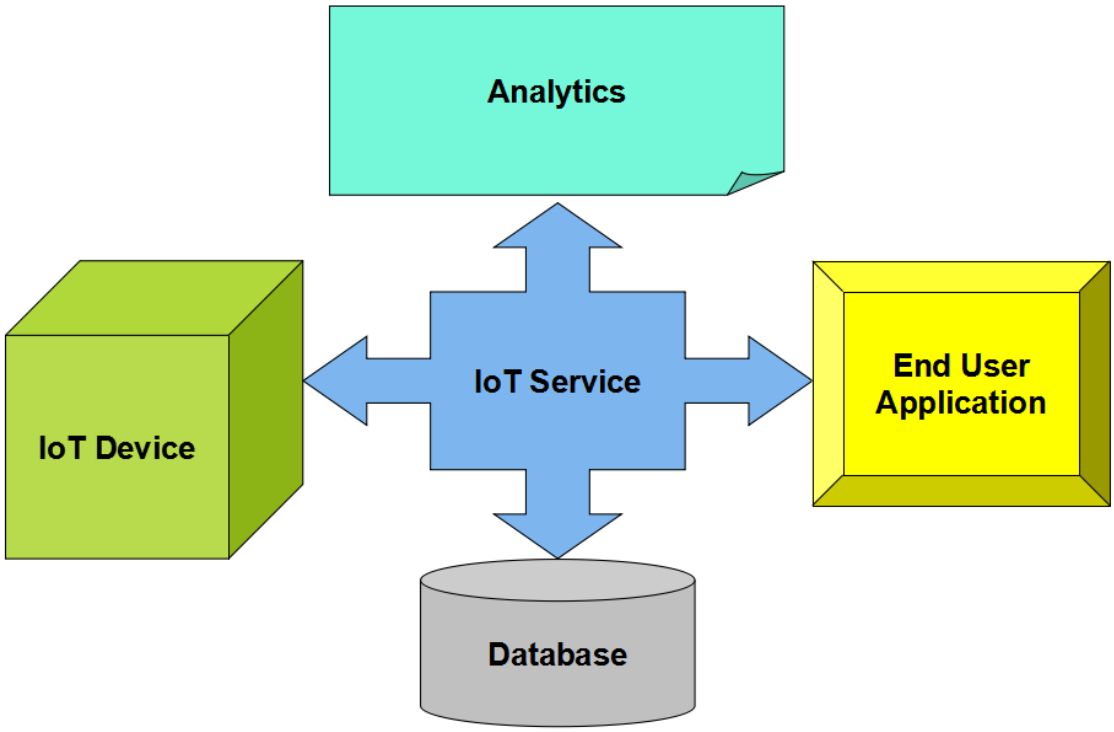
Introduction

The Internet of Things(IoT) is a system of 'connected things'. The things generally comprise of an embedded operating system and an ability to communicate with the internet or with the neighbouring things. One of the key elements of a generic IoT system that bridges the various 'things' is an IoT service. An interesting implication from the 'things' comprising the IoT systems is that the things by themselves cannot do anything. At a bare minimum, they should have an ability to connect to other 'things'. But the real power of IoT is harnessed when the things connect to a 'service' either directly or via other 'things'. In such systems, the service plays the role of an invisible manager by providing capabilities ranging from simple data collection and monitoring to complex data analytics. The below diagram illustrates where an IoT service fits in an IoT ecosystem:



One such IoT application platform that offers a wide variety of analysis, monitoring and counter-action capabilities is 'ThingSpeak'. Let us consider ThingSpeak in detail.

What is ThingSpeak

ThingSpeak is a platform providing various services exclusively targeted for building IoT applications. It offers the capabilities of real-time data collection, visualizing the collected data in

the form of charts, ability to create plugins and apps for collaborating with web services, social network and other APIs. We will consider each of these features in detail below.

