

RASPBERRY PI PERFORMANCE MONITORING ON THINGSPEAK CLOUD

Many a times we would need to monitor the performance of any server or device that we are working on. With the dawn of 'Internet of Things', things like such have become attainable. In this article one such application i.e. uploading Raspberry pi performance parameters onto ThingSpeak IoT platform is illustrated. This system sends Raspberry pi CPU temperature, memory available and disk usage to ThingSpeak platform and can be reviewed from anyplace via internet.

THINGS REQUIRED:

1. Raspberry pi (I used Raspbian buster OS)
2. Wi-Fi or LAN connectivity

WHAT IS THINGSPEAK?

ThingSpeak is a platform that provides services and facilitates in building IoT applications. It helps with real-time data collection from sensors, data visualization, plugins and apps for collaborating with web services, social network and many other APIs.

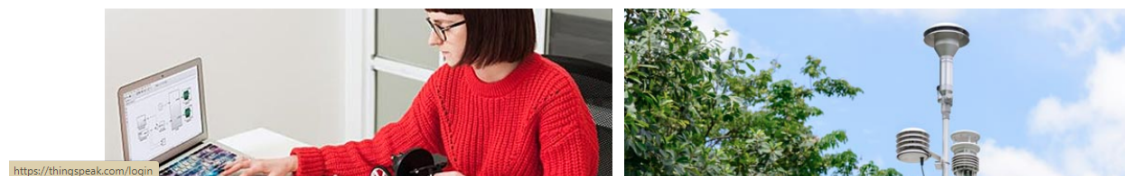
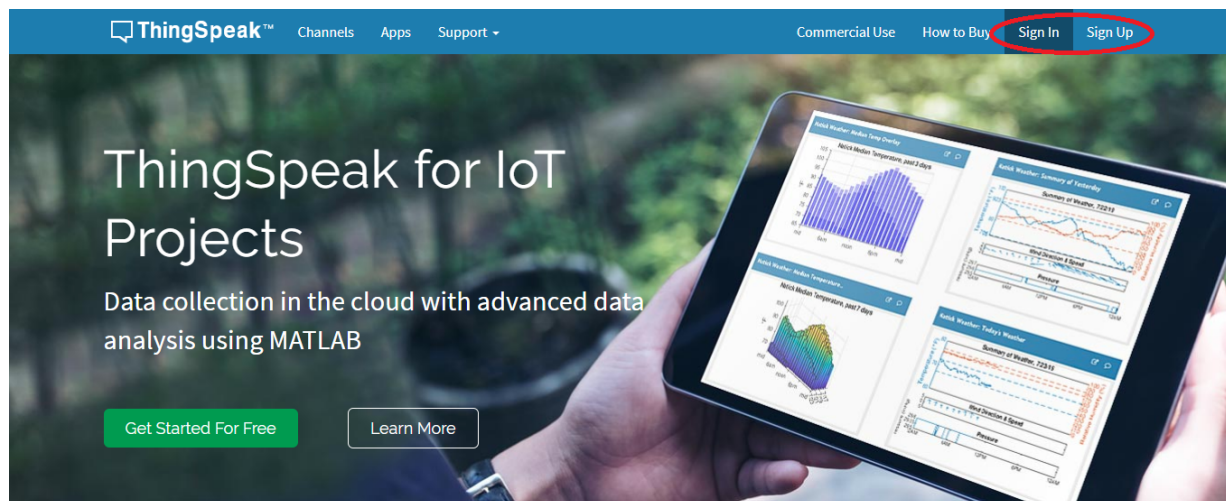
It consists of a channel that stores the data we desire to send to ThingSpeak. It has several attributes like: eight fields for collecting data from a sensor or any other device, an attribute named 'Metadata' that can store JSON, XML and CSV data. Links can also be provided to any external sites, YouTube videos or github that have information about the channel. There are three location fields that store the latitude, longitude and the elevation of the device being interfaced.

The dashboard shows all existing channels with channel name and useful options associated with it. Channel's private and public view can be monitored, channel settings can be modified from dashboard itself also API key and data sharing options are at hand.

RASPBERRY PI PERFORMANCE MONITORING:

RASPBERRY PI PERFORMANCE MONITORING SYSTEM

1. First off, sign up with ' <https://thingspeak.com> '. You can also sign in with your Mathworks account.



2. Head to New Channel to create a new channel.

Signed in successfully.

