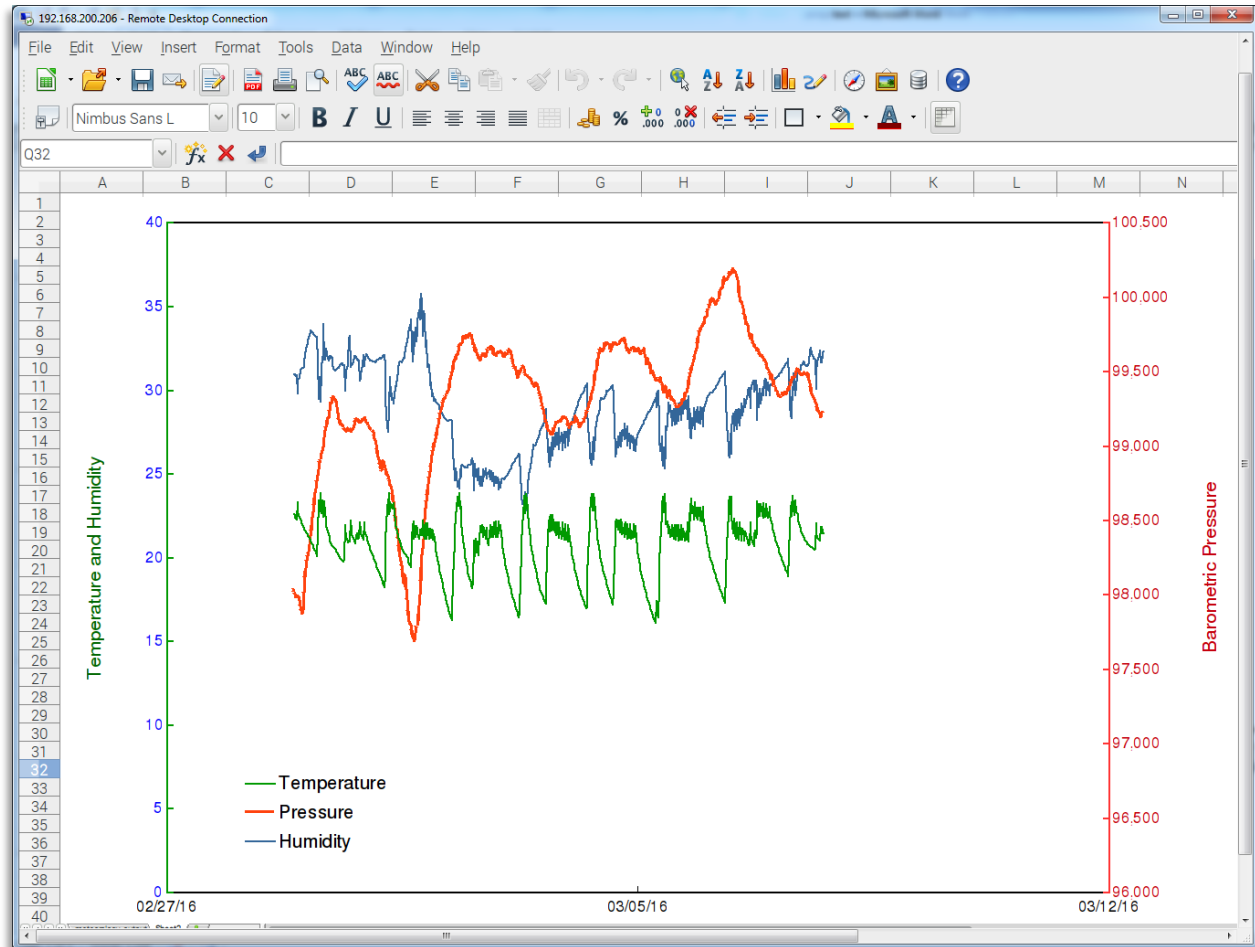


Figure 4. Output file displayed in *LibreOffice Calc* on Raspberry Pi Zero.

Values of time, date, temperature, and relative humidity are continuously scrolled across the 8x8 LED matrix at a speed and orientation specified by values given to *LED_orientation* and *scroll_speed* (program lines 165 and 166 respectively). Temperature is displayed in degrees Fahrenheit by connecting a jumper between GPIO4 and ground or in degrees centigrade by connecting the jumper between GPIO4

⁴ Note that path must be changed for your specific installation.

and +3.2V (if no jumper is connected to GPIO4, display defaults to degrees Fahrenheit). If the relative humidity is “outside” the comfort zone (*i.e.*, <40% or >60%), its value is displayed in red, otherwise it is displayed in green.

