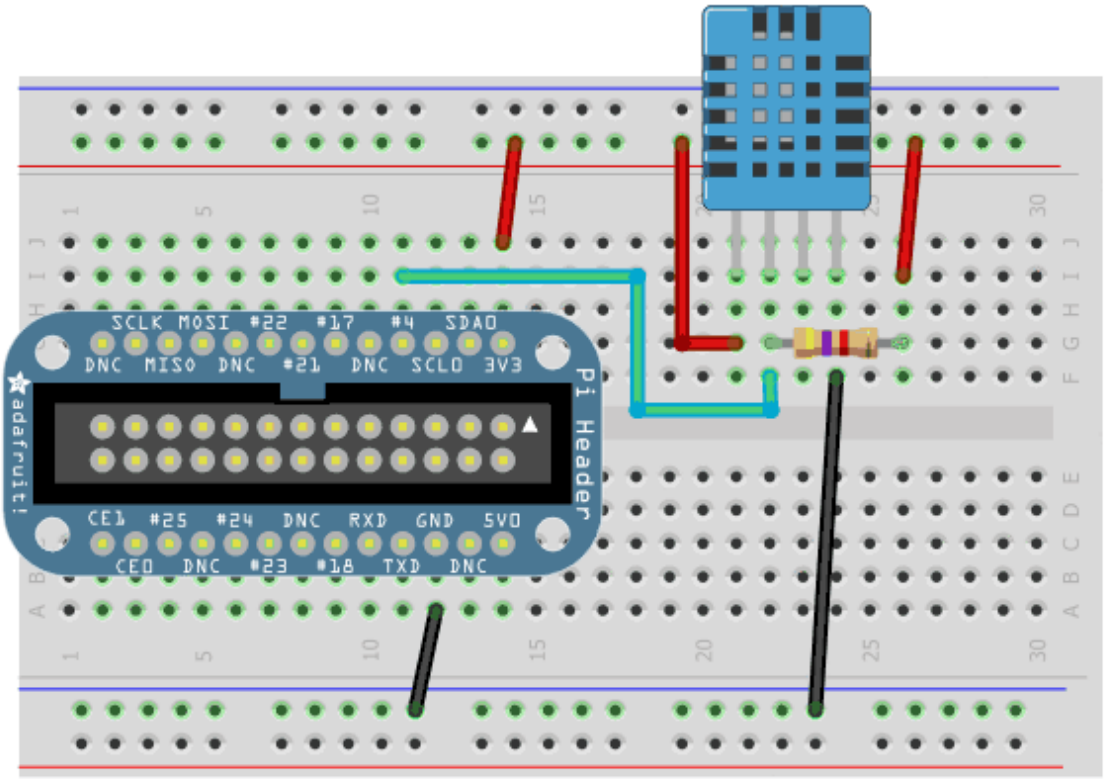
DHT Humidity Sensing on Raspberry Pi or Beaglebone Black with GDocs Logging

Created by lady ada



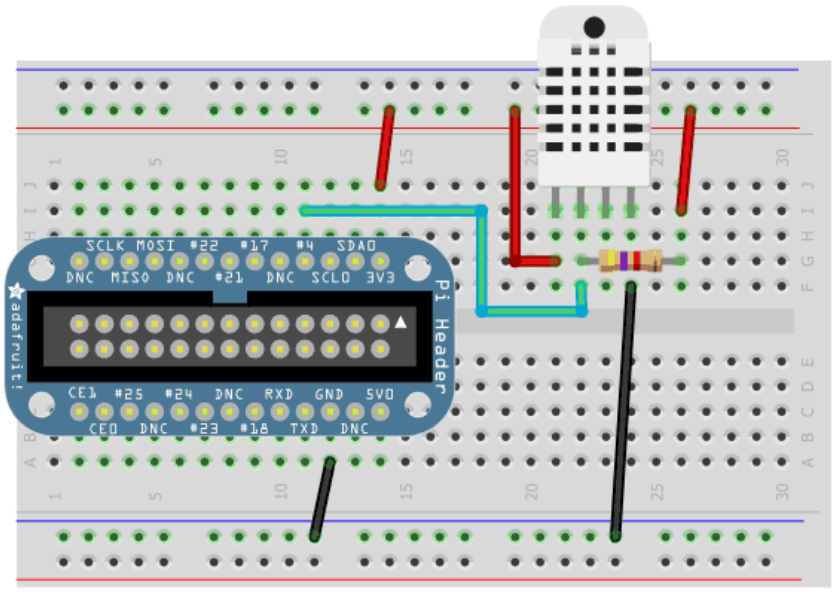
https://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging

Last updated on 2023-08-29 02:11:09 PM EDT

Table of Contents

Overview

This tutorial is a first attempt to develop a DHT interface driver. It is not guaranteed to work, and is for experimentation and hacking!



Time to start exploring more sensors with the Raspberry Pi and Beaglebone Black! Today we'll be checking out the DHT11 (http://adafru.it/386), DHT22 (http://adafru.it/385) and AM2302 (http://adafru.it/393) humidity and temperature sensors available from Adafruit

In this tutorial we'll be showing how to install a DHT sensor Python library which utilizes C for high-speed GPIO polling to handle bit-banged sensor output. Many low cost sensors have unusual output formats, and in this case, a "Manchester-esque" output that is not SPI, I2C or 1-Wire compatible must be polled continuously by the Pi to decode. Luckily, the C GPIO libraries are fast enough to decode the output.

Once we have that working, we add the fun of Python to update a google spreadsheet live with the temperature/humidity data. This project would be the great basis for home or garden automation!

You can check out our spreadsheet here, it wont be updated live after Aug 24 2012 but it will show you the format of data you get ()

Wiring

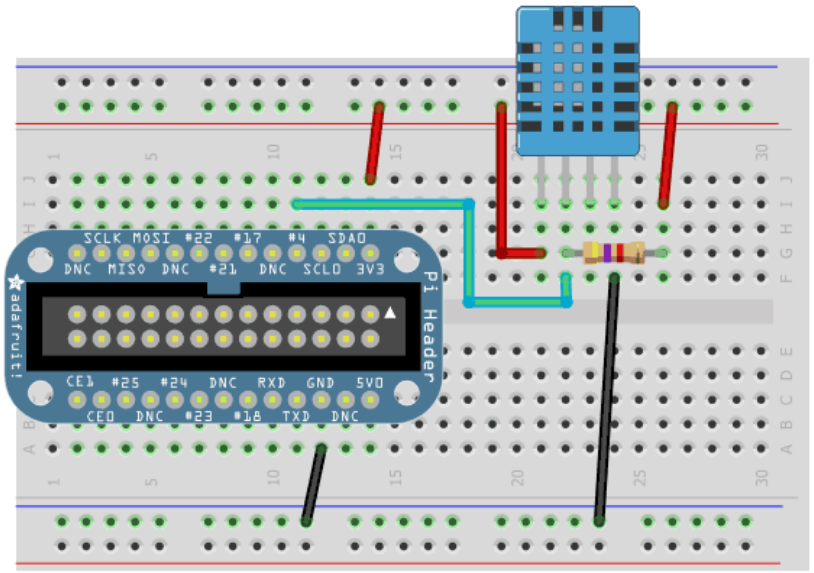
Wiring up DHT humidity sensors

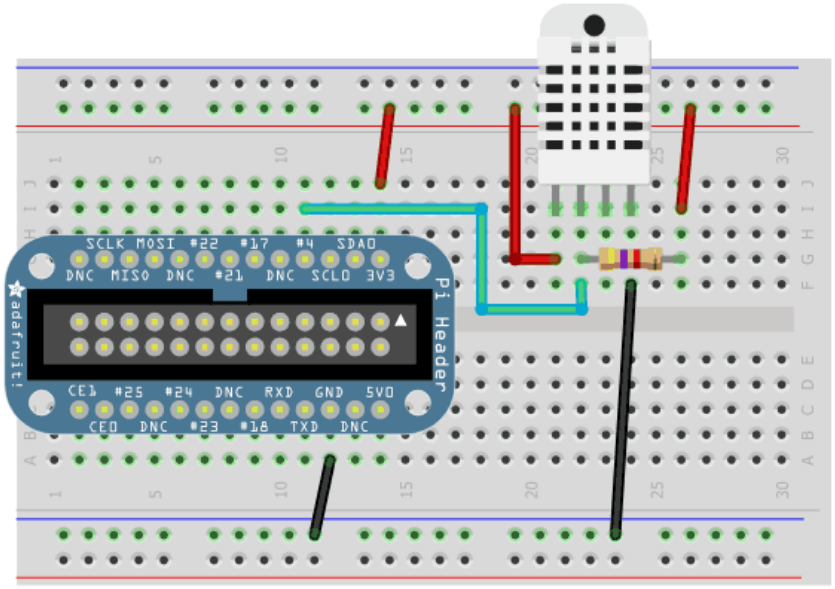
Raspberry Pi

Its easy to connect these sensors to your Raspberry Pi. Our code can use any GPIO pin, but we'll be using GPIO #4 for our diagrams and code. Once you have it working, you can simply adapt the code to change to any other GPIO pin (e.g. pin #18). You can also have as many DHT sensors as you want but they cannot share the data pin - each sensor needs a unique data pin!

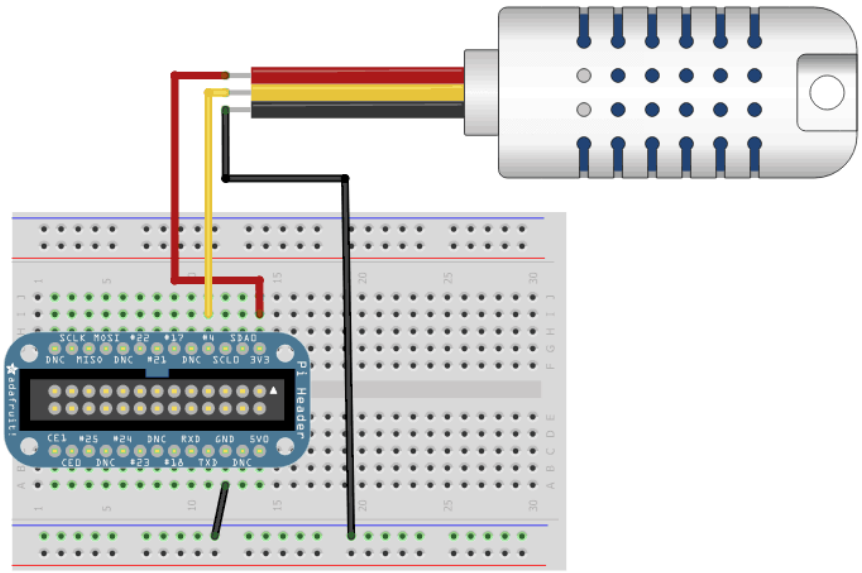
For DHT11 and DHT22 sensors, don't forget to connect a 4.7K - 10K resistor from the data pin to VCC

& if 4.7K doesnt work, try 10K





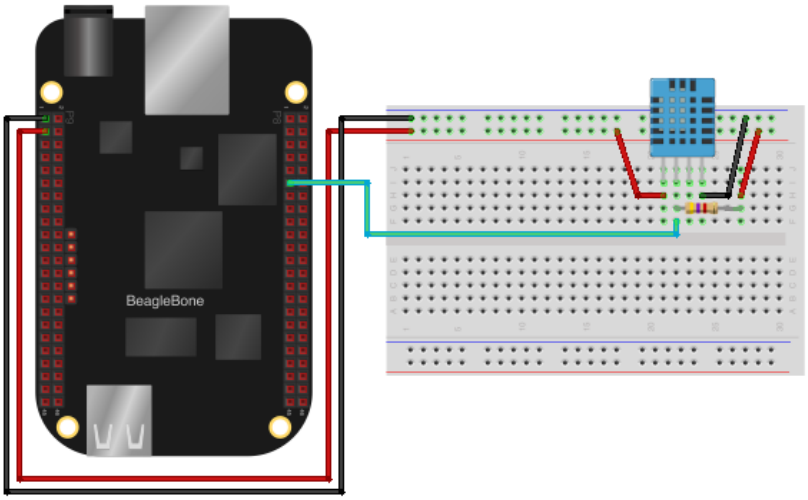
If your sensor has a white wire, leave it disconnected

