Objective: To learn how to stream real-time data from sensors on the RPi to Google Visualization Charts and use AJAX to update the data on a URI using Flask without physically refreshing the web page.

Required Setup: Connect GrovePi+ board to RPi and have all GrovePi+ libraries installed, Flask

Parts:

- RPi 3 B+
- GrovePi+ board
- Multiple Grove connection wires
- 1 x Grove Temperature and Humidity Sensor
- 1 x Ultrasonic Ranger
- 1 x Buzzer

buzzer | ultrasonic ranger sensor | temperature/ humidity sensor

Figure 1: GrovePi DHT Sensor
Source: http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/

Overview

In this activity you will learn how to display real-time sensor data via HTML. To this extent, this activity requires that you create a dashboard that utilizes interactive charts or data tools to display temperature and ultrasonic ranger data in real-time. Thankfully, Google provides developers with an easy yet powerful method for creating interactive charts with minimal coding effort. The ultrasonic ranger will cause a buzzer to turn on when an object is detected from a near distance.

*What is an **ultrasonic ranger**?*

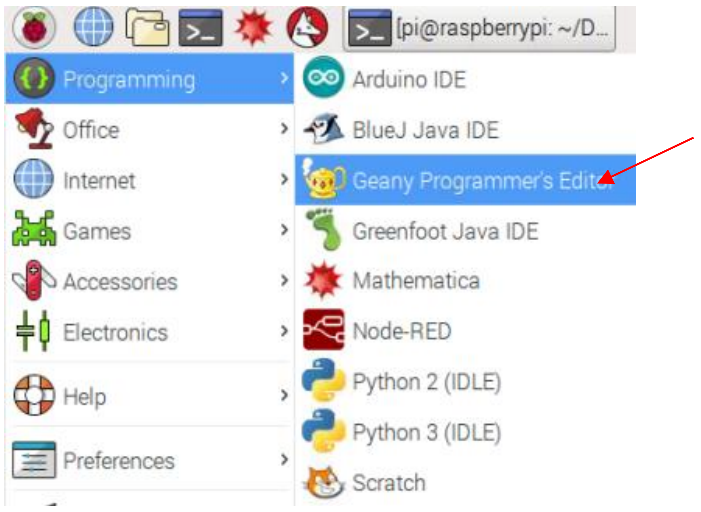
An ultrasonic ranger is a module is a non-contact sensor that is used to measure distance. It emits a signal (or a wave), which will bounce back if it hits a reflective surface (e.g. an object). Hence, we can use an ultrasonic sensor to detect moving objects in its pathway. That is, if you place an object in front of the sensor within the perimeter of the ultrasonic emitted signal range or width, it will report a value, which indicates how close the object (i.e. distance) is from the sensor.

*What is a **GrovePi DHT Sensor**?*

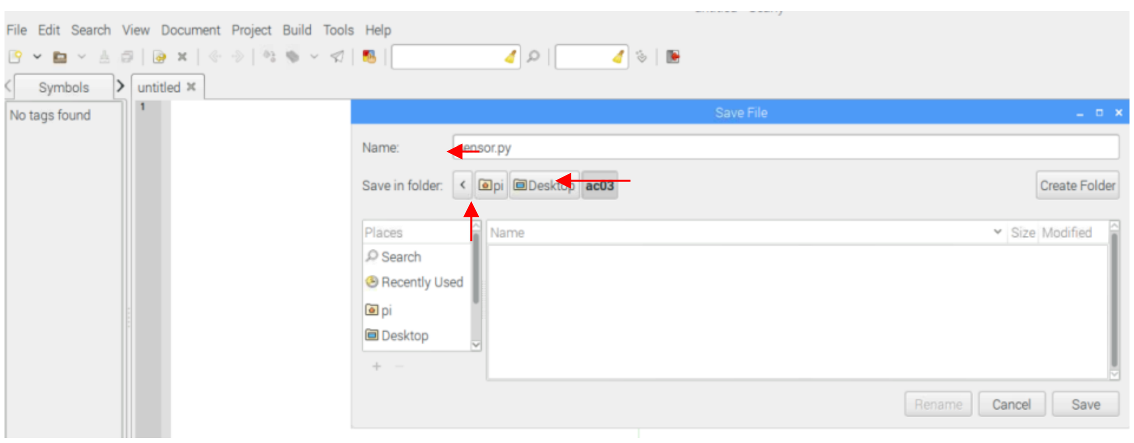
We would like to detect the temperature and humidity using the GrovePi DHT sensor. This capacitive sensor measures relative humidity and temperature. This sensor can be integrated in smart IoT applications such as weather stations, humidity regulator, air conditioners, smart cities, smart buildings, among many others.

