ThingSpeak – an API and Web Service for the Internet of Things

Marcello A. Gómez Maureira
LIACS, Leiden University
ma.gomezmaureira@gmail.com

Daan Oldenhof
LIACS, Leiden University
daanoldenhof@gmail.com

Livia Teernstra
LIACS, Leiden University
helloliefje@gmail.com

ABSTRACT

In this report we describe the use of *ThingSpeak*, an "Application Programming Interface" (API) and web service for the "Internet of Things" (IoT). While the interpretation as to what should be understood under the term is changing over time, here we refer to enabling objects or simple devices to be identified and communicated with via the Internet. The *ThingSpeak* API is an open source interface which listens to incoming data, timestamps it, and outputs it for both human users (through visual graphs) and machines (through easily parse-able code). We look into practical examples using the *Arduino* micro-controller as well as communication with graphical interface operating systems through a *Python* script. Our report concludes that *ThingSpeak* is especially useful for smaller hardware projects where connectivity over the Internet is required but in which the maintenance of a dedicated communication server is not practical. Alternative IoT services exist but tend to require payment for some of their functionality and are consequently not open source.

PURPOSE, CONTEXT AND HISTORY

The term "Internet of Things" (IoT), coined by Kevin Ashton in 1999 [2], has been in use for several years and continues to be of interest, specifically when it comes to technological progress. But what exactly is the IoT? Essentially, it refers to giving objects representation in the digital realm through giving them a unique ID and connecting them in a network [2]. In other words, these things are connected to the internet and are able to automatically transfer data without relying on human interaction [30] - hence being "Machine to Machine" (M2M) interaction. Essentially, M2M interaction enables networked devices to exchange data and perform actions without the input or assistance of humans, for instance in remote monitoring [31]. Indeed, the lack of necessity for human intervention seems to conjure dystopian images of the future. But this is not necessarily the case.

For instance, one can envision the IoT to become an important feature of the 'home of the future', where one can begin pre-heating the oven just before they get home from work via a (mobile) application. Or perhaps, automatically turning on the washing machine when the power grid has less load, as communicated by a remote power plant. Or, businesses can anticipate when a popular item is running low on stock due to notification from the shelves that they sit on. Hence, the IoT has many interesting applications that can be applied to both individuals and corporations.

According to Dr. Lara Sristava, advisor to the European Commission, the IoT is predicted to become pervasive and as ubiquitous as the Internet today [26]. Increasingly embedded processing power in objects will lead to increased embedded intelligence in objects, consequently giving rise to the popular term of referring to such objects as "smart" [19]. Objects will create their own content and merge with society. So this content will merge with user content, creating a semantic world informed by key patterns. These patterns will help us better leverage our increasingly large amount of available data. So in the absolute most general sense, the IoT can be thought of as creating intelligent devices that are interconnected with people and other devices. The type of device that is connected is only limited by the imagination of its creator.

History

The concept of the IoT, although uncoined at the time, has been in discussion since the early 1990s [20]. The notion was popularized by using it for market analysis at MIT's Auto-ID center. Originally, the network was based on items tagged with "Radio Frequency ID" (RFID) chips, a technology which has already existed for half a decade. The first applications were to use RFID chips in inventory management, from the facilitation of routing to loss prevention.

Over time, technologies developed and other methods of networking objects emerged such as using bar-codes, "Quick Response" (QR) codes, digital watermarking and "Near Field Communication" (NFC) [26]. As costs lowered for the technologies to make objects "smart", an increased amount of applications for networking objects came to be. These applications included connecting objects for surveillance and security. It was also applied in other industrial uses such as transport, food safety and document management [7]. As a relatively 'new' web technology, the timeline for the IoT looks quite sparse:

- 1991: Concept realized
- 1999: Term "Internet of Things" coined
- 2002: Popularization by MIT and use in businesses
- 2005–2010: Applications in surveillance, security, healthcare, transport, food safety and document management
- 2011: *ThingSpeak* API committed to GitHub
- 2020: 50 billion devices predicted to be connected to the IoT [7]

The European Commission has big plans for the use of the IoT by 2020, as noted in its digital agenda. It is predicted that in just 6 years time, there will be over 50 billion devices connected to the IoT [7]. Overall, the IoT is still in its infancy and there are many promises for its application in the future, with seemingly endless possibilities.