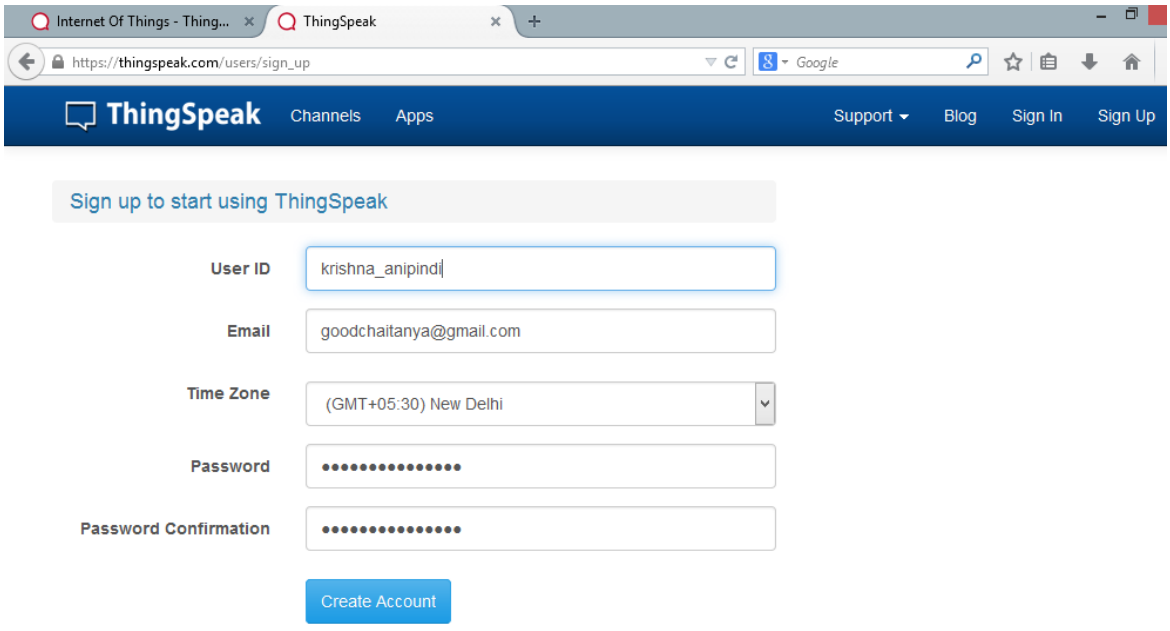
The core element of ThingSpeak is a 'ThingSpeak Channel'. A channel stores the data that we send to ThingSpeak and comprises of the below elements:

- 8 fields for storing data of any type - These can be used to store the data from a sensor or from an embedded device.
- 3 location fields - Can be used to store the latitude, longitude and the elevation. These are very useful for tracking a moving device.
- 1 status field - A short message to describe the data stored in the channel.

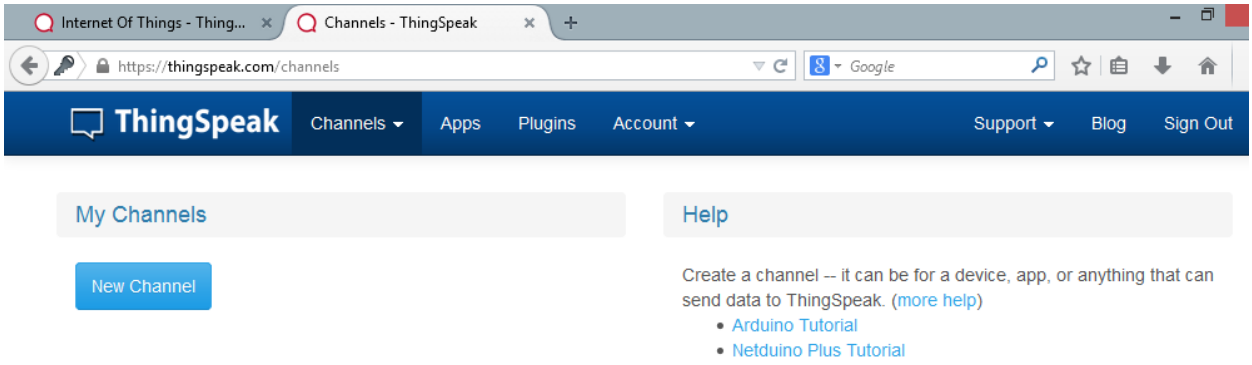
To use ThingSpeak, we need to signup and create a channel. Once we have a channel, we can send the data, allow ThingSpeak to process it and also retrieve the same. Let us start exploring ThingSpeak by signing up and setting up a channel.

