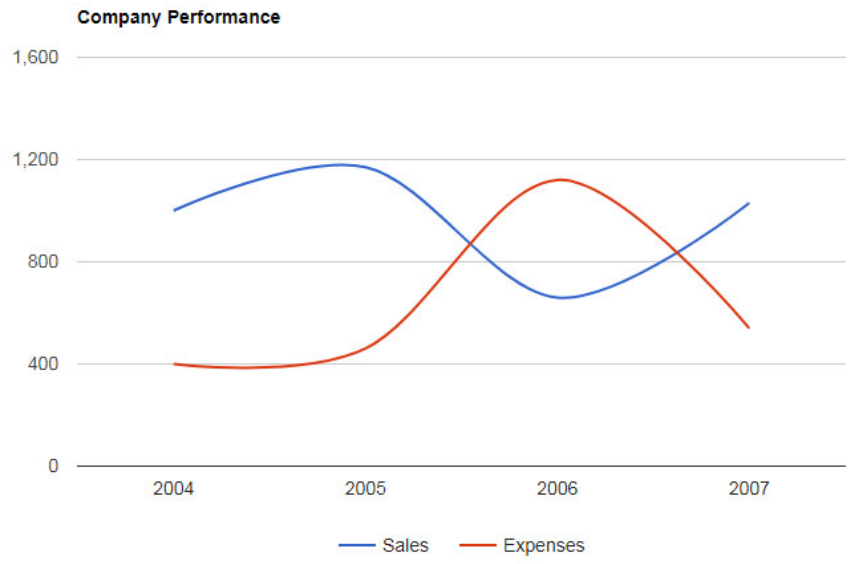
Google has a number of Javascript-powered apps for plotting data on a graph, displayed in a web page. You will use one of those apps in this lesson to graph temperature data collected by your Raspberry Pi. Below is HTML code for a web page containing a Google graph. The graph code is inside a <script> tag in the <head> portion of the page code. On the next page there is an image of what the code looks like displayed in a web browser.

```
<!doctype html>
 <html>
  <head>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
   <script type="text/javascript">
    google.charts.load('current', {'packages':['corechart']});
    google.charts.setOnLoadCallback(drawChart);
    function drawChart() {
     var data = google.visualization.arrayToDataTable([
      ['Year', 'Sales', 'Expenses'],
      ['2004',  1000,      400],
      ['2005',  1170,      460],
      ['2006',  660,       1120],
      ['2007',  1030,      540]
     ]);
     var options = {
      title: 'Company Performance',
      curveType: 'function',
      legend: { position: 'bottom' }
     };
     var chart = new google.visualization.LineChart(document.getElementById('curve_chart'));
     chart.draw(data, options);
    }
   </script>
  </head>
  <body>
   <div id="curve_chart" style="width: 900px; height: 500px"></div>
  </body>
</html>
```



The data of the graph is in this part of the HTML code:

```
['Year', 'Sales', 'Expenses'],
['2004',  1000,     400],
['2005',  1170,     460],
['2006',  660,      1120],
['2007',  1030,     540]
```

If we change the code to that below, then it would graph temperature between the time of 10:12 and 10:21 am. During that time, the temperature rose slowly from 75.3 to 75.6 degrees F.

```
['Time', 'Temp °F'],
['10:12',  75.3],
['10:15',  75.4],
['10:18',  75.5],
['10:21',  75.6]
```

The only thing we would then need to change is the title of the graph, which we find in the code at the line below:

```
title: 'Company Performance',
```

Set up your Raspberry Pi with a temperature sensor on GPIO pin 4. Then copy the code starting on the next page using the Python 2 IDE. The code is commented to help you understand the program.

