Uploading Sensor Data to Cloud

Description:

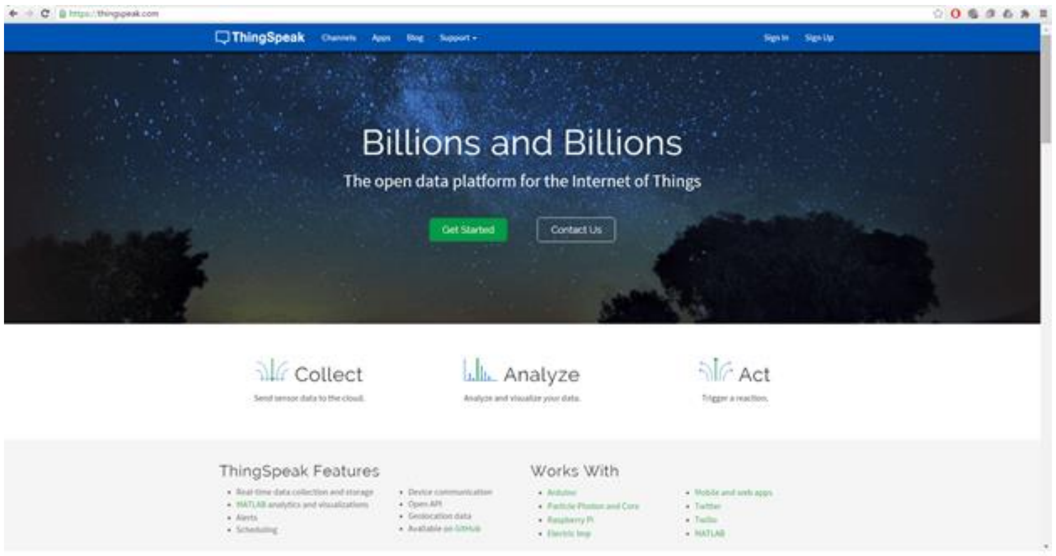
This project aims at making your NodeMCU as a gateway device for uploading your sensor data on ThingSpeak server. This tutorial allows the user to measure the surrounding temperature and humidity, measured values are sent to a personal cloud server in Thingspeak for every 15 seconds , so you will be able to see both visualization in a web page.

In order to send data to ThingSpeak using a Node MCU, you need an NodeMCU with network connectivity. ThingSpeak requires a user account and a channel. A channel is where you can send data and ThingSpeak stores data.

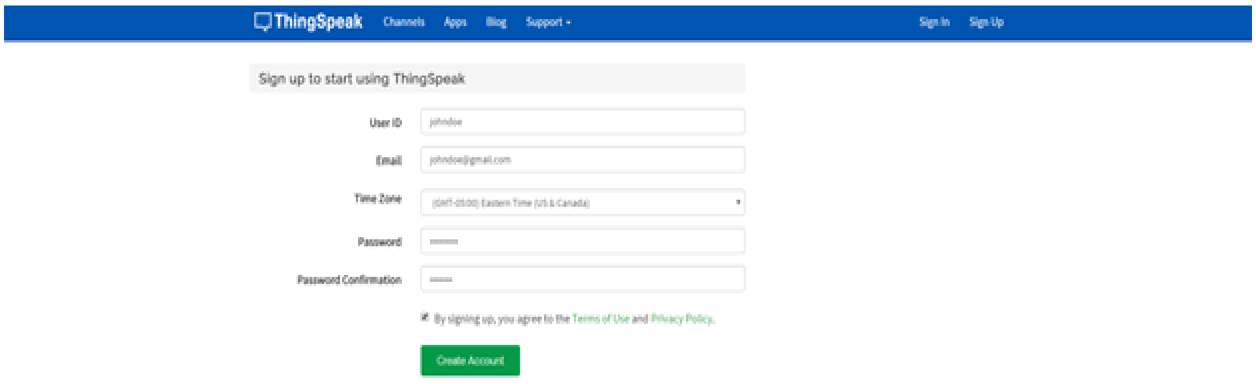
Step wise procedure:

Step 1: Signup for Thingspeak

Go to **www.thinspeak.com**



Click on "Sign Up" option and complete the details



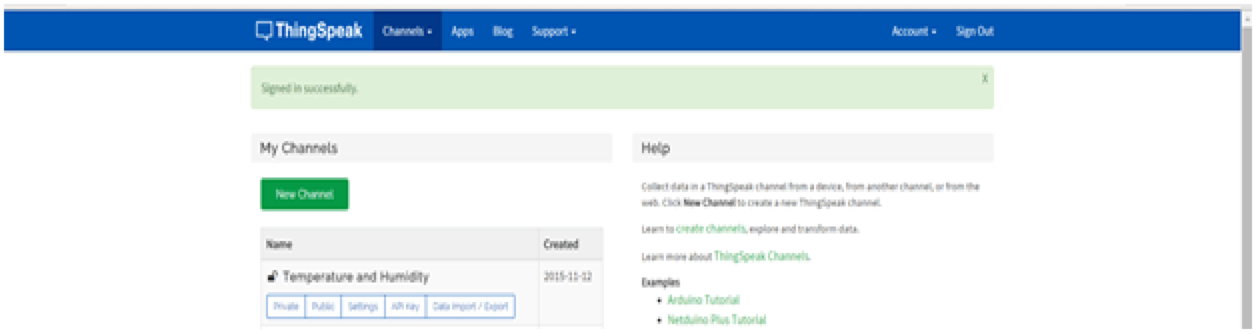
Sign up to start using ThingSpeak

User ID	johndoe
Email	johndoe@gmail.com
Time Zone	(GMT-05:00) Eastern Time (US & Canada)
Password	•••••••
Password Confirmation	••••••

✓ By signing up, you agree to the Terms of Use and Privacy Policy.

Step 2: Create a Channel for Your Data

Once you Sign in after your account activation, Create a new channel by clicking "New Channel" button



Signed in successfully.

My Channels

Name	Created
Temperature and Humidity	2015-11-12

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.

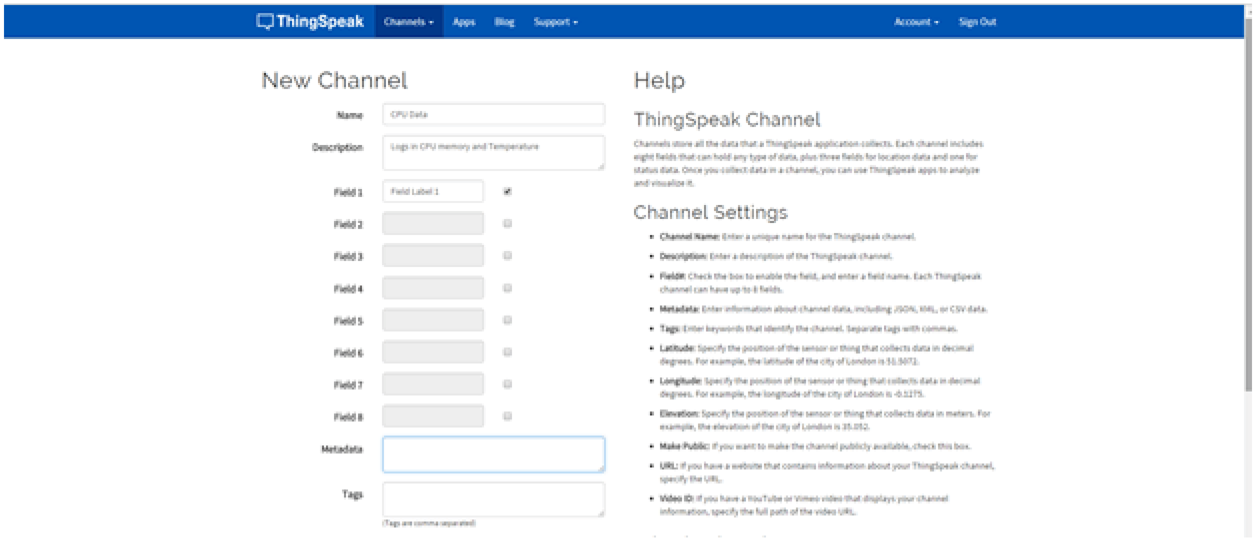
Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples
- Arduino Tutorial
- Netduino Plus Tutorial

