[PROJECTS](https://learn.adafruit.com/category/projects) [RASPBERRY PI](https://learn.adafruit.com/category/raspberry-pi)/[PI A+, B+, 2, 3](https://learn.adafruit.com/category/pi-b-plus-2-3) [LCDS & DISPLAYS](https://learn.adafruit.com/category/lcds-and-displays) / [CHARACTER DISPLAYS](https://learn.adafruit.com/category/character-displays)

<https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io>

Adafruit IO

[Tony DiCola](https://learn.adafruit.com/users/tdicola)

Now that you have a cool physical dashboard built, how do you hook it up to Adafruit IO?  Check out the video below that walks through creating a simple Python script to display Adafruit IO feeds on a physical dashboard.  Below the video are the basic steps to follow and the code to use.

Setup

To hook up the Pi physical dashboard to Adafruit IO you'll first need to do a little bit of setup.  Before you start make sure you have the physical dashboard project assembled and working (using curl or other tools to test it).  In addition you'll want to have an account on Adafruit IO setup (visit [http://io.adafruit.com](http://io.adafruit.com/) to join the beta), and it will help to familiarize yourself with Adafruit IO feeds and dashboards by checking out [these AIO basics guides](https://learn.adafruit.com/series/adafruit-io-basics).

Next you'll need to connect to the Pi and run the following commands to install a Python Adafruit IO client library:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604360/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. sudo pip install adafruit-io
2. sudo pip install requests

Now configure your physical dashboard with the gauges and widgets you'd like to display.  For this example I'm using the following dashbord config.ini:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604362/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. [slider]
2. type = SevenSegmentWidget
3. address = 0x74
4. decimal\_digits = 0
6. [humidity]
7. type = SevenSegmentWidget
8. address = 0x76
9. decimal\_digits = 2
11. [temp]
12. type = SevenSegmentWidget
13. address = 0x71
14. decimal\_digits = 1

This configuration defines the following three 7-segment displays:

* **slider** - A 7-segment display at I2C address 0x74 with no digits after the decimal point displayed.
* **humidity** - A 7-segment display at I2C adress 0x76 with 2 digits after the decimal point displayed.
* **temp** - A 7-segment display at I2C address 0x71 with 1 digit after the decimal point displayed.

Code

To hook up the dashboard to display Adafruit IO feeds you can create a simple Python script similar to the one shown below.  For example create a script called [**aio\_dashboard.py**](https://github.com/adafruit/Pi_Physical_Dashboard) on the Raspberry Pi and fill in:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604365/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. # Raspberry Pi Physical Dashboard & Adafruit IO Example
2. # Author: Tony DiCola
3. # License: Public domain
4. #
5. # This example will display three feeds from Adafruit IO on a dashboard. The
6. # feeds are:
7. # - pi-dashboard-slider: Will display a slider with values from 0-100 on a
8. # dashboard 7-segment display called 'slider'.
9. # - pi-dashboard-humidity: Humidity sensor feed displayed on a 7-segment display
10. # called 'humidity'.
11. # - pi-dashboard-temp: Temperature sensor feed displayed on a 7-segment display
12. # called 'temp'.
13. #
14. # To modify this example to use your own feeds you'll want to change the following
15. # code:
16. # - connected function: Add a client.subscribe() call for each feed you want to
17. # display on the dashboard.
18. # - message function: Add an if/elif check to check for each feed you want
19. # to display and have it make an HTTP POST against the
20. # dashboard using the feed payload as a value.
21. # - DASHBOARD\_URL variable: Change this to the URL of your Pi dashboard if you
22. # aren't running this script on the same Pi as the
23. # dashboard (otherwise keep it the same localhost:5000
24. # value). Make sure NOT to end this with a slash!
25. #
26. # Be sure to modify the ADAFRUIT\_IO\_KEY and ADAFRUIT\_IO\_USERNAME variables to
27. # set your AIO key and username!
29. # Import standard python modules.
30. import random
31. import sys
32. import time
34. # Import Adafruit IO MQTT client.
35. from Adafruit\_IO import MQTTClient
37. # Import requests library used for making HTTP calls to the dashboard server.
38. import requests

41. # Set to your Adafruit IO key & username below.
42. ADAFRUIT\_IO\_KEY = 'YOUR ADAFRUIT IO KEY' # Set to your Adafruit IO key.
43. ADAFRUIT\_IO\_USERNAME = 'YOUR ADAFRUIT IO USERNAME' # See https://accounts.adafruit.com
44. # to find your username.
45. # Set the URL of the physical dashboard to use. If running on the same Pi as
46. # the dashboard server then keep this the default localhost:5000 value. If
47. # modified make sure not to end in a slash!
48. DASHBOARD\_URL = 'http://localhost:5000' # URL of the physical dashboard.
49. # Don't end with a slash!