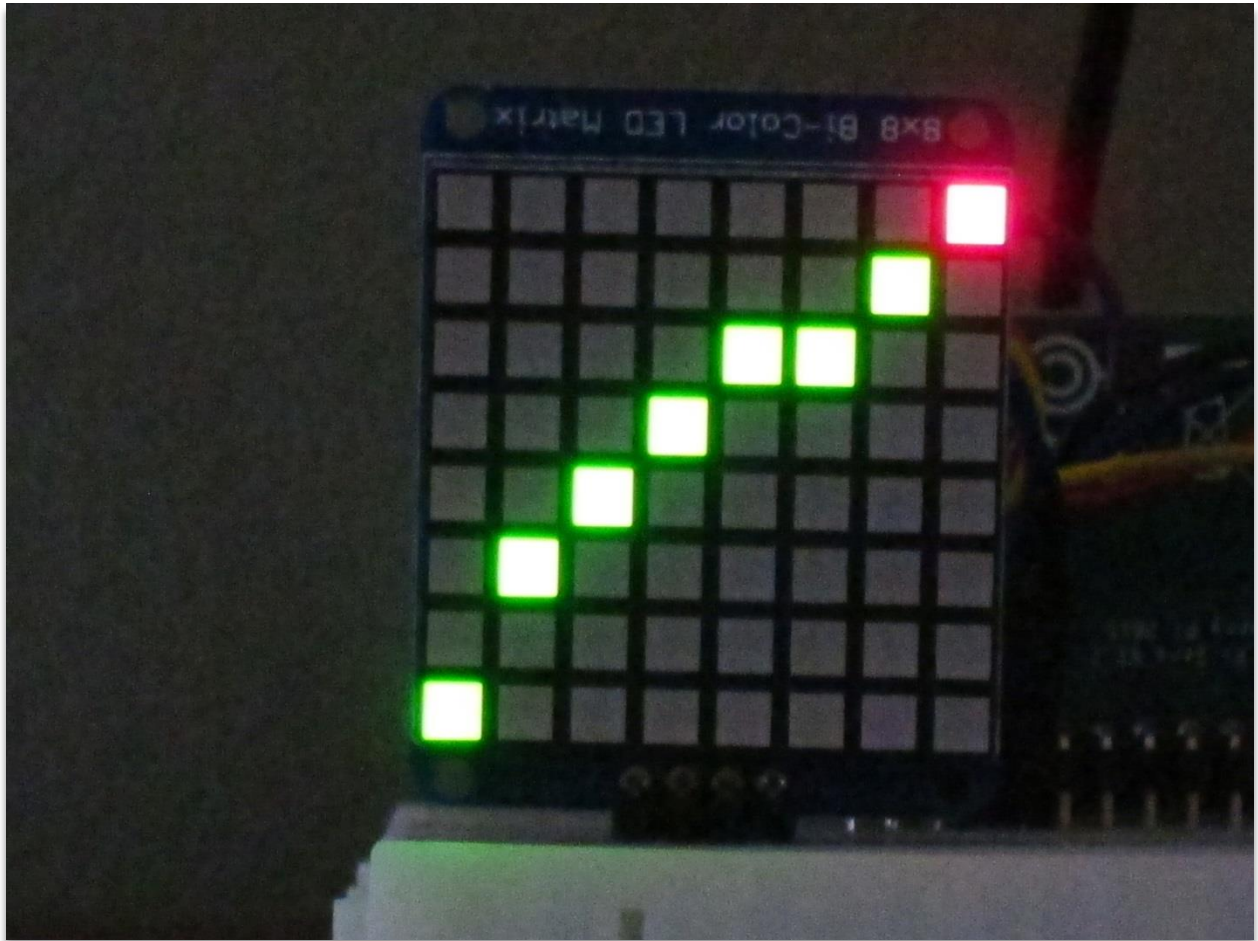


Figure 5. Pressure trend over the preceding eight times *record_time* (two hours). Because the pressure has changed more than 150 Pa over the period, the final point is displayed in red.

The trend in barometric pressure (**Figure 5**) of the preceding period of eight times *record_time* (two hours in program listing) is graphically (one point per *record_time*). The final (*i.e.*, most recent) point on the trend line is color coded depending on the change in pressure over the previous eight times *record_time* seconds: red if the change is more than 150 Pa, orange if between 150 and 100 Pa, and green if less than 100 Pa (lines 39-41 in the program listing). If less than eight times *record_time* seconds have elapsed, there is insufficient data to generate the trend line so a simple spiral is displayed instead (**Figure 6**).