,

```
#####  Libraries #####
from datetime import datetime  ### this is how we get time data
from time import sleep
import time
import RPi.GPIO as GPIO
import subprocess

#####  Logging Settings #####
FILENAME = ""
WRITE_FREQUENCY = 1

##### Functions #####
def get_dallas_data():  #### function retrieves a temperature reading from sensor
  subprocess.call(['modprobe', 'w1-gpio'])   ### Raspberry Pi takes a temperature reading
  subprocess.call(['modprobe', 'w1-therm'])
  ####  on line below you will need to change 28-0000093b782d to the number for your sensor
  filenameD = "/sys/bus/w1/devices/28-0000093b782d/w1_slave"
  tfile = open(filenameD)  #### open the file that contains the temperature
  text = tfile.read()  ### add the contents of the file to an object named text
  tfile.close()   ### now that the file has been read, close it
  secondline = text.split("\n")[1]  ### spilt the text at the line break (\n), contents of second line
                                    ###assigned to object named secondline
  temperaturedata = secondline.split(" ")[9]  ###split the line at each space, add contents of tenth
                                              ###segment to object named temperaturedata
  temperature = float(temperaturedata[2:])  ### add all but first two characters of temperaturedata to
                                            ###object named temperature, which will be float number
  temperature = temperature / 1000          ### temperature needs to be divided by 1000 to place
                                            ###decimal point
  temp = float(temperature)   ### make temperature a float number assigned to object named temp
  temp = temp*1.8+32  ### convert temperature from Centigrade to Fahrenheit
```

```
  tempDallas = str(round(temp,1))  ### the object tempDallas will contain the temperature as a string,
                                   ###rounded to one place past the decimal point


  return tempDallas  ### return the temperature to the line calling the function


def get_sense_data():  ###function to collect the time and temperature

  sense_data=[]    ### object used to store data

  dateNow = str(datetime.now())  ### get day and time and store in dateNow

  sense_data.append("['" + dateNow[11:16] + "'")  ### add time to the sense_data object, dateNow
                                                  ###contains day and time, so we need to trim
                                                  ###to get just the time [11:16]

  sense_data.append(get_dallas_data() +"],")  ### now add the temperature to the object sense_data
                                              ### we need to end the data line with the characters ],  to
                                              ###keep the graphing code correct.


  return sense_data  ### return the time and temperature to the line calling the function


def log_data():  ###  this function is used to store data as the program runs and repeatedly takes time
                 ###and temperature measurements

  output_string = ", ".join(str(value) for value in sense_data)  ###adds a comma after data

  batch_data.append(output_string)  ### add data to object named batch_data



def file_setup(filename):  ### this function writes the first part of the web page HTML code

  with open(filename,"w") as f:      ### open file in write mode

    f.write('<!doctype html>\n')   ### write first line of HTML code

  with open(filename, "a") as f:  ### open file in append mode and write more lines of code

    f.write('<html>\n')

    f.write('<head>\n')

    f.write('<meta charset="UTF-8">\n')

    f.write('<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>\n')
```

```
    f.write('<script type="text/javascript">\n')
    f.write("google.charts.load('current', {'packages':['corechart']});\n")
    f.write('google.charts.setOnLoadCallback(drawChart);\n')
    f.write('function drawChart() {\n')
    f.write('var data = google.visualization.arrayToDataTable([\n')
    f.write("['Time', 'Temp °F'],\n")   ### at this point we must stop writing the HTML code.  This is the
                                        ###place where the data is inserted into the HTML code, which
                                        ###happens in the Main Program section


##### Main Program ######


batch_data = [ ]  ### create an object that will collect the data
if FILENAME == "":  ### here is where the file name of the web page is generated
  filename = "TempLog-"+str(datetime.now())+".htm"   ### the date will be part of file name
else:
  filename = FILENAME+"-"+str(datetime.now())+".htm"
file_setup(filename)  ### run function named file_setup

### line below gets user input to determine the number of measurements to take
ReadingsToTake = int(input("Enter the number of readings to take: "))
numReadings = 0
dayNow = str(datetime.now())

while numReadings < ReadingsToTake:  ### each time loop runs a time and temperature are done
  numReadings = numReadings + 1 ### counter to stop loop at specified number
  sense_data = get_sense_data()  ### run function to collect data, assign to sense_data
  log_data()   ### run function to log data (store data)
```

```
  if len(batch_data) >= WRITE_FREQUENCY:  ### if something is in bach_data then do loop
    print("Writing to file, please wait..")  ### print this message
    with open(filename, "a") as f:  ###open file containing data in append mode
      for line in batch_data:   #### for each line in batch_data do below
        f.write(line + "\n")   ### write the line to the file and include a line break
        sleep(60)   ### wait 60 seconds then taken next measurement


      batch_data = [ ]  ### clear the contents of this object


with open(filename, "a") as f:  ### at this point the data has been added to the HTML code, now we
                                ### finish writing the last part of the HTML code
    f.write(']);\n')
    f.write('var options = {\n')
   ### notice that the title of graph will contain the current day of year (code: dayNow[0:10])
    f.write("title: 'GEAR Weather Station for " + dayNow[0:10] + "',\n")
    f.write("curveType: 'function',\n")
    f.write("legend: { position: 'bottom' }\n")
    f.write('};\n')
   ###next two lines should be just one line
    f.write("var chart = new
google.visualization.LineChart(document.getElementById('curve_chart'));\n")
    f.write('chart.draw(data, options);\n')
    f.write("}\n")
    f.write("</script>\n")
    f.write("</head>\n")
    f.write("<body>\n")
    f.write('<div id="curve_chart" style="width: 900px; height: 500px"></div>\n')
    f.write("</body>\n")
    f.write("</html>")
print("Program has terminated properly")
```