52. # Define callback functions which will be called when certain events happen.
53. def connected(client):
54. # Connected function will be called when the client is connected to Adafruit IO.
55. # This is a good place to subscribe to feed changes. The client parameter
56. # passed to this function is the Adafruit IO MQTT client so you can make
57. # calls against it easily.
58. print 'Connected to Adafruit IO! Listening for feed changes...'
59. # Subscribe to the three pi-dashboard feeds that will be displayed on the
60. # dashboard. Modify this to subscribe to all the feeds you want to display.
61. client.subscribe('pi-dashboard-slider')
62. client.subscribe('pi-dashboard-humidity')
63. client.subscribe('pi-dashboard-temp')
65. def disconnected(client):
66. # Disconnected function will be called when the client disconnects.
67. print 'Disconnected from Adafruit IO!'
68. sys.exit(1)
70. def message(client, feed\_id, payload):
71. # Message function will be called when a subscribed feed has a new value.
72. # The feed\_id parameter identifies the feed, and the payload parameter has
73. # the new value.
74. print('Feed {0} received new value: {1}'.format(feed\_id, payload))
75. # Update physical dashboard depending on the changed feed.
76. # Notice the feed\_id is checked to find out which feed changed, then the
77. # appropriate physical dashboard widget is changed.
78. if feed\_id == 'pi-dashboard-slider':
79. # The requests.post function will make an HTTP request against the
80. # dashboard. See the requests documentation for more information:
81. # http://docs.python-requests.org/en/latest/
82. requests.post('{0}/widgets/slider'.format(DASHBOARD\_URL), data={'value': payload})
83. elif feed\_id == 'pi-dashboard-humidity':
84. requests.post('{0}/widgets/humidity'.format(DASHBOARD\_URL), data={'value': payload})
85. elif feed\_id == 'pi-dashboard-temp':
86. requests.post('{0}/widgets/temp'.format(DASHBOARD\_URL), data={'value': payload})

89. # Create an MQTT client instance.
90. client = MQTTClient(ADAFRUIT\_IO\_USERNAME, ADAFRUIT\_IO\_KEY)
92. # Setup the callback functions defined above.
93. client.on\_connect = connected
94. client.on\_disconnect = disconnected
95. client.on\_message = message
97. # Connect to the Adafruit IO server.
98. client.connect()
100. # Use the loop\_blocking function to run the message loop for processing Adafruit
101. # IO events. Since this script doesn't do any other processing this blocking
102. # version of the message loop is fine. All the program logic will occur in the
103. # callback functions above when Adafruit IO feeds are changed.
104. client.loop\_blocking()

This script is based on the [**mqtt\_client.py**](https://github.com/adafruit/io-client-python/blob/master/examples/mqtt_client.py) example from the Adafruit IO Python library, but with a few small changes to make it send feed data to a dashboard.  You'll want to modify a few parts of the script as described below.

First set your Adafruit IO key and username in this section near the top of the script:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604367/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. # Set to your Adafruit IO key & username below.
2. ADAFRUIT\_IO\_KEY = 'YOUR ADAFRUIT IO KEY' # Set to your Adafruit IO key.
3. ADAFRUIT\_IO\_USERNAME = 'YOUR ADAFRUIT IO USERNAME' # See https://accounts.adafruit.com
4. # to find your username.

(you might have noticed in the video above I load the AIO key from a local file, however it's not really necessary to load from a file unless you'd like to keep the key out of the script code)

Next if you're running this script outside of the Pi that's running your dashboard you'll need to change the **DASHBOARD\_URL** variable below:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604369/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. # Set the URL of the physical dashboard to use. If running on the same Pi as
2. # the dashboard server then keep this the default localhost:5000 value. If
3. # modified make sure not to end in a slash!
4. DASHBOARD\_URL = 'http://localhost:5000' # URL of the physical dashboard.
5. # Don't end with a slash!

Make sure the **DASHBOARD\_URL** value is set to the IP address or hostname of your Pi that's running the dashboard.  If you're running this script on the same Pi as the dashboard then leave it as [http://localhost:5000](http://localhost:5000/).  Make sure this URL does **not** end in a slash!