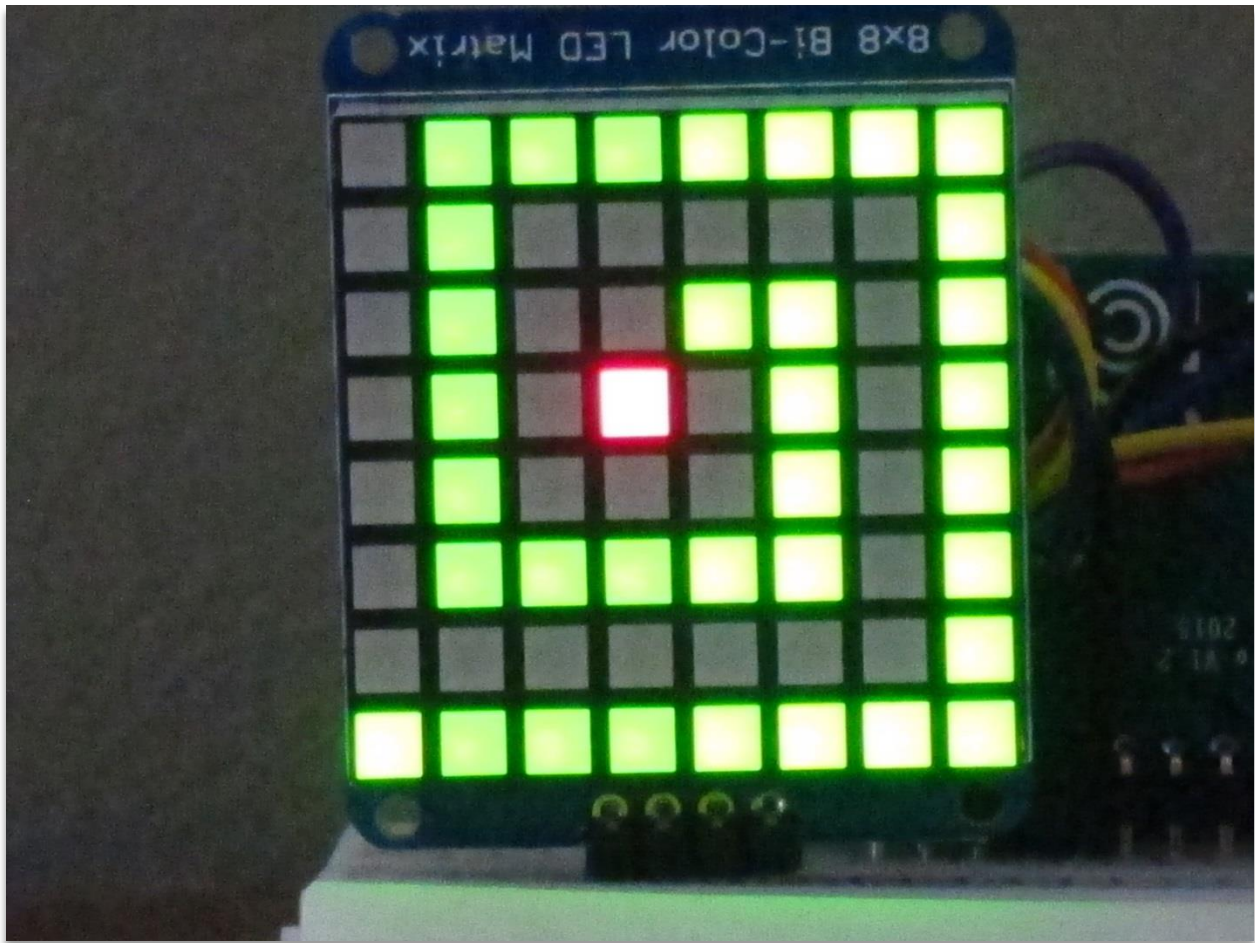


Figure 6. A simple spiral is displayed instead of the pressure trend (**Figure 5**) until eight times record_time seconds have elapsed.