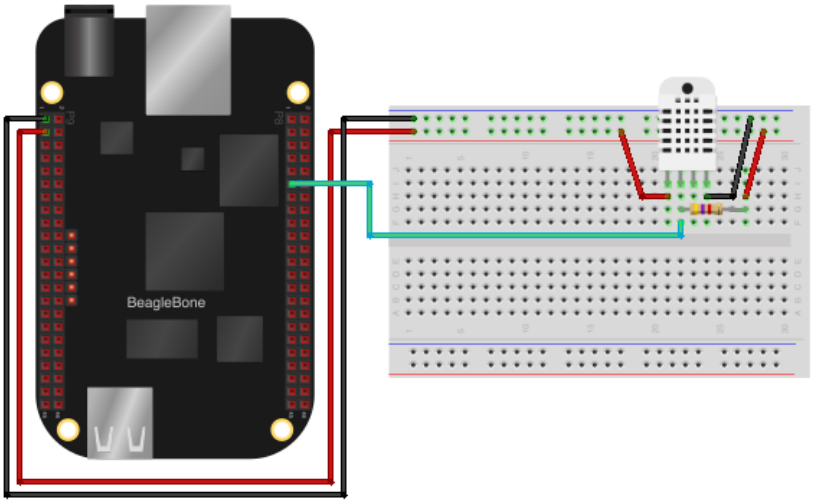


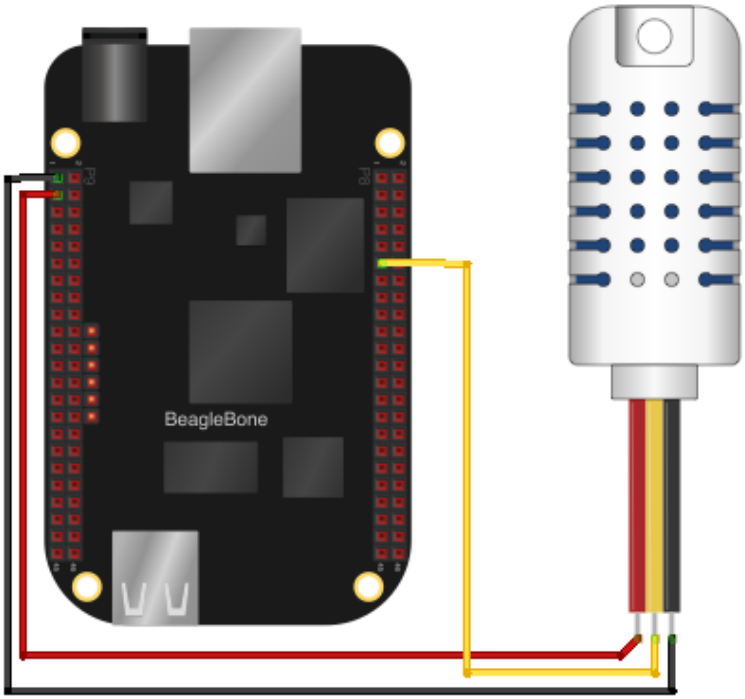
Beaglebone Black

Connecting a DHT sensor to the Beaglebone Black is just as simple as connecting to a Raspberry Pi. In this example pin P8_11 will be used, but you can use any free digital GPIO pin. If you aren't familiar with how pins are numbered on the Beaglebone Black, check out the tutorial on Beaglebone Black GPIO ().

Don't forget to connect the 4.7k-10k resistor from the data pin to VCC, like with the Raspberry Pi wiring above.


BeagleBone


BeagleBone



Python Setup

We're going to use a special library called adafruit_blinka () (named after Blinka, the CircuitPython mascot ()) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides.