Getting Started

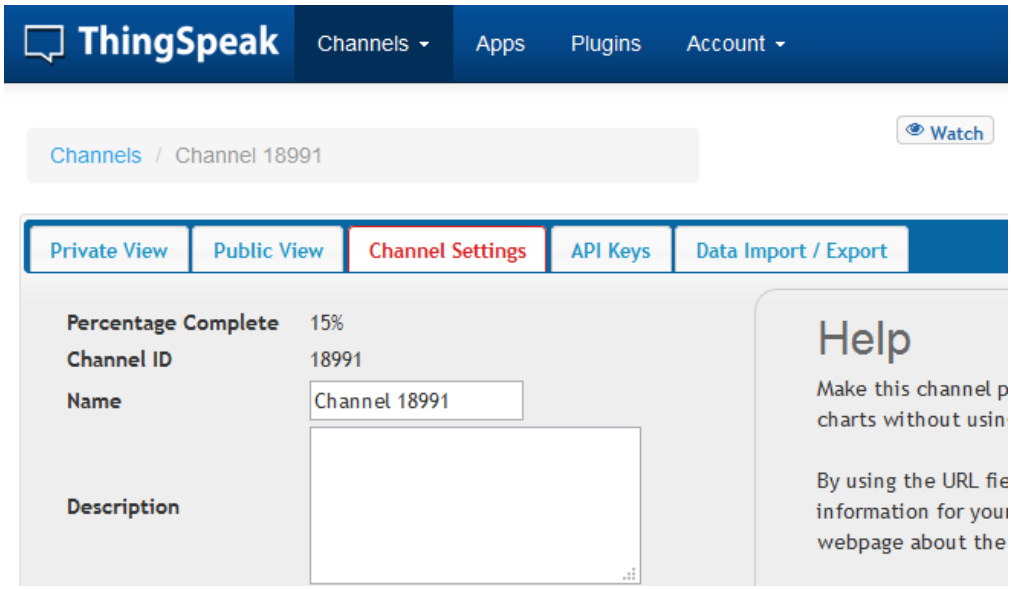
Open https://thingspeak.com/ and click on the 'Get Started Now' button on the center of the page and you will be redirected to the sign-up page(you will reach the same page when you click the 'Sign Up' button on the extreme right). Fill out the required details and click on the 'Create Account' button.



Now you should see a page with a confirmation that the account was successfully created. The confirmation message disappears after a few seconds and the final page should look as in the below screen:



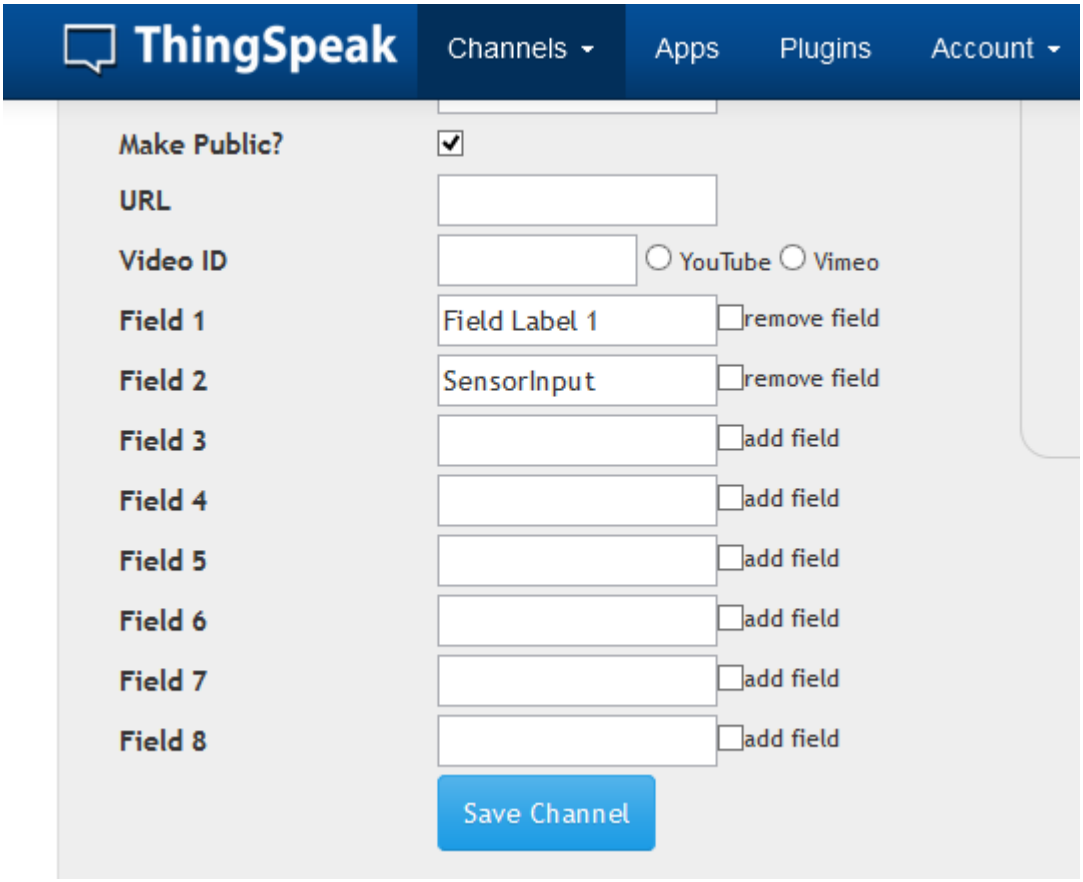
Go ahead and click on 'New Channel'. You should see a page like the below:



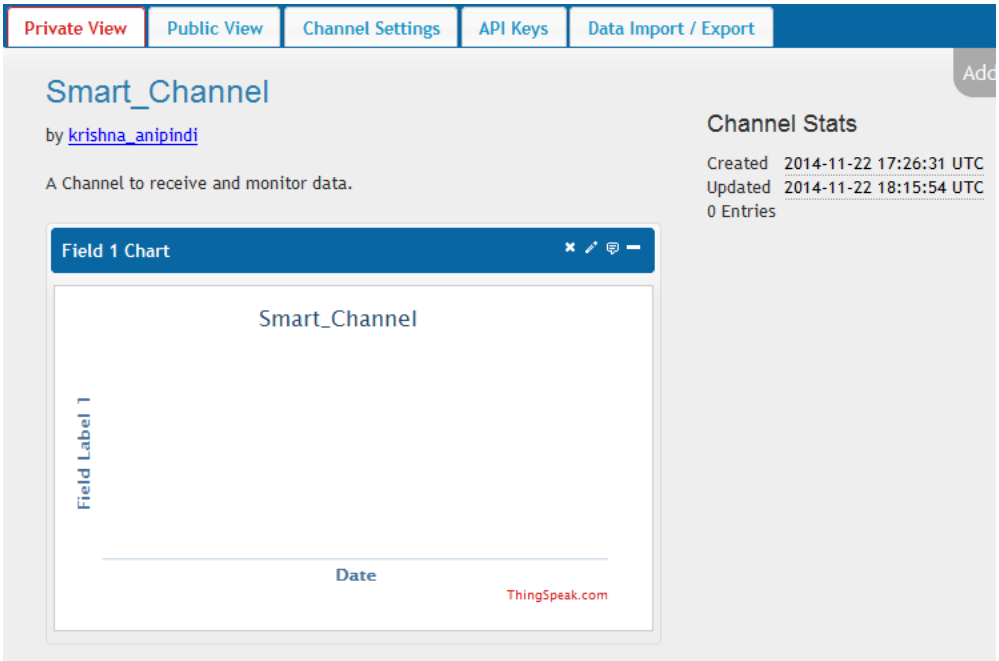
You can change the name to fit your need and you can add a description corresponding to the channel. You can add any other useful description into the metadata field. In the same page, you should see the fields for Latitude, Longitude and Elevation. Also, when you scroll down you should see a check box that says 'Make Public?'. Let us consider the significance of the various fields and the tabs:

- Latitude, longitude and elevation - These fields correspond to the location of a 'thing' and are especially significant for moving things.
- Make Public? - If the channel is made public, anyone can view the channel's data feed and the corresponding charts. If this check box is not checked, the channel is private, which means for every read or write operation, the user has to pass a corresponding API key.
- URL - This can be the URL of your blog or website and if specified, will appear on the public view of the channel.
- Video ID - This is the ID corresponding to your YouTube or Vimeo ID. If specified, the video appears on the public view of the channel.
- Fields 1 to 8 - These are the fields which correspond to the data sent by a sensor or a 'thing'. A field has to be added before it can be used to store data. By default, Field 1 is added. In case you try posting to fields that you have not added, your request will still be successful, but you will not be able to see the field in the charts and the corresponding data. You can click on the small box before the 'add field' text corresponding to each field to add it. Once you click the 'add field' box,

a default label name appears in the text box corresponding to each field and the ‘add field’ text changes to ‘remove field’. You can edit the field text that appears by default when a field is added to make more sense. For example, in the below screen, I have modified the text for Field 2 to ‘SensorInput’. To remove a field which is added, just check on the ‘remove field’ box. Once you click this, the text ‘remove field’ changes back to ‘add field’ and the corresponding field text is cleared.



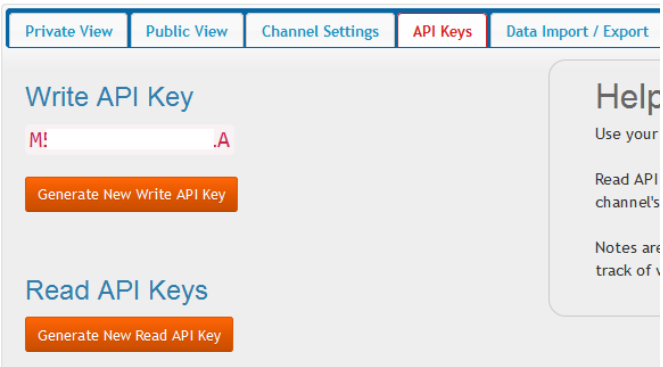
Once you have edited the fields, click on ‘Save Channel’ button. You should now see a page like the below in which the ‘Private View’ tab is defaulted:



The Private View shows a chart corresponding to each of the fields that we have added. Now click on the 'Public View' tab. This should look exactly similar to the what we see in the 'Private View' tab since our channel is public. In case your channel is not public('make public' check box not checked in the 'channel settings' tab), the public view tab shows a message that 'This channel is not public'.

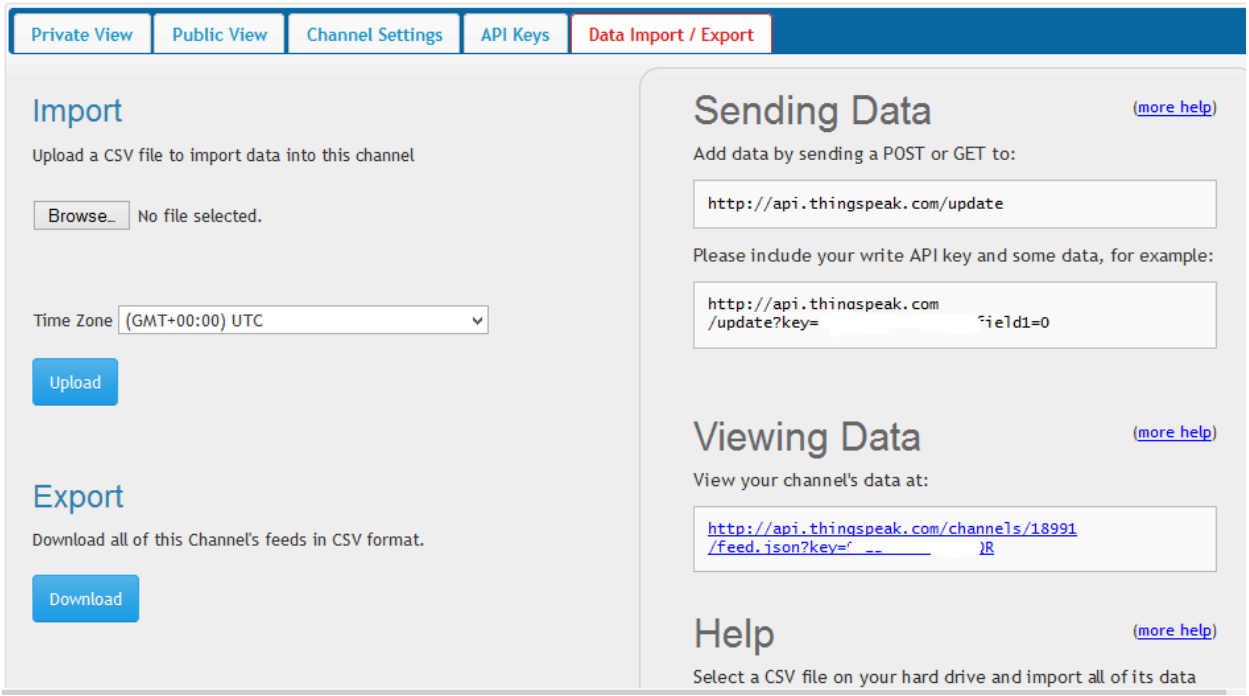
Now click on the 'API Keys' tab. You should see a screen similar to the below. The write API key is used for sending data to the channel and the read API key(s) is used to read the channel data. When we create a channel, by default, a write API key is generated. We generate read API keys by clicking the 'Generate New Read API Key' button under this tab.You can also add a note corresponding to each of the read API keys.