After the "New Channel" page loads, enter the Name and Description of the data you want to upload

You can enter the name of your data (ex: Temperature) in Field1. If you want more Fields you can check the box next to Field option and enter the corresponding name of your data.



New Channel

Name: CPU Data

Description: Logs in CPU memory and Temperature

Field 1: Field Label 1 ✓

Field 2

Field 3

Field 4

Field 5

Field 6

Field 7

Field 8

Metadata

Tags

(Tags are comma separated)

Help

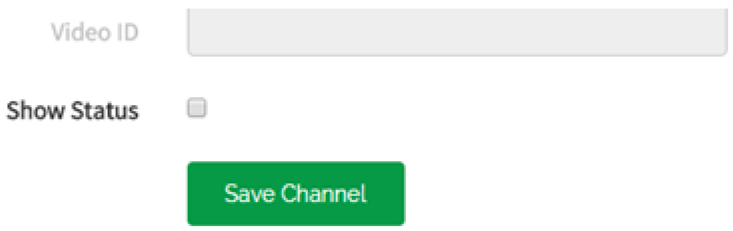
ThingSpeak Channel

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.

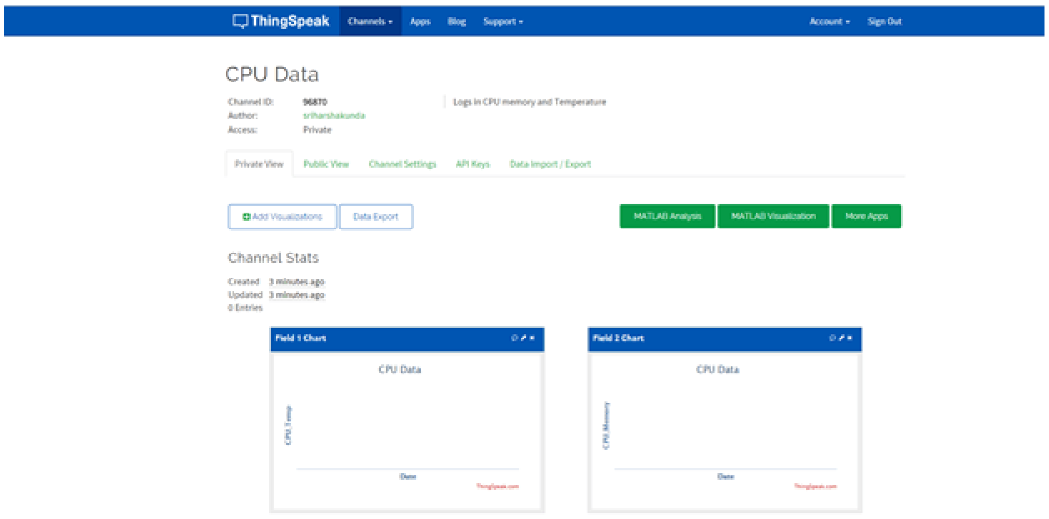
Channel Settings

- **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.
- **Latitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the latitude of the city of London is 51.5072.
- **Longitude:** Specify the position of the sensor or thing that collects data in decimal degrees. For example, the longitude of the city of London is -0.1275.
- **Elevation:** Specify the position of the sensor or thing that collects data in meters. For example, the elevation of the city of London is 35.052.
- **Make Public:** If you want to make the channel publicly available, check this box.
- **URL:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Video ID:** If you have a YouTube or Vimeo video that displays your channel information, specify the full path of the video URL.