Part A: Connect Temperature/Humidity Sensor

1. Connect the DHT sensor any of the digital ports on the GrovePi (e.g. D3, D4, etc.). Then turn on the RPi to begin programming and controlling the sensor.
2. You can use a programming development environment called **Geany Programmer's Editor** (or any IDE on the RPi) to write the necessary script (or using the terminal window) for this activity. If you wish to use the terminal, proceed to the next step.
   To use the IDE, click on the RPI start icon on the taskbar, select Programming → Geany Programmer's Editor



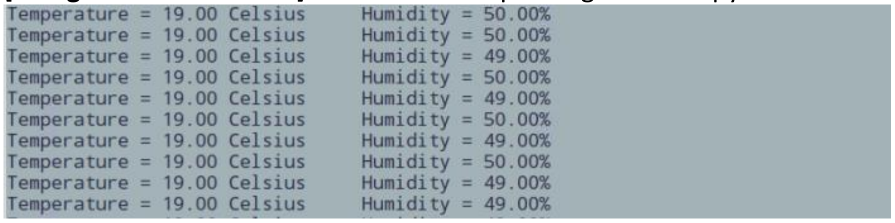
Geany Programmer’s Editor is a lightweight editor for various programming languages including Python. When the program starts, it automatically created untitled file (without any extension). Click on File → Save As → and save this file as **ac03.py** in the ac03 (on the Desktop/ac03) folder.



3. Using the IDE or a terminal window, insert the following code segment into a file named ac03.py:
   Note: If D4 is inaccessible due to the casing, use an alternative digital port to D4.

```
#**********************************#
# TCSS573: IoT                     #
# Activity 03                      #
#**********************************#
# This program will collect        #
# sensor data and store it in a    #
# SQLite database                  #
#**********************************#
# Author: Eyhab Al-Masri           #
# Date:   April 15, 2018           #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
import datetime
import time
import math
import grovepi


#*******************************************#
# Connect the temperature sensor to D4      #
#*******************************************#
sensor = 4  # connect DHT11 sensor to D4

#*******************************************#
# function: read_sensor                     #
# read sensor data from the GPi+ sensor     #
#*******************************************#
def read_sensor():
 try:
   [temp,hum] = grovepi.dht(sensor,0)
   if ((math.isnan(temp) == False) and (math.isnan(hum) == False) and (hum >= 0)):
     temperature = temp;        # copy temp into variable temperature
     humidity = hum
     print("Temperature = %.2f Celsius\tHumidity = %.2f% %" % (temperature, humidity))

 except KeyboardInterrupt:
   print("Exiting ...")

 except IOError as IOe:
   print("An error has occured. %" % IOe)

 except Exception as e:
   print(e)



while (1):
    read_sensor()       # read sensor data
    time.sleep(1)       # delay the next reading by 1 second before next iteration
```

4. [**To execute using Geany Programmer's Editor**] From the Geany Programmer's Editor IDE environment, click on Build Execute (or press F5) to execute ac03.py. Verify that the sensor is now sensing (reading) the temperature and humidity from the surrounding environment. You should see the following output (results will vary).
   [**Using terminal window**] execute the script using the sudo python command:



```
Temperature = 19.00 Celsius     Humidity = 50.00%
Temperature = 19.00 Celsius     Humidity = 50.00%
Temperature = 19.00 Celsius     Humidity = 49.00%
Temperature = 19.00 Celsius     Humidity = 50.00%
Temperature = 19.00 Celsius     Humidity = 49.00%
Temperature = 19.00 Celsius     Humidity = 50.00%
Temperature = 19.00 Celsius     Humidity = 49.00%
Temperature = 19.00 Celsius     Humidity = 50.00%
Temperature = 19.00 Celsius     Humidity = 49.00%
Temperature = 19.00 Celsius     Humidity = 49.00%
```

Part B: Build a Simple Dashboard using Google Visualization Charts

Synopsis

A Python script should continuously detect the **temperature and humidity** as well as detecting any objects that are reflected by the **ultrasonic sensor**. If an object is detected in the pathway of the ultrasonic sensor, it should trigger the buzzer. The level of the buzzer depends on how far the object is from the ultrasonic sensor.