Python Program

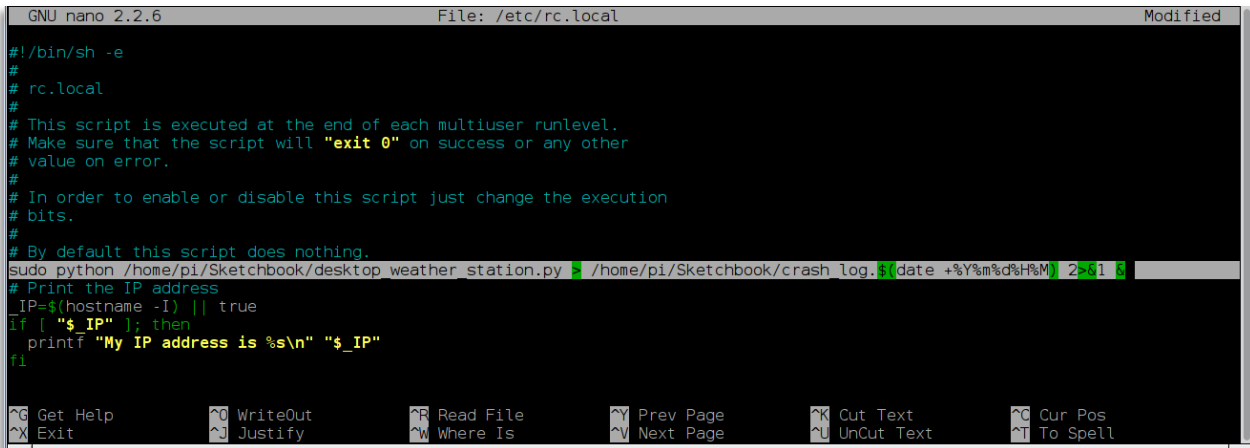
The commented program listing that follows is written for maximum clarity. It not written in the *pythonic way*, particularly regarding short line lengths (in order to have comments at the end of lines). The *scroll_display* subroutine (lines 83-143) is, IMHO, particularly useful in that it will display any string of printable ASCII characters passed to it. Scroll speed, text color, and orientation on the “screen” may all be specified in the passed parameters.

In order to have the project start when the Raspberry Pi is powered up, add the following command to the file *rc.local* in the */etc* directory (all on one line):

```
sudo python /[path to program]/desktop_weather_station.py > /[path to output directory]/crash_log.$(date +%Y%m%d%H%M) 2>&1 &.
```

This *rc.local* can only be changed as super user so use

```
sudo nano /etc/rc.local.
```



```

GNU nano 2.2.6      File: /etc/rc.local      Modified
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
sudo python /home/pi/Sketchbook/desktop weather station.py > /home/pi/Sketchbook/crash log.$(date +%Y%m%d%H%M) 2>&1
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

```

Figure 7. The highlighted line must be added to /etc/rc.local in order for the program to launch automatically on power up. Note that the paths to the python code and error output file are for my particular setup.

Aesthetics

I've had various earlier versions of this project running in my living room for several years now. Although I think it's a conversation piece as well as being a remarkably good predictor of the weather, its appearance could certainly be improved by putting it in an attractive enclosure. If you do so, however, remember if you are using a Raspberry Pi Zero within the enclosure, it will generate a non-negligible amount of heat which make the room temperature reading several degrees warmer than it actually is. This can be avoided by mounting the BME280 outside (*e.g.*, behind) the enclosure.

desktop_weather_station.py Program Listing

```

1  from Adafruit_BME280 import *
2  from Adafruit_LED_Backpack import BicolorMatrix8x8
3  import datetime, math, time, RPi.GPIO as io, smbus
4
5  grid = BicolorMatrix8x8.BicolorMatrix8x8()
6  sensor = BME280(mode=BME280_OSAMPLE_8)
7  io.setmode(io.BCM)
8
9  metric_switch=4
10 io.setup(metric_switch, io.IN, pull_up_down = io.PUD_DOWN)
11
12
13 ## Plots a spiral on screen when less than eight periods of record are available.
14
15 def display_spiral():
16     spiral=[0,1,2,3,4,5,6,7,15,23,31,39,47,55,63,62,61,60,59,58,57,49,41,33,25,17,18,19,20,21,29,37,45,44,43,35]
17     grid.clear()
18     for n in range(len(spiral)-1):
19         time.sleep(0.1)
20         grid.set_pixel(int(spiral[n]/8), spiral[n]%8, 1)
21         grid.write_display()
22     time.sleep(0.1)
23     grid.set_pixel(int(spiral[len(spiral)-1]/8), spiral[len(spiral)-1]%8, 2)
24     grid.write_display()
25     time.sleep(.5)
26     for n in range(len(spiral)-1):
27         time.sleep(0.1)
28         grid.set_pixel(int(spiral[len(spiral)-2-n]/8), spiral[len(spiral)-2-n]%8, 0)
29         grid.write_display()
30     return
31
32
33 ## Plot relative change in barometric pressure over the last eight recording periods.
34
35 def display_trend(array,orientation,plot_speed):
36     minimum_value=min(array)
37     maximum_value=max(array)
38     pressure_change=maximum_value-minimum_value
39     if pressure_change>150: warning_level=2
40     elif pressure_change>100: warning_level=3
41     else: warning_level=1
42     if maximum_value-minimum_value>0.0:
43         scale=7.0/(max(array)-minimum_value)
44     else:
45         minimum_value-=3
46         scale=1.0
47     for i in range(len(array)):
48         col = max(0,min(7,7-int(scale*(array[i]-minimum_value)+0.5)))
49         row = i
50         if orientation==0:
51             first=row
52             second=col
53         elif orientation==1:
54             first=col
55             second=7-row
56         elif orientation==2:
57             first=7-row
58             second=7-col
59         else:
60             first=7-col
61             second=row
62         if i == len(array)-1:
63             for i in range(3):
64                 grid.set_pixel(first,second,warning_level)
65                 grid.write_display()
66                 time.sleep(2*plot_speed)
67                 grid.set_pixel(first,second,0)
68                 grid.write_display()
69                 time.sleep(2*plot_speed)
70             grid.set_pixel(first,second,warning_level)
71             grid.write_display()
72         else:
73             grid.set_pixel(first,second,1)
74             grid.write_display()
75             time.sleep(plot_speed)
76     return
77
78
79 ## Displays passed character string on Adafruit 8x8 bicolor LED (with backpack).
80 ## Color is an ineger (1=green, 2=red, 3=orange). Array symbol holds fonts for all
81 ## Alphanumeric ASCII characters and some symbols.
82
83 def scroll_display(input_string,color,orientation,stream_delay):
84     ring_pointer=0
85     insert_point=7
86     ring_pointer=insert_point
87     string_pointer=-1