Installing CircuitPython Libraries on Raspberry Pi or BeagleBone Black

If you haven't set up your Raspberry Pi or BeagleBone Black for running CircuitPython libraries yet, follow our guide and come back to this page when you've completed the steps listed on the page and verified that your setup is working:

- Setup your Linux Board for using CircuitPython Libraries ()

Installing the CircuitPython-DHT Library

You'll also need to install a library to communicate with the DHT sensor. Since we're using Adafruit Blinka (CircuitPython), we can install CircuitPython libraries straight to our small linux board. In this case, we're going to install the CircuitPython_DHT library. This library works with both the DHT22 and DHT11 sensors.

Run the following command to install the CircuitPython-DHT library ():

```
pip3 install adafruit-circuitpython-dht
```

```
sudo apt-get install libgpiod2
```

Testing the CircuitPython DHT Library

To make sure you've installed everything correctly, we're going to test that we can read values from the DHT sensor connected to your device.

Create a new file called dht_simpletest.py with nano or your favorite text editor and put the following in:

```
# SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
# SPDX-License-Identifier: MIT

import time
import board
import adafruit_dht

# Initial the dht device, with data pin connected to:
dhtDevice = adafruit_dht.DHT22(board.D18)

# you can pass DHT22 use_pulseio=False if you wouldn't like to use pulseio.
# This may be necessary on a Linux single board computer like the Raspberry Pi,
# but it will not work in CircuitPython.
# dhtDevice = adafruit_dht.DHT22(board.D18, use_pulseio=False)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} F / {:.1f} C    Humidity: {}% ".format(
                temperature_f, temperature_c, humidity
            )
        )

    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)
```

Next, you're going to need to modify a line of code in this file with information about the pin the DHT sensor is connected to and the type of DHT sensor you're using.

If you're using a Raspberry Pi with a DHT22 (or an AM2302) sensor connected to Pin 4, change the following line from:

```
dhtDevice = adafruit_dht.DHT22(board.D18)
```

to

```
dhtDevice = adafruit_dht.DHT22(board.D4)
```

If you're using a BeagleBone Black with a DHT22 (or an AM2302) sensor connected to Pin P8_11, change the following line from

```
dhtDevice = adafruit_dht.DHT22(board.D18)
```

to

```
dhtDevice = adafruit_dht.DHT22(board.P8_11)
```

If you're using a DHT11 sensor, you can change the sensor type by renaming the DHT22 class to DHT11:

```
dhtDevice = adafruit_dht.DHT11(board.D18)
```

Then, save the example. Next, run the example by typing the following command into the terminal:

```
python3 dht_simpletest.py
```

Press enter to run the example. You should see the temperature (in Fahrenheit and Celsius) and humidity values displayed in the terminal:

```
Temp: 73.4 F / 23.0 C    Humidity: 40.3%
Temp: 73.4 F / 23.0 C    Humidity: 40.3%
```

If your output looks like the output above - your linux board is set up for use with CircuitPython libraries and can read values from a connected DHT sensor.

If you do not see the correct values printed or if the script is unable to create a dhtDevice object

Check that the wiring is correct (Are ground and power connected? is the correct DHT pin pulled up to 3.3v?). Then, make sure you provided the `dhtDevice` with the correct physical pin number your board is connected to.

If you're receiving ImportError: No module named 'adafruit_dht'

Make sure you installed the CircuitPython DHT library with pip3 and are running the command with python3.

If running the script returns ImportError: No module named 'board'

Make sure you're running the script with python3. If you're still receiving this error, follow the setup guide for installing CircuitPython libraries () again.

Connecting to Google Docs

As of April 2015 Google has deprecated an old authentication interface for updating Google Sheets. You must carefully read the new steps below to make your Google Sheets work with the new OAuth2 authentication scheme.