OPERATING PRINCIPLES

In order to connect an object to the IoT, several things are needed in the hardware and software realm. First of all, if one wishes to go beyond simply connecting data from a computer, objects to gather (sensors) or receive (actuators) data are necessary. For example, a digital thermometer can be used to measure temperature. In this case, the data needs to be uploaded to a network of connected servers which run applications [16]. Such a network is commonly referred to as 'the cloud'. The cloud utilizes the process of visualization, meaning that several physical servers can be connected and used in tandem, but appear to the user as one machine (despite that at the physical level, the machines function independently) [9]. This method of computing thus allows changes to be made to the 'virtual' server (such as software updates or changes in storage space) much easier than before.

In this case, an object will connect to the cloud through a (possibly wireless) Internet connection to upload or receive data. Objects to be connected are typically augmented with either sensors or actuators. A sensor is something that tells us about our environment. Think of a temperature sensor, or even the GPS receiver on your mobile phone. Actuators are something that you want to control. Things like thermostats, lights, pumps, and outlets. The IoT brings everything together and allows us to interact with our things and, even more interestingly, allows things to interact with other things.

For the purpose of connecting an object to the IoT, we focus on the *ThingSpeak* API. The interface provides simple communication capabilities to objects within the IoT environment, as well as interesting additional applications (such as *ThingTweet*, which will be further discussed in a later section). Moreover, *ThingSpeak* allows you to build applications around data collected by sensors. It offers near real-time data collection, data processing, and also simple visualizations for its users. Data is stored in so-called channels, which provides the user with a list of features [32]. Each channel allows you to store up to 8 fields of data, using up to 255 alphanumeric characters each. There are also 4 dedicated fields for positional data, consisting of: Description, Latitude, Longitude, and Elevation. All incoming data is time and date stamped and receives a sequential ID. Once a channel has been created, data can be published by accessing the *ThingSpeak* API with a 'write key', a randomly created unique alphanumeric string used for authentication. Consequently, a 'read key' is used to access channel data in case it is set to keep its data private (the default setting). Channels can also be made public in which case no read key is required.

According to the *ThingSpeak* website, the API works as noted in Figure 1. Essentially, 'things' are objects that are given sensors to collect data. Data is sent and received via simple "Hypertext Transfer Protocol" (HTTP) POSTs, much like going to a web page and filling out a form. This communication happens through plaintext, *JSON* or *XML*. The data is then uploaded to the cloud and from there can be used for a variety of purposes. In turn, data (such as commands or choosing certain options) can be gathered and communicated to the cloud, which in turn sends these messages to the object.

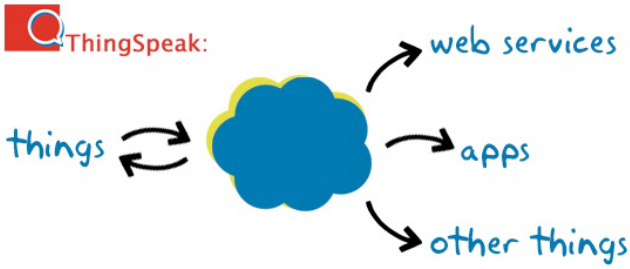

Figure 1. ThinkSpeak representing itself as 'cloud' interface

A deeper level of what occurs, especially on the server side, can be seen in Figure 2 [15]. When a device sends data through a HTTP request (communication), it is processed by the IoT service (in this case *ThingSpeak*), which communicates with a virtual server. Both the server and the IoT service communicate directly with the application. Finally, at all levels of communication from the device to the application there is both requirements regarding security and management of the data transfer. Unfortunately, *ThingSpeak* does not document how the specific parts of the diagram are handled on a technical level. Given enough time and expertise, it should however be possible to answer this by looking into the (open source) code base.