```

[illegible]

```

180 start_time=time.time()
181 f=open(output_file,'w')
182 f.write(output_file+'\n')
183 f.close()
184
185
186 ## Colors for Adafruit bicolor LED matrix
187
188 green=1
189 orange=3
190 red=2
191
192 while go:
193     try:
194         read_sensors = True
195         while read_sensors:
196             try:
197                 temperature = sensor.read_temperature()
198                 pressure = sensor.read_pressure()
199                 relative_humidity = sensor.read_humidity()
200                 read_sensors = False
201             except IOError:
202                 read_sensors = True
203             show_metric = io.input(metric_switch)
204             scroll_display(datetime.datetime.now().strftime('%H:%M') +
205                             datetime.datetime.now().strftime(' %m-%d-%y'),1,LED_orientation,scroll_speed)
206             if show_metric:
207                 scroll_display('{0:.1f}'.format(temperature)+chr(176)+'C',orange,LED_orientation,scroll_speed)
208             else:
209                 scroll_display('{0:.1f}'.format(1.8*temperature+32)+chr(176)+'F',
210                                 orange,LED_orientation,scroll_speed)
211             if relative_humidity > 40 and relative_humidity < 60: color=green
212             else: color=red
213             scroll_display('{0:.0f}'.format(relative_humidity)+'% ',color,LED_orientation,scroll_speed)
214             current_time=time.time()
215             if current_time-time_in_cycle>=record_time:
216                 time_in_cycle=current_time
217                 f=open(output_file,'a')
218                 f.write(datetime.datetime.now().strftime('%m/%d/%Y %H:%M:%S')+','+'
219                     str(average_temperature/max(1,count))+','+'
220                     str(average_pressure/max(1,count))+','+'
221                     str(average_relative_humidity/max(1,count))+'\n')
222                 f.close()
223                 previous_pressure=average_pressure/max(1,count)
224
225
226 ## "Push down" the record stack, deleting oldest record and enter latest pressure average to record
227
228 for i in range(7):
229     pressure_array[i]=pressure_array[i+1]
230     pressure_array[7]=previous_pressure
231
232
233 ## Reset variables
234
235 count=0
236 average_temperature=0.0
237 average_pressure=0.0
238 average_relative_humidity=0.0
239 else:
240     count+=1
241     average_temperature+=temperature
242     average_pressure+=pressure
243     average_relative_humidity+=relative_humidity
244 if current_time-start_time > 8*record_time:
245     display_trend(pressure_array,LED_Orientation,0.1)
246 else:
247     display_spiral()
248     time.sleep(1)
249 except KeyboardInterrupt:
250     go =False
251     grid.clear()
252     grid.write_display()

```

#Recycled variable to hold time at which the entire program started

#Read temperature from Adafruit BME280

#Read pressure from Adafruit BME280 in Pascals

#Read relative humidity from Adafruit BME280 in %

#Read and display time and date

#Display temperature (Centigrade)

#Display temperature (Fahrenheit)

#Display relative humidity (%)

#Determine and record average temperature and pressure if record_time has elapsed since last output

#If the pressure record array is fully populated, display trend