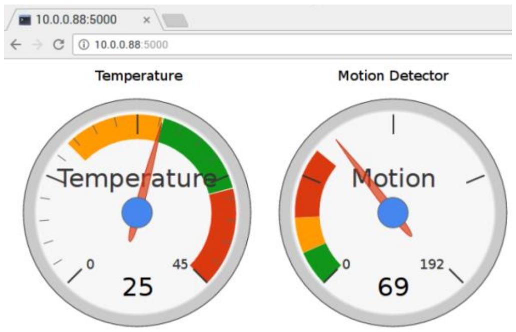
How to get started?

1) The Python script is used to collect sensor data and routes this information via a REST service. In order to send this data via a REST service, this activity requires the use of Flask library. A sample Python (ac3.py) and HTML (gauge.html) files can be downloaded from Canvas → Activities → Activity 3.

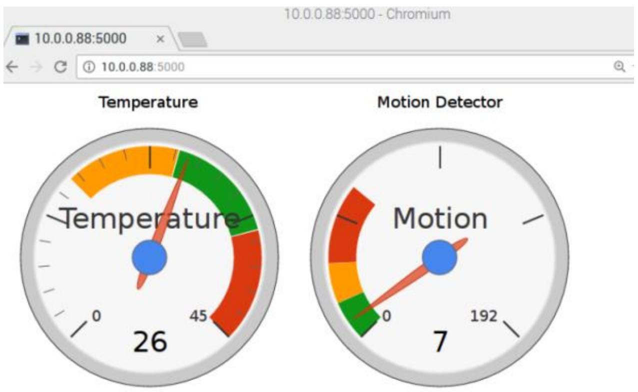
   The **Python script** (ac3.py) should collect temperature, humidity and ultrasonic ranger values. Then, this data will be posted in real-time to the **HTML web page** (gauge.html) which uses Google Charts to display **guage**(**s**) of the real-time data sent via the REST service of the Python script.

2) Google provides simple tools for enriching webpages with interactive charts and data tools. To this extent, we would like to use Google Charts to display the sensor data collected via the Python script. Hence, the **HTML web page** should display two gauges: (a) one for temperature and (b) one for the ultrasonic ranger. As the temperature or ultrasonic ranger changes in values, the gauges should display in real-time this on the Google charts. An illustrative example is shown below:

   The URL should display two Google Chart gauges: (a) one for temperature and (b) one for ultrasonic ranger.



   When an object comes near the ultrasonic ranger (e.g. let's say 20cm), the buzzer beeps and the gauge displays the real-time distance (i.e. in cm).



Hints:

- When creating a gauge for the ultrasonic ranger, create a **separate chart** from that of the temperature to be able to control the visibility of the chart/gauge. Charts should be displayed horizontally (as shown above, use HTML table).
- In order to send the data to the browser in real-time using AJAX, use the **`setInterval`** function which creates a variable that executes a URL. This URL is simply calling a function in the Python script. To illustrate this, check the Python script, you will see a function called **`getTemp`** with the route '**`getTemp`**'. When the URL http://10.0.0.0:5000 is accessed, it calls the **`getStatus`**. But since we do not have a while (true), this function is only called once. Hence, we would like to "push" the data onto the web page using AJAX. But in order to get the latest temperature, we need to create a separate function (e.g. **`getTemp`**) such that it only returns the value of the temperature. (There are many ways we can accomplish this functionality, but this is one of the simplest ways; we can, for example, read the data from the database!).
- To align the gauges next to each other, use HTML tables! To create HTML tables, go to https://htmlg.com/html-editor/ and create two rows (first row used as a header, second row used to display the gauges).
- To learn more about the benefits of AJAX, go to https://www.w3schools.com/xml/ajax_intro.asp
- To learn more about Google Charts for Developers, go to https://developers.google.com/chart/

Deliverable (upload to canvas → activities → activity 3) *(only one submission is required per group)*

☐	Python script (file ac3.py) → for Part B implementation
☐	HTML file (file gauge.html)
☐	short video exhibiting working prototype

Grading

☐	correct output (5 marks)
☐	real-time sensitivity of the gauges (2 marks)
☐	smooth buzzer rhythm based on object detection (2 marks)
☐	gauges display horizontally in tabular format (1 mark)

References (some content and images are adopted from the following reference):
- http://wiki.seeedstudio.com/Grove-TemperatureAndHumidity_Sensor/
- https://lawsie.github.io/guizero