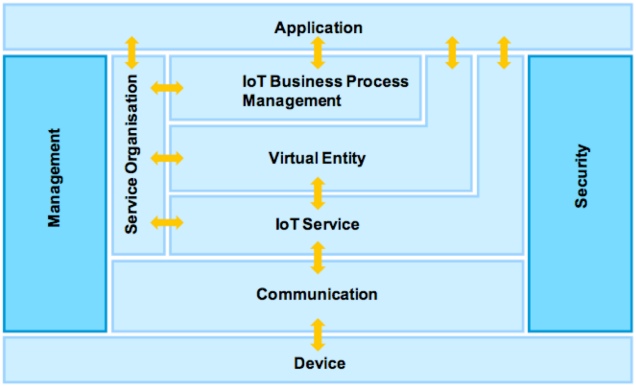

Figure 2. Model of the IoT

For this report, we take the example of connecting a doorbell (herein referred to as the 'tweeting doorbell') to the *ThingSpeak* web service which will send a message to Twitter whenever the doorbell was rung.

The connection from *ThingSpeak* to *Twitter* [1] is handled by the application 'ThingTweet', a script independent from the *ThingSpeak* API but available when using the API from the *ThingSpeak* web service. Messages from *ThingSpeak* are communicated to *Twitter* via an HTTP POST. This feature actually enables users to bypass displaying data in a channel and directly upload it to *Twitter*, if desired.

STRENGTHS AND WEAKNESSES

There are several notable reasons why we chose the *ThingSpeak* API over other available IoT APIs. This section will discuss the strengths and weaknesses of *ThingSpeak* in comparison to a handful of its direct competitors, including *Carriots* [4], *SmartObject* [25], *Skynet* [24], and *Sensorthings* [22]. Each API offers a similar service to *ThingSpeak*, although with some nuances. For instance, *Carriots* seems to mainly focus on business and industry applications. Hence, the API is not open source and charges for parts of its services. On the other hand, other alternatives like *Skynet*, *SensorThings* and *SmartObject* are all open source, and can be changed and adapted to the will and needs of the user just like *ThingSpeak*.

These open source APIs are quite similar in terms of functionality, each with their own strengths. As an example, the *Skynet* API is most tested and used with Arduino boards, therefore there is a lot of user generated programs and support for using this particular board. On the other hand, *SmartObject* is not as user-friendly as *Skynet* or *ThingSpeak*, and one really needs to be knowledgeable of programming in order to get started with this API. It is therefore not recommended for beginners or those who just wish to experiment on a very basic level. Finally, the *SensorThings* API on the surface appears to operate under similar principles as *ThingSpeak*, such as using HTTP requests to transfer data. Yet, the API has split its data model into two parts; sensing and tasking. Hence, there is a distinctive profile depending on if you are using the API to simply gather sensor data, or to communicate with an actuator. This would be most useful when troubleshooting issues, or if you wish to use the API for either sensing or tasking separately. Similarly however, all platforms have the potential to be compatible with almost any open or custom hardware, as long as it has connectivity to the Internet and can talk to the aforementioned APIs.

ThingSpeak Strengths

The key thing which separates *ThingSpeak* from any of the mentioned competitors is that it creates a sense of community through the possibility of creating public channels. It is noted to be the only open data platform specifically designed for the IoT in 'the cloud' [11]. Also, the API allows for very easy visualization of collected data through using spline charts. Therefore it is visually appealing and is much easier when examining collected data compared to other open source APIs.

Another point in *ThingSpeak*'s favor is the fact that it uses *Phusion Passenger Enterprise*, a web and application server. Therefore, the API provides additional support for the programming languages *Ruby*, *Python* and *Node.js*. These languages are noted to be powerful and popular, so additional support and features for these are seen as a nice supplement. Although the other open source APIs are able to work with most languages, extra features for the most popular ones are always a bonus. Further, it is unclear whether or not the servers of the other APIs are running *Phusion Passenger*.

Unlike *Carriots*, the APIs *ThingSpeak*, *SmartObject* and *SensorThings* are open source and are therefore able to be integrated with any hardware device, including *Arduino*, *Raspberry Pi* and any home-made micro-controller. The source for *ThingSpeak* includes processing HTTP requests, storing (alphanumeric and numeric) data, processing numeric data, location tracking and status updates. In addition, anyone is able to develop the platform, so if a desired feature does not exist, anybody is free to code it. Therefore, open-source APIs that allow connectivity with any micro-controller would be the ideal ones to choose between when conducting any "do-it-yourself" (DIY) project.