Now modify the **connected** function code to subscribe to all the feeds you wish to display on the dashboard.  This function will be called when a successful connection is made to the Adafruit IO service and it serves as a good point to subscribe to feeds or do other initialization.

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604371/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. # Define callback functions which will be called when certain events happen.
2. def connected(client):
3. # Connected function will be called when the client is connected to Adafruit IO.
4. # This is a good place to subscribe to feed changes. The client parameter
5. # passed to this function is the Adafruit IO MQTT client so you can make
6. # calls against it easily.
7. print 'Connected to Adafruit IO! Listening for feed changes...'
8. # Subscribe to the three pi-dashboard feeds that will be displayed on the
9. # dashboard. Modify this to subscribe to all the feeds you want to display.
10. client.subscribe('pi-dashboard-slider')
11. client.subscribe('pi-dashboard-humidity')
12. client.subscribe('pi-dashboard-temp')

Notice I'm subscribing to three feeds in the above code:

* **pi-dashboard-slider**: This will be displayed on the **slider** dashboard widget.
* **pi-dashboard-humidity**: This will be displayed on the **humidity** dashboard widget.
* **pi-dashboard-temp**: This will be displayed on the **temp** dashboard widget.

Change the code to subscribe to each of the feeds you're using.

Finally modify the **message** function code to look for your feeds and push their payload to the dashboard:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604373/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. def message(client, feed\_id, payload):
2. # Message function will be called when a subscribed feed has a new value.
3. # The feed\_id parameter identifies the feed, and the payload parameter has
4. # the new value.
5. print('Feed {0} received new value: {1}'.format(feed\_id, payload))
6. # Update physical dashboard depending on the changed feed.
7. # Notice the feed\_id is checked to find out which feed changed, then the
8. # appropriate physical dashboard widget is changed.
9. if feed\_id == 'pi-dashboard-slider':
10. # The requests.post function will make an HTTP request against the
11. # dashboard. See the requests documentation for more information:
12. # http://docs.python-requests.org/en/latest/
13. requests.post('{0}/widgets/slider'.format(DASHBOARD\_URL), data={'value': payload})
14. elif feed\_id == 'pi-dashboard-humidity':
15. requests.post('{0}/widgets/humidity'.format(DASHBOARD\_URL), data={'value': payload})
16. elif feed\_id == 'pi-dashboard-temp':
17. requests.post('{0}/widgets/temp'.format(DASHBOARD\_URL), data={'value': payload})

Notice the message function checks the **feed\_id** parameter for each of the feeds that were subscribed to earlier.  If one of those feeds changes then the [requests library](http://docs.python-requests.org/en/latest/) is used to make a HTTP POST request that updates the appropriate dashboard widget.  The **payload** parameter will hold the updated value of the feed and is used by the requests.post function to send the updated feed value to the dashboard.

For example in the code above changes to the **pi-dashboard-humidity** feed will update the dashboard widget called **humidity** (which is exposed at the <http://raspberrypi:5000/widgets/humidity> URL with the dashboard).

You'll want to modify the if/elif statements to check for your feeds and push the value to the appropriate dashboard widget.

That's all you need to change to use this script!  Make sure your physical dashboard is running on the Pi, then connect to the Pi and run the script:

 Download: [file](https://learn.adafruit.com/pages/6847/elements/1604375/download)

[Copy Code](https://learn.adafruit.com/raspberry-pi-physical-dashboard/adafruit-io)

1. python aio\_dashboard.py

The script should print out a message when it's connected to Adafruit IO, and when it receives a subscribed feed change.  Build a dashboard on Adafruit IO or use another script to update your subscribed feeds and you should see those feeds appear on your physical dashboard widgets--cool!

If you see an error that the script can't connect to Adafruit IO, make sure you have the AIO key and username correctly set for your account.

If you don't see your feed updated on the physical dashboard when it changes on Adafruit IO check a few things:

* Make sure the script is successfully connecting to Adafruit IO (i.e. your username and key are correct).
* Make sure you're subscribing to the feeds that you're changing.
* Make sure the if/elif checks in your message function are looking for the feeds you've subscribed to and are changing.
* Check the DASHBOARD\_URL points to the URL of the Pi running your dashboard (or the default localhost:5000 if running the script on the same Pi as the dashboard).

[SOFTWARE](https://learn.adafruit.com/raspberry-pi-physical-dashboard/software)

This guide was first published on Jan 16, 2016. It was last updated on Jan 16, 2016.This page (Adafruit IO) was last updated on Dec 27, 2019.