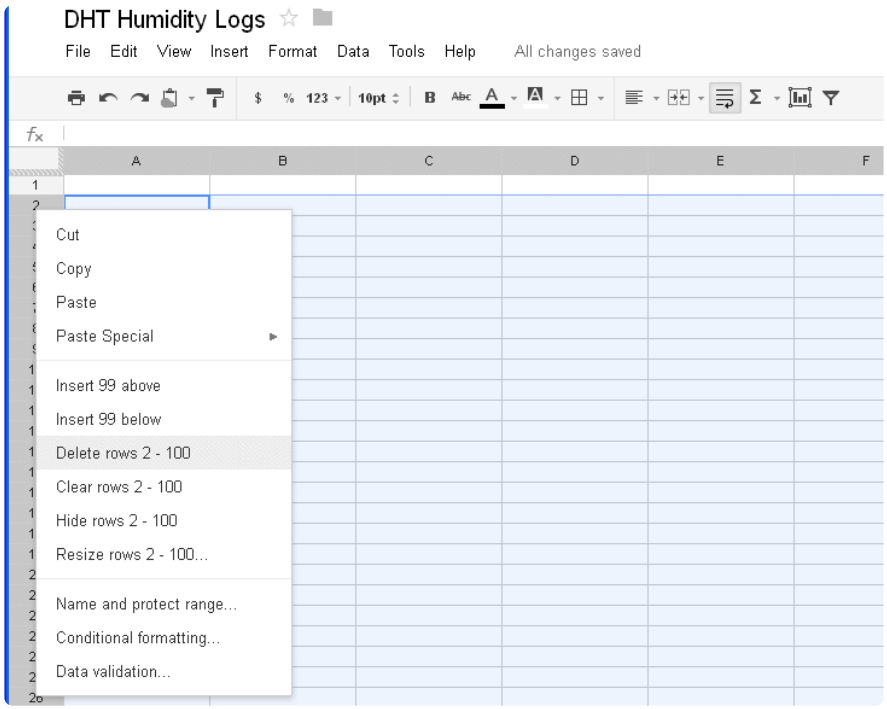
Google sometimes will update their API and cause issues with the gspread library. Consult the following thread for information on converting a spreadsheet to an old-style spreadsheet if you have problems accessing your sheet:

RE: DHT 22 Temperature and Humidity sensor with adafruit code ()

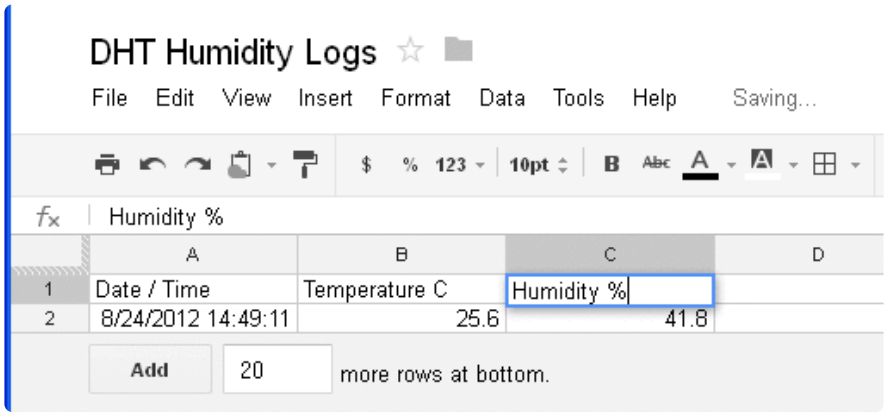
Create and prepare spreadsheet

First up you will need to sign up for Google Docs () and create a spreadsheet. We're going to call ours DHT Humidity Logs.

Once you've created it, delete all but one line (since we don't want 1000 empty rows):



Then make the one remaining line a header with row names:



Get OAuth2 credentials

As of April 2015 Google has deprecated the older simple authentication interface for accessing Google spreadsheet data. You must carefully follow the steps below to enable OAuth2 access to your Google spreadsheet. Unfortunately these steps are somewhat complex, so go through them very carefully to make sure you don't miss a step. If you run into problems try consulting the gspread python library () that this script uses.

To get your OAuth2 credentials follow the steps on this page:

- gspread - Using OAuth2 for Authorization ()

After you follow the steps in the document above you should have downloaded a .json file, like `SpreadsheetData-(gibberish).json`. Place this .json file in the same directory as the **`google_spreadsheet.py`** example. If you don't place this file in the same directory then authentication will fail and you will not be able to update your spreadsheet!