Another benefit of having an open source API is that it can be run locally or on your own server. Therefore, a user is provided with a lot of flexibility and control for their projects. The advantage of *ThingSpeak* over the others however, is that free hosting is provided for data channels. As mentioned earlier, these channels can be private or public. Not only does this make it easier for someone who does not operate and maintain a server, but the use of HTTP POST and GET for the data is relatively easy for newcomers to web technology. Lastly, public channels also serve as a source of inspiration, as users are able to examine and admire the projects of others.

All things considered, the API which is strongest for anybody wishing to connect an object to the IoT really depends on exactly what the individual (or organization) wishes to do. Each have their own strengths depending on what is required. *ThingSpeak* excels at being one of the simpler APIs to get started quickly, as well as the ability to use a community server to host a channel to upload data. It also provides additional features for *Ruby*, *Node.js* and *Python*.

ThingSpeak Weaknesses

After considering all the strengths, there are some aspects of the *ThingSpeak* API that can be called its weaknesses. For example when uploading data to the API there is a limit up one update per channel every fifteen seconds. We postulate that the reason for this limit in uploads is due to the excess bandwidth that could be used, and therefore would end up costing *ThingSpeak* additional funds for a non-profit service. This limit can be removed by taking the whole API to another web host provider to run the API.

Although it is not a weakness in comparison with the competing APIs introduced earlier, it would be nice to have other types of charts available for graphing data, such as pie charts, bar charts and histograms. However, it is very simple to export the data to other programs that are able to build more complex and colorful charts to display data.

As with many open source projects, using the *ThingSpeak* API can be a hurdle for beginners. In other words, *ThingSpeak* is not necessarily a 'turn key' API, although we have noted it to be slightly simpler than some of its competitors. Consequently, to use *ThingSpeak* some coding knowledge is needed. *Carriots* or some of the other paid APIs offer code snippets that make integration easier. As they are paid, they offer more support as well. In a similar manner, the *ThingSpeak* community is only moderately active (roughly 3 days between posts on their forum) [28]. As a consequence, if one needs technical help from others in the community, it may be frustrating to wait a relatively long amount of time to receive a response.

Another weakness was found while developing the tweeting doorbell example. One of the biggest downfalls of the API and using *Twitter* is the fact that *Twitter* messages have to be unique, or more specifically two identical messages can not be used after each other due to spam protection on the *Twitter* side. We can think of a number of examples where the message will always be the same, for example an automatic TV shutdown timer thats tweets "TV is shutdown" or an autonomous car that parked "I am parked in you garage". When using this extension of *ThingSpeak* it is therefore necessary to add unique data (such as a timecode) to ensure undisrupted operation. It is perhaps not necessarily a limitation of *ThingSpeak* itself but something that should be kept in mind when working with one of its features.

As a last note, we have to mention something about privacy. Channels are by default set on private and users or machines need a read or write key to access the data. Changing the channel to public makes the data available for everyone without the need for a read key. *ThingSpeak* does not inform on their website where the data is stored and how it is secured. It is also not known for what duration data is stored on their servers.

TYPICAL APPLICATIONS

Although the original intended use of the *ThingSpeak* API is to 'give voice' to everyday objects, it seems that there is a very common emerging trend on what the users are building and sharing with the IoT. Scholars note that the IoT will be most useful in an organizational environment, especially for inventory management, production efficiency, waste management, urban planning, environmental sensing, social interaction gadgets, continuous care, emergency response, smart product management, as well as other uses focusing on creating an efficient and sustainable urban environment [18]. For individuals and private homes, the IoT is predicted to incorporate smart metering of electricity, home automation and intelligent shopping [13]. Overall then, the typical applications of any API used to connect objects to the IoT is broad with far-reaching implications.

More specifically though, *ThingSpeak* is an open source API. Therefore, we expect much more 'home/private' use applications rather than organizational as this is the typical way that new, open source technologies are first put to use. Without a doubt, the majority of public channels are focused on sensors in homes. Interestingly, most of these were measuring weather data, especially temperature (indoor/outdoor), humidity, light levels and atmospheric pressure. Hence, most *ThingSpeak* users choose to apply the API for personal measures. That said, it was not completely uncommon for *ThingSpeak* to be applied to organizations and some of those used it to monitor the temperature and humidity of an office.