Note: Please note that clicking on the 'Generate New Write API Key' will over-write the previous key. You will only have one Write API key at any point of time. Also, in case your channel is private, others can only view the channel's feed and charts by using a Read API key. Please share the Read API keys with people who are approved and authorized to view your channel.



Now click on the 'Data Import/Export' tab and you should see a screen similar to the below. This tab is used to import the 'Comma Separated Values(CSV)' data from a file into the channel. You

can also download the channel's feed from here in CSV format. This tab also outlines how to send and view data by providing the URIs to the send and view APIs.



Updating Status Fields using C# .NET

Now that we have come this far, let us get into some action. Let us try updating the fields in our channel using C# and .NET. I have used Visual Studio 2013 Express for Web, but the code relatively works the same with other versions too.

1. Open Visual Studio and Create a new project. In case you are prompted for a template, choose an empty web application.
2. In the solution explorer, right click on the project, Add --> New Item and choose Web Form. Once the form gets added, add the below code inside the form tag:

Hide Copy Code

```
<label id="lblError"  runat="server"  style="display:none;font-
weight:bold;color:red"></label>
```

This label is used to display an error message in case we run into any exceptions.

3. Go to the code behind and add the below code in the page load:

Hide Shrink Copy Code

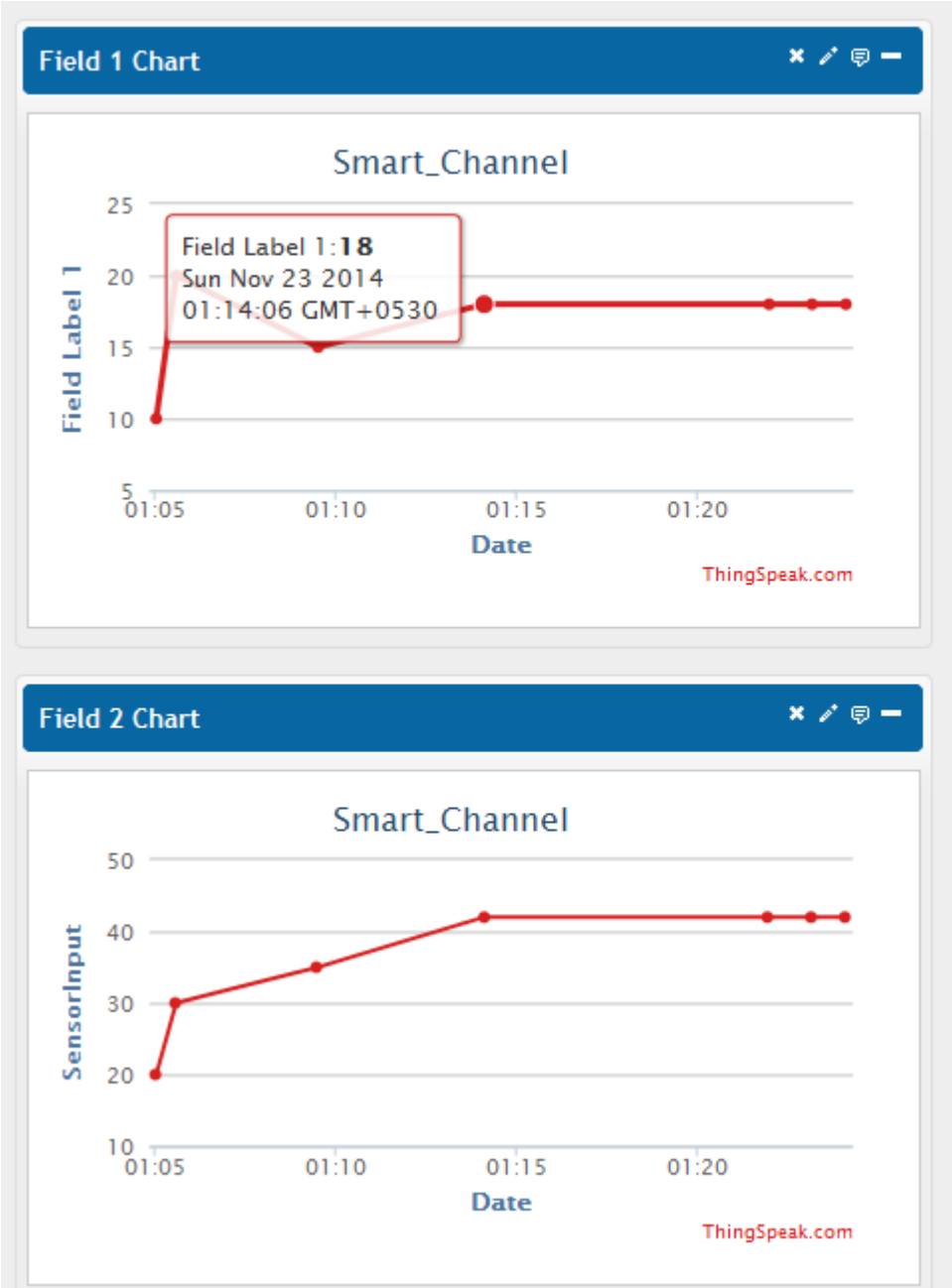
```
try
{

    const string WRITEKEY = "YOUR_KEY";
    string strUpdateBase = "http://api.thingspeak.com/update";
    string strUpdateURI = strUpdateBase + "?key=" + WRITEKEY;
    string strField1 = "18";
    string strField2 = "42";
    HttpWebRequest ThingsSpeakReq;
```

```