My Channels

[New Channel](#)

Name	Created	Updated
<div> <div></div> <div>Pi_Performance</div> </div> <div> Private Public Settings Sharing API Keys Data Import / Export </div>	2020-01-11	2020-01-13 06:13

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to [create channels](#), explore and transform data.

Learn more about [ThingSpeak Channels](#).

Examples

- [Arduino](#)
- [Arduino MKR1000](#)
- [ESP8266](#)
- [Raspberry Pi](#)
- [Netduino Plus](#)

Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

[Upgrade](#)

3. Fillout the fields, giving channel a name, description and the parameters to be monitored.

My channel's name is 'Pi_performance' and it monitors 3 parameters: CPU temperature, available memory and disk usage.

Channels
Apps
Support

Commercial Use
How to Buy
Account
Sign Out

Pi_Performance

Channel ID: 957656 | Monitors Raspberry pi performance.

Author: mwa0000017048253
Access: Private

Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export

Channel Settings

Percentage complete: 50%

Channel ID: 957656

Name: Pi_Performance

Description: Monitors Raspberry pi performance.

Field 1: CPU_temperature ☒

Field 2: Memory Usage ☒

Field 3: Disk Usage(%) ☒

Field 4: ☐

Field 5: ☐

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- Percentage complete**: Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- Channel Name**: Enter a unique name for the ThingSpeak channel.
- Description**: Enter a description of the ThingSpeak channel.
- Field#**: Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- Metadata**: Enter information about channel data, including JSON, XML, or CSV data.
- Tags**: Enter keywords that identify the channel. Separate tags with commas.
- Link to External Site**: If you have a website that contains information about your ThingSpeak channel, specify the URL.

4. Copy API key which has to get into the python code.

Channels
Apps
Support

Commercial Use
How to Buy
Account
Sign Out

Pi_Performance

Channel ID: 957656 | Monitors Raspberry pi performance.

Author: mwa0000017048253
Access: Private

Private View
Public View
Channel Settings
Sharing
API Keys
Data Import / Export

Write API Key

Key: **UL0FQPLP4DLT6XIM**

[Generate New Write API Key](#)

Read API Keys

Key: 0NN8Q4TJD0PBG8G1

Note:

[Save Note](#) [Delete API Key](#)

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key**: Use this key to write data to a channel. If you feel your key has been compromised, click [Generate New Write API Key](#).
- Read API Keys**: Use this key to allow other people to view your private channel feeds and charts. Click [Generate New Read API Key](#) to generate an additional read key for the channel.
- Note**: Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

[Write a Channel Feed](#)

```
GET https://api.thingspeak.com/update?api_key=UL0FQPLP4DLT6XIM&field1=
```

[Read a Channel Feed](#)

5. Python code:

This system is coded in Python 3 platform on Raspberry pi model B. Assuming you have already installed 'pip' in Python 3, let's dive in. The modules used come pre-installed with Python 3.

'http.client' defines classes which implement the client side of the HTTP and HTTPS protocols. It is normally used with the module 'urllib.request' to handle URLs that use HTTP and HTTPS.

'psutil'(python system and process utilities) is basically a cross-platform library that helps retrieving information on running

processes and system utilization (CPU, memory, disks, network, sensors) in Python. It will facilitate with the parameters we select to monitor pi's performance.

Code:

```
import http.client
import urllib
import time
import psutil

key = "YOUR API KEY FROM THINGSPEAK" # Put your API Key here
def perf_monitor():
    while True:
        temp = int(open('/sys/class/thermal/thermal_zone0/temp').read()) / 1e3 # Get Raspberry
        # This is how we get pi CPU temperature.
        mem = psutil.virtual_memory()
        mem_available = mem.available # Get available CPU memory
        # psutil.virtual_memory() returns metrics such as total memory, available memory, memory in use, buffers,etc. This code
        # extracts available CPU memory.
        x = psutil.disk_usage('/')
        disk_us = x.used # Get used disk space
        # psutil.disk_usage(path) returns metrics about the partition which contains the given path. It returns total, used space in
        # bytes and also the percentage usage. This code extracts used disk space in bytes.
        params = urllib.parse.urlencode({'field1':
temp, 'field2':mem_available, 'field3':disk_us, 'key':key })
        # Returns 'field1=temp&field2=mem_available&field3=disk_us&key=YOUR API KEY FROM THINGSPEAK'
        headers = {"Content-type": "application/x-www-form-urlencoded", "Accept": "text/plain"}
        conn = http.client.HTTPConnection("api.thingspeak.com:80")
        # This line represents one transaction with an HTTP server with 80 being the default port number.
        try:
            conn.request("POST", "/update", params, headers)
            response = conn.getresponse()
            print (temp)
            print(mem_available)
            print(disk_us)
            print (response.status, response.reason)
            data = response.read()
            conn.close()
        except:
            print( "connection failed")
            break
        # system sleeps for 15 seconds here
if __name__ == "__main__":
    while True:
        perf_monitor()
        time.sleep(15)
```