A second common use found for *ThingSpeak* is still connected to light measurement, but not to determine how bright a room is. Instead, ThingSpeak was used to measure the amount of energy created by a photovoltaic panel [14]. This particular panel was located in a remote field in Germany. It is not uncommon for roof space to be rented in rural areas for placement of such panels where one can be paid for feeding the public power grid. Therefore, many owners of photovoltaic panels are nowhere near their placement, which makes it critical to be able to monitor the power generation from a remote location. The ability to monitor panels allows owners to see which have the most ideal placement, and also if there are any technical errors with the panels.

Furthermore, still in line with measuring the environment, *ThingSpeak* has also been used to measure and control the temperature of a water bed [29]. This is interesting since one can note the ideal temperature for getting into bed, and also perhaps be able to detect when the bed is in use by (unwanted) inhabitants, such as pets. Hence, there are many different uses and necessities for measuring changes in ones environment.

Finally, another common use is the use of *ThingSpeak* to monitor the electric consumption of a household [21]. This is useful for people to note the times which electricity consumption is at its highest, and perhaps alter their usage patterns to reduce grid load and save money. However, there are some dangers with making this data public, which will be discussed in the upcoming section.

SURPRISING APPLICATIONS

While truly surprising examples of *ThingSpeak* are currently sparse, it is only a matter of time before more outstanding applications arise. It is also interesting to consider some 'out of the box' applications (or perhaps, implications) of having so much device-use data available by allowing *ThingSpeak* to upload data to the IoT.

One surprising example of using *ThingSpeak* is a project called 'CheerLights' [6].This project allows people's lights all across the world to synchronize, stay linked based on social networking trends. It's a way to connect physical things with social networking experiences. In short; everyone can send a message to *Twitter* containing the keyword `#cheerlights`. The 'TweetControl' App from *ThingSpeak* is used to listening to *Twitter*, when a tweet contains the keyword, TweetControl updates the CheerLights *ThingSpeak* channel with the last command received. The lights that listen to this channel will change color. This project ended up being so popular that it won the best DIY IoT project of 2013 [12].

Another example discovered made use of the 'React'

application included in the *ThingSpeak* web service [23]. This application allows the user to trigger an HTTP request or send a tweet when a channel meets a set condition (hence, it appears the device is 'reacting' in an autonomous way). In this example, by using geographical data, the channel owner can ensure that their home thermostat is turned on just on time depending on the proximity of the owner [3]. This ensures that their home is the right temperature upon arrival.

Although this is not using the *ThingSpeak* API specifically, there has been a concept design by artist Nathan Brunstein [17], which envisions a toaster that imprints the weather forecast of the day on your toast. Hence, toast-eaters of the future might not need to look further than their breakfast to ascertain the weather. Unfortunately, this is only a concept and is yet to be created. But, it does show creative ways to employ the IoT.

Furthermore, judging from the current popular channels on *ThingSpeak*, the use of the API to connect a doorbell to the IoT is somewhat a surprising application. On the surface it may seem as though it is quite a useless device to communicate, but there are uses for those who may be hard of hearing, or if the use of headphones prevents one from hearing the doorbell. Therefore, a tweeting doorbell can be seen as an unconventional and useful use of the *ThingSpeak* application.

As a last point, we must consider the implications or unintended use of this technology; crime may become easier for criminals. If access is made to something as seemingly innocent as electricity use in the home, or if for instance, the temperature of the waterbed is monitored, everyone could check if the residents are sleeping, or have a night out. Even more worrisome is publishing the power usage of ones domicile [10]. With the graphed information, it is easy to see if these people are home and thus consuming electricity. Additionally, for many channels, users provide geolocation information. This private and sensitive data is stored in the cloud, accessible for everyone. Thus anyone can make predictions of people, based on the data about their lives that they share with the world.

GETTING STARTED

In this section we describe how to set up a small example with ThingSpeak. Note that in this section we personally address the user as 'you' to make directions easier to read.