    HttpWebResponse ThingsSpeakResp;

    strUpdateURI+= "&field1=" + strField1;
    strUpdateURI+= "&field2=" + strField2;

    ThingsSpeakReq = (HttpWebRequest)WebRequest.Create(strUpdateURI);

    ThingsSpeakResp = (HttpWebResponse)ThingsSpeakReq.GetResponse();

    if(!(string.Equals(ThingsSpeakResp.StatusDescription,"OK")))
    {
        Exception exData = new Exception(ThingsSpeakResp.StatusDescription);
        throw exData;
    }
}
catch (Exception ex)
{
    lblError.InnerText = ex.Message;
    lblError.Style.Add("display", "block");
    throw;
}
```

Exploring the code: I have started with building the URI for updating the fields. The initial URI ishttp://api.thingspeak.com/update to which the key has to be appended. Please note that you need to replace the string YOUR_KEY with the actual write API key which you will get once you sign up and create a channel.

After that I have declared two constants which I will be pushing to my ThingSpeak channel. After that I am appending these fields to the URL under the parameters field1 and field2. You can push the other fields until field8 similarly. After this, I am creating a HTTPWebRequest object corresponding to this URI by using the 'WebRequest.Create' method. After this, I am retrieving the response using the GetResponse method and assigning the same to the response object. In case the data was successfully sent to the channel, the resulting StatusDescription will be 'OK'. I am checking for any non-OK status descriptions and just in case of an error, I am throwing an exception and displaying the resultant message in a label.

After a series of updates, the charts in the private view tab for each of the fields will look like the below:



Each of the dots correspond to the value and the time at which the value was posted to the channel. Place the mouse over a dot to get more details on the exact date and the GMT offset from which the value was posted.