Click on "Save Channel" button to save all of your settings.



I created two Fields; one is temperature and one for humidity



CPU Data

Channel ID: 96870

Author: sriharshakunda

Access: Private

Logs in CPU memory and Temperature

Channel Stats

Created 3 minutes ago

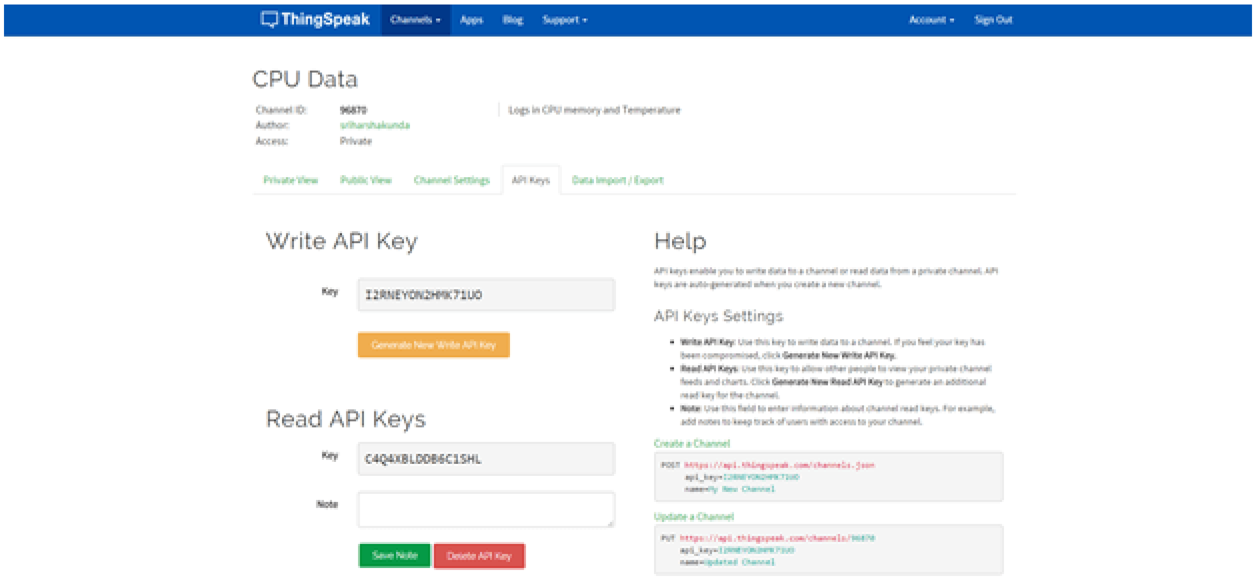
Updated 3 minutes ago

0 Entries

Step 3: Get an API Key

To upload our data, we need an API key, which we will later include in a piece of Arduino code to upload our sensor data to Thingspeak Website.

Click on "API Keys" tab to get the key for uploading your sensor data.



CPU Data

Channel ID: 96870
Author: sriharshakunda
Access: Private

Logs in CPU memory and Temperature

Write API Key

Key I2RNEYON2HMK71UO

Read API Keys

Key C4Q4XBLDDB6C1SHL

Note

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.

Create a Channel

```
POST https://api.thingspeak.com/channels.json
     api_key=I2RNEYON2HMK71UO
     name=My New Channel
```

Update a Channel

```
PUT https://api.thingspeak.com/channels/96870
    api_key=I2RNEYON2HMK71UO
    name=Updated Channel
```

(Note: The advantage of using Thingspeak compared to Xively or any other websites is that the convenience of using Matlab Analysis and Matlab Visualizations. This is a good option especially if you are doing some kind of research projects)

Once you have the "Write API Key". We are almost ready to upload our data, except for the python code.

Step 4: Modifying the Arduino sketch

The ThingSpeak Client sketch is designed for the Node MCU. This sketch updates a channel feed via the ThingSpeak API (http://community.thingspeak.com/documentation/) using HTTP

GET. The Arduino uses DHCP and DNS for a simpler network setup. The sketch also includes a Watchdog / Reset function to make sure the Node MCU stays connected and/or regains connectivity after a network outage. Use the Serial Monitor on the Arduino IDE to see feedback and ThingSpeak connectivity status. Here we upload Temperature data.

Use your Write API Key to replace the **key** with your **API Key**

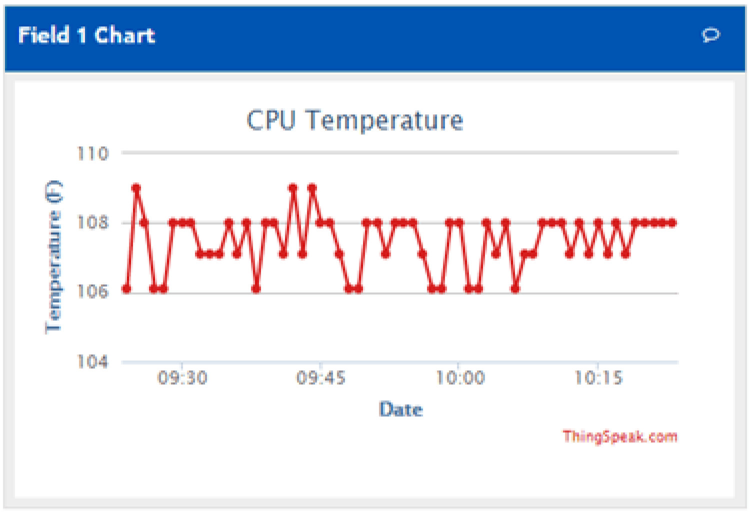
Save the file to overwrite changes

Step 5: open serial monitor and check for updation

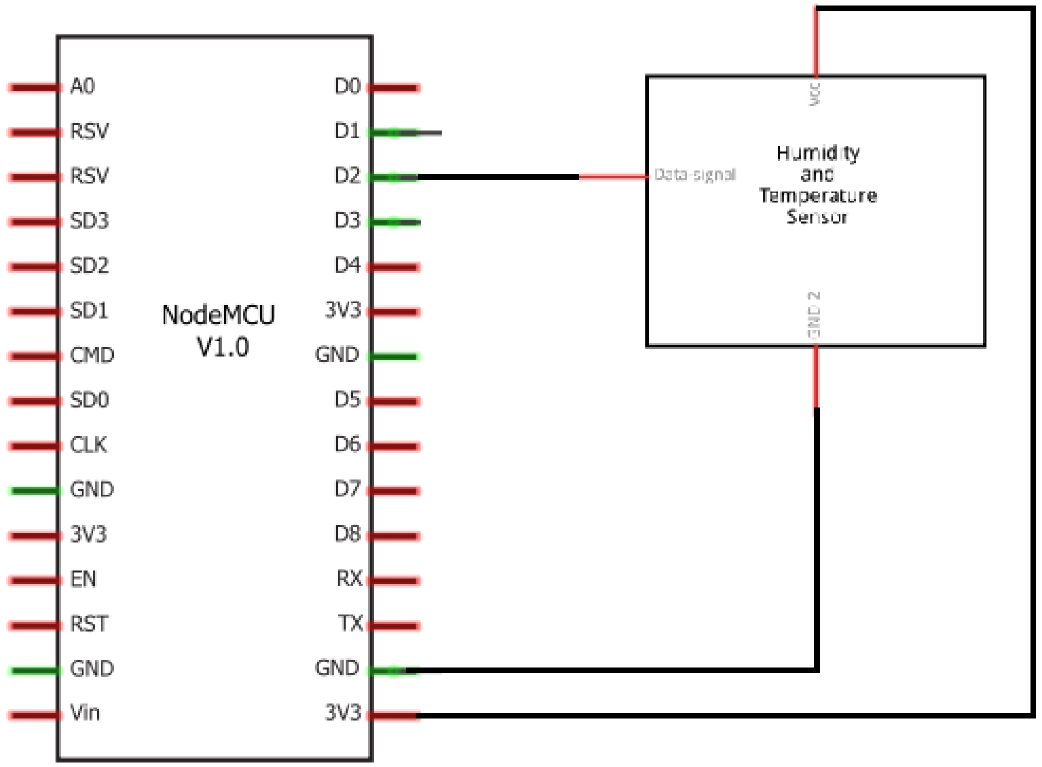
In case if there are any errors uploading the data, you will receive “connection failed” message

Step 6: Check Thinspeak API and Confirm data transfer

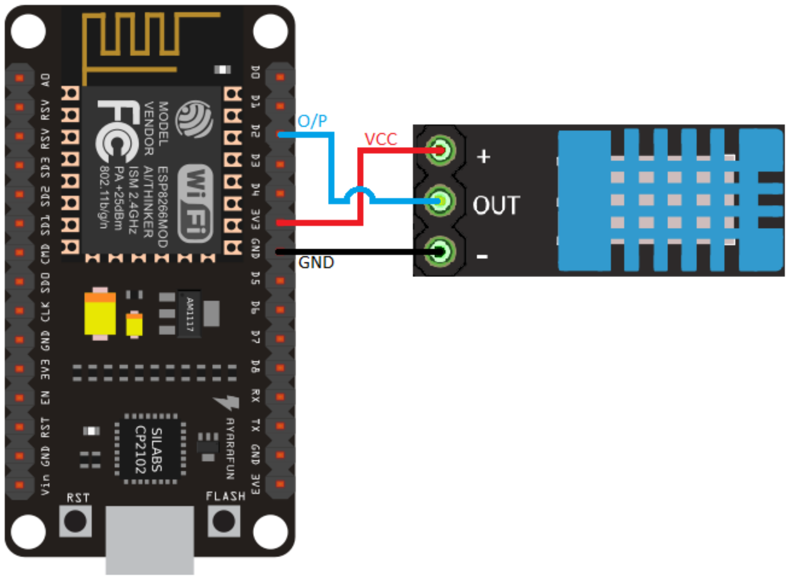
Open your channel and you should see the temperature uploading into thinspeak website.



Circuit Diagram:



Connection Diagram:



Components Required:

- NodeMCU
- Humidity Sensor (DHT11)

Library Required:

#include<ESP8266WiFi.h>

#include<DHT.h>

Code:

```
#include <ESP8266WiFi.h>

#include <DHT.h>

#define DHTPIN D2

#define DHTTYPE DHT11

DHT dht(DHTPIN, DHTTYPE);

const char* ssid     = "SSID";

const char* password = " password ";

const char* host = "api.thingspeak.com";

const char* privateKey = "5VK8LDE9EN1D5O4A";

void setup() {

Serial.begin(115200);

delay(10);

// We start by connecting to a WiFi network

Serial.println();

Serial.println();

Serial.print("Connecting to ");

Serial.println(ssid);

WiFi.begin(ssid, password);

while (WiFi.status() != WL_CONNECTED) {
```

```
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
float value, temp;
void loop() {
delay(15000);
float h = dht.readHumidity();
float t = dht.readTemperature();
Serial.print("connecting to ");
Serial.println(host);

// Use WiFiClient class to create TCP connections
WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
Serial.println("connection failed");
return;
}
// We now create a URL for the request
String url = "/update";
url += "?api_key=";
url += privateKey;
url += "&field1=";
url += t;
url += "&field2=";
url += h;
Serial.print("Requesting URL: ");
Serial.println(url);
// This will send the request to the server
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
"Host: " + host + "\r\n" +
"Connection: close\r\n\r\n");
delay(10);

// Read all the lines of the reply from server and print them to    Serial
while(client.available()){
String line = client.readStringUntil('\r');
Serial.print(line);
}
Serial.println();
Serial.println("closing connection");
}
```