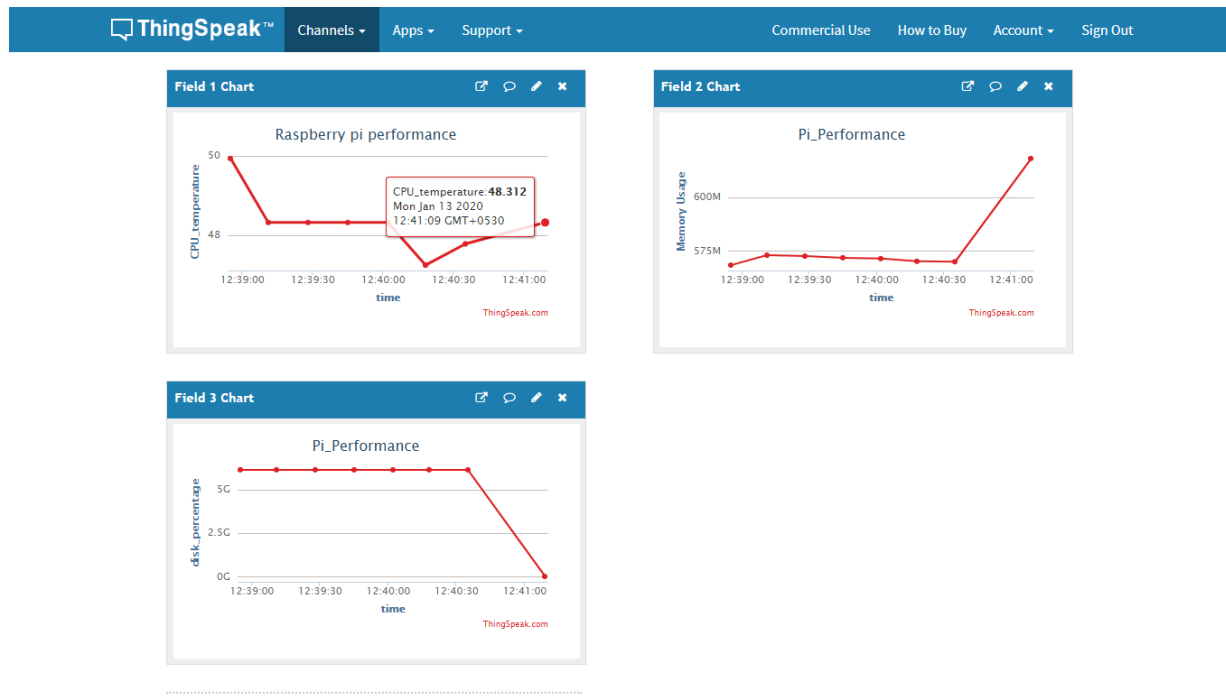
You can either run this code directly in IDLE by clicking on 'Run' or in terminal using the command:

```
sudo python3 filename.py
```

Output as viewed on terminal. It displays CPU temperature, available memory and disk used.

```
pi@raspberrypi: ~
File Edit Tabs Help
6141386752
200 OK
48.312
572973056
6141411328
200 OK
48.312
572542976
6141435904
200 OK
48.312
571785216
6141493248
200 OK
48.312
571375616
6141509632
200 OK
47.236
570122240
6141534208
200 OK
47.774
569929728
6141558784
200 OK
^Z
[3]+ Stopped sudo python3 iotpi.py
pi@raspberrypi:~ $ scrot
```

6. The output can be monitored from anywhere using 'ThingSpeak platform'. The figure below shows the uploaded data.



7. The subsequent link shows my work and the above picture.

<https://thingspeak.com/channels/957656>

FURTHER POSSIBILITIES:

- This system can be interfaced with an LCD or OLED to locally display the performance parameters.
- Data can be visualized in many other ways using MATLAB.
- Data can be exported for additional analysis or processing.