Please note that in the above example, I have sent some sample values to the channel. You can send any data here, say the periodic readings from a temperature sensor or RPM values from a motor. The Y-axis show the names that we specified to each of the labels.

Let us try something else interesting.

Post your current latitude, longitude and altitude to ThingSpeak

Consider the below example where I retrieve my current latitude, longitude and altitude using the HTML5 GeoLocation API and post the same to ThingSpeak using JavaScript. I have a simple HTML page in which I have a function by the name 'fOnLoad' which I am invoking once the page load is complete. Inside this function, I have the below code which invokes the 'getCurrentPosition' method of the HTML5 GeoLocation API:

Hide Copy Code

```
if(navigator.geolocation)
            {
                navigator.geolocation.getCurrentPosition(UpdateDataToThingSpeak);
            }
```

The code 'navigator.geolocation' checks whether the browser supports geolocation. If this condition is met, we are invoking the 'getCurrentPosition' method. Once this method call is successful, the 'UpdateDataToThingsSpeak' method is invoked:

Hide Copy Code

```
function UpdateDataToThingSpeak(GeoData)
        {
            var lat = GeoData.coords.latitude;
            var lon = GeoData.coords.longitude;
            var alt = GeoData.coords.altitude;

            if(alt == null)
            {
                alt = 0;
            }

            var HttpAgent = new XMLHttpRequest();
            var URI = "http://api.thingspeak.com/update?key=YOUR_KEY&lat=" + lat + "&long="
+ lon + "&elevation=" + alt;

            HttpAgent.open("POST", URI);
            HttpAgent.send();
        }
```

In this method, I retrieved the latitude, longitude and altitude from the GeoData parameter(you can keep any name here in place of GeoData) using 'GeoData.coords'. In some browsers, the altitude is returned as null. Hence I have added a check for this and I am making the altitude as zero in this case. After this, I instantiated an object of type XMLHttpRequest. We will be using this to invoke the ThingSpeak API.

NOTE: The altitude always has to be an integer and hence we cannot pass 'null' or any other string here.

The next step is to build the URI by appending the latitude, longitude and altitude as query string parameters under lat, long and elevation. After this, I am invoking the API using the open and

send methods of the XMLHttpRequest object. I have used Internet Explorer 11 to run this example. After running this, to check whether the feed got updated, run the below URL:

http://api.thingspeak.com/channels/YOUR_CHANNEL_ID/feed.json?key=YOUR_KEY&location=true

NOTE: As mentioned above, you have to replace YOUR_KEY with your write API key and replace YOUR_CHANNEL_ID with the ID corresponding to your channel. Also, if you invoke the above URL without specifying location=true, you will not be able to see the latitude, longitude and altitude fields.

The below is a screen after running the above URL after updating my location details for few times:

ThingSpeak Apps

ThingSpeak provides apps that allow us for an easier integration with the web services, social networks and other APIs. Below are some of the apps provided by ThingSpeak:

- ThingTweet - This allows you to post messages to twitter via ThingSpeak. In essence, this is a TwitterProxy which re-directs your posts to twitter.
- ThingHTTP - This allows you to connect to web services and supports GET, PUT, POST and DELETE methods of HTTP.
- TweetControl - Using this, you can monitor your Twitter feeds for a specific key word and then process the request. Once the specific keyword is found in the twitter feed, you can then use ThingHTTP to connect to a different web service or execute a specific action.
- React - Send a tweet or trigger a ThingHTTP request when the Channel meets a certain condition.

- TalkBack - Use this app to queue up commands and then allow a device to act upon these queued commands.
- Timecontrol - Using this app, we can do a ThingTweet, ThingHTTP or a TalkBack at a specified time in the future. We can also use this to allow these actions to happen at a specified time throughout the week.

  In addition to the above, ThingSpeak allows us to create the ThingSpeak applications as plugins using HTML, CSS and JavaScript which we can embed inside a website or inside our ThingSpeak channel.

Conclusion

One of the key elements of an IoT system is an IoT service. ThingSpeak is one such application platform offering a wide variety of features. At the heart of ThingSpeak is a channel which can be used for storing and processing data collected from the 'things'. ThingSpeak also provides various apps for integration with web services, other APIs and social networks and provides the capability to create the applications as plugins. It is a great platform with extensive possibilities to explore the integration of the Internet of Things.