First Steps

Firstly, although it is not required in actually setting up the API, for *ThingSpeak* to be of any meaning, it is imperative to have a micro-controller with an internet connection. The type of micro-controller is completely up to the user, as *ThingSpeak* is able to communicate with any type, as long as it is networked. In our example, we have used an Arduino board with an ethernet shield. Once the micro-controller has been chosen, the first steps to setting up *ThingSpeak* are always the same. After having signed up for a new user account you can log in and create new channels [27]. When logged in, you can create a new channel by selecting **Channels** > **My Channels** and then **Create New Channel**. The channel has an unique identifier key which is used to identify the channel when reading or uploading data.

Uploading Data

Each channel has up to eight fields where data (both numeric and alphanumeric formats) can be stored, as well as four additional fields for location details. All entries are stored with an unique identifier and a date and time stamp. Existing data can be imported from a "Comma-Separated Values" (CSV) file, which is a popular format for storing tabular data.

If you do not have existing data you can immediately continue to putting your own data to your channel. This is done by using the POST method in HTTP in combination with your unique channel write key. In the channels view, select the channel and select the API keys tab. From here we can already reach the 'Hello world' stage with the API, as seen in Table 1.

```
POST /update HTTP/1.1
Host: api.thingspeak.com
Connection: close
X-THINGSPEAKAPIKEY: (HGFKU61VOPOEO77B)
Content-Type: application/x-www-form-urlencoded
Content-Length: (number of characters in message)
field1=(hello world)
```

Table 1. Sending 'hello world' to field 1

This HTTP POST can be used in all kinds of programming such as in the Arduino IDE. But, clearly it is desirable to do more than just write 'Hello world' to a meaningless, object-free channel, hence we will continue on how to get data to a *Twitter* channel.

Arduino IDE and ThingSpeak Integration

One of the example uses of the *ThingSpeak* API was the ThingTweet add-on, which enabled the creation of a doorbell that would tweet (from the account of the doorbell) whenever someone 'rang' it. For this example we used an Arduino with an ethernet shield. The Arduino continuously checks if there is a signal coming in, the signal would come when the doorbell was pressed. In other words, the moment the doorbell is pressed, an update is sent to *ThingSpeak* via a simple HTTP post using the API. This post to the server contained a message to be tweeted including the date and time the doorbell when rang.

```
if (buttonState == HIGH) {
updateThingSpeak("field1=1&twitter=doorbelldaan&tw
eet=Ring Ring);
void updateTwitterStatus(String tsData){
if (client.connect(thingSpeakAddress, 80))
  { // Create HTTP POST Data
   tsData = "api_key=" + thingtweetAPIKey +
   "&status=" + tsData;
   client.print("POST
/apps/thingtweet/1/statuses/update HTTP/1.1\n");
   client.print("Host: api.thingspeak.com\n");
   client.print("Connection: close\n");
   client.print("Content-Type: application/x-www-
form-urlencoded\n");
```

```
  client.print("Content-Length: ");
  client.print(tsData.length());
  client.print("\n\n");
  client.print(tsData);
  }
}
```

Table 2. ThingTweet code on Arduino

The *ThingSpeak* API was connected to a *Twitter* account that we created for the doorbell [8]. The date and time had to be included to avoid having the message flagged as spam, as described earlier. Connecting the accounts took two clicks to accept and allowed the API to update *Twitter*. To get this connection, log in, click Apps, click the ThingTweet button, and finally click Link Twitter Account.

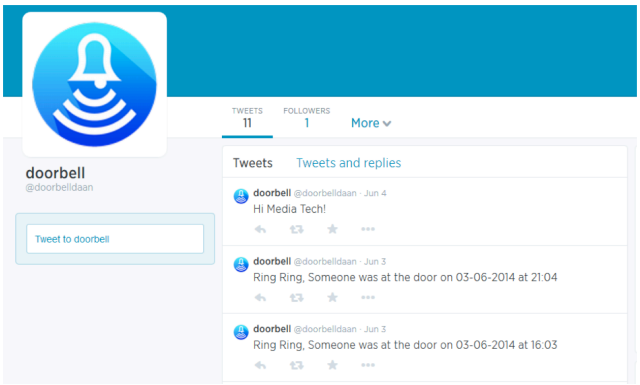

Figure 3. Screenshot showing the Twitter doorbell

Retrieval of Data

To get the data from *ThingSpeak* you can retrieve the data by time selection or by entry ID. The most direct way is