One last step that must be completed is to share your Google spreadsheet to the email address associated with the OAuth2 credentials. Open the .json file and search for the "client_email": line that looks like this (but with a different email address):

```
"client_email": "149345334675-
md0qff5f0kib41meu20f7d1habos3qcu@developer.gserviceaccount.com",
```

Take note of that email address value and go to your Google spreadsheet in a web browser. Using the File -> Share... menu item share the spreadsheet with read and write access to the email address found above. Make sure to share your spreadsheet or you will not be able to update it with the script!

Run Python Code

First up we will have to install the gspread python library, which will do the heavy lifting of connecting to google docs and updating the spreadsheet! With your board connected and online, run the following:

```
sudo pip3 install gspread oauth2client pyasn1 pyasn1-modules
```

Create a new file called google_spreadsheet.py with nano or your favorite text editor and put the following in:

```
# SPDX-FileCopyrightText: 2019 Tony DiCola for Adafruit Industries
#
# SPDX-License-Identifier: MIT

"""
Google Spreadsheet DHT Sensor Data-logging Example

Copyright (c) 2014 Adafruit Industries
Author: Tony DiCola
Modified by: Brent Rubell for Adafruit Industries

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

```
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
"""
import sys
import time
import datetime

import board
import adafruit_dht
import gspread
from oauth2client.service_account import ServiceAccountCredentials

# Type of sensor, can be `adafruit_dht.DHT11` or `adafruit_dht.DHT22`.
# For the AM2302, use the `adafruit_dht.DHT22` class.
DHT_TYPE = adafruit_dht.DHT22

# Example of sensor connected to Raspberry Pi Pin 23
DHT_PIN  = board.D4
# Example of sensor connected to Beaglebone Black Pin P8_11
# DHT_PIN  = 'P8_11'

# Initialize the dht device, with data pin connected to:
dhtDevice = DHT_TYPE(DHT_PIN)

# Google Docs OAuth credential JSON file.  Note that the process for authenticating
# with Google docs has changed as of ~April 2015.  You _must_ use OAuth2 to log
# in and authenticate with the gspread library.  Unfortunately this process is much
# more complicated than the old process.  You _must_ carefully follow the steps on
# this page to create a new OAuth service in your Google developer console:
#   http://gspread.readthedocs.org/en/latest/oauth2.html
#
# Once you've followed the steps above you should have downloaded a .json file with
# your OAuth2 credentials.  This file has a name like SpreadsheetData-
<gibberish>.json.
# Place that file in the same directory as this python script.
#
# Now one last _very important_ step before updating the spreadsheet will work.
# Go to your spreadsheet in Google Spreadsheet and share it to the email address
# inside the 'client_email' setting in the SpreadsheetData-*.json file.  For example
# if the client_email setting inside the .json file has an email address like:
#   149345334675-md0qff5f0kib41meu20f7d1habos3qcu@developer.gserviceaccount.com
# Then use the File -> Share... command in the spreadsheet to share it with read
# and write acess to the email address above.  If you don't do this step then the
# updates to the sheet will fail!
GDOCS_OAUTH_JSON       = 'your SpreadsheetData-*.json file name'

# Google Docs spreadsheet name.
GDOCS_SPREADSHEET_NAME = 'DHT'

# How long to wait (in seconds) between measurements.
FREQUENCY_SECONDS      = 30


def login_open_sheet(oauth_key_file, spreadsheet):
    """Connect to Google Docs spreadsheet and return the first worksheet."""
    try:
        scope =  ['https://spreadsheets.google.com/feeds', 'https://
www.googleapis.com/auth/drive']
        credentials =
ServiceAccountCredentials.from_json_keyfile_name(oauth_key_file, scope)
        gc = gspread.authorize(credentials)
        worksheet = gc.open(spreadsheet).sheet1 # pylint: disable=redefined-outer-
name
        return worksheet
    except Exception as ex: # pylint: disable=bare-except, broad-except
```

```
        print('Unable to login and get spreadsheet.  Check OAuth credentials,
spreadsheet name, \
        and make sure spreadsheet is shared to the client_email address in the
OAuth .json file!')
        print('Google sheet login failed with error:', ex)
        sys.exit(1)


print('Logging sensor measurements to\
 {0} every {1} seconds.'.format(GDOCS_SPREADSHEET_NAME, FREQUENCY_SECONDS))
print('Press Ctrl-C to quit.')
worksheet = None
while True:
    # Login if necessary.
    if worksheet is None:
        worksheet = login_open_sheet(GDOCS_OAUTH_JSON, GDOCS_SPREADSHEET_NAME)

    # Attempt to get sensor reading.
    temp = dhtDevice.temperature
    humidity = dhtDevice.humidity

    # Skip to the next reading if a valid measurement couldn't be taken.
    # This might happen if the CPU is under a lot of load and the sensor
    # can't be reliably read (timing is critical to read the sensor).
    if humidity is None or temp is None:
        time.sleep(2)
        continue

    print('Temperature: {0:0.1f} C'.format(temp))
    print('Humidity:    {0:0.1f} %'.format(humidity))

    # Append the data in the spreadsheet, including a timestamp
    try:
        worksheet.append_row((datetime.datetime.now().isoformat(), temp, humidity))
    except: # pylint: disable=bare-except, broad-except
        # Error appending data, most likely because credentials are stale.
        # Null out the worksheet so a login is performed at the top of the loop.
        print('Append error, logging in again')
        worksheet = None
        time.sleep(FREQUENCY_SECONDS)
        continue

    # Wait 30 seconds before continuing
    print('Wrote a row to {0}'.format(GDOCS_SPREADSHEET_NAME))
    time.sleep(FREQUENCY_SECONDS)
```

Next, in the examples directory again, edit google_spreadsheet.py and adjust the configuration values towards the top of the file:

```
# Type of sensor, can be adafruit_dht.DHT11 or adafruit_dht.DHT22.
# For the AM2302, use the adafruit_dht.DHT22 class.
DHT_TYPE = adafruit_dht.DHT22

# Example of sensor connected to Raspberry Pi Pin 23
DHT_PIN  = board.D4
# Example of sensor connected to Beaglebone Black Pin P8_11
# DHT_PIN  = 'P8_11'

# Google Docs OAuth credential JSON file.  Note that the process for authenticating
# ...
GDOCS_OAUTH_JSON       = 'your SpreadsheetData-*.json file name'

# Google Docs spreadsheet name.
GDOCS_SPREADSHEET_NAME = 'your google docs spreadsheet name'
```

Make sure `DHT_TYPE` is set to the type of sensor you are using (either `adafruit_dht.DHT11` or `adafruit_dht.DHT22`), and `DHT_PIN` is set to the GPIO pin number which is connected to your DHT sensor. If you're using an AM2302, use the `adafruit_dht.DHT22` class.

In the example above a Raspberry Pi GPIO pin #23 is shown, however commented below it is an example of a Beaglebone Black using GPIO pin P8_11.

Next make sure to set the `GDOCS_OAUTH_JSON` to the name of the `SpreadsheetData-*.json` file in the same directory as the `google_spreadsheet.py` file. If you don't have a `SpreadsheetData-*.json` file then you accidentally missed the steps above. Go back and carefully follow the OAuth2 credential steps () to get an OAuth2 credential .json file before continuing!

Finally set `GDOCS_SPREADSHEET_NAME` to the name of your spreadsheet, like 'DHT Humidity Logs'.

Save the file and execute the Python script by running:

```
python3 google_spreadsheet.py
```

You should see the program run and after about 30 seconds a humidity and temperature measurement is displayed and written to the spreadsheet. The program will continue to run and log a measurement every 30 seconds until you force it to quit by pressing `Ctrl-C`.

The measurement frequency can be adjusted by changing the `FREQUENCY_SECONDS` configuration in the python code.

Open the spreadsheet on Google's site and you should see measurements added in real time!

You can also see our spreadsheet here, it wont be running live after Aug 24, 2012 but it gives you an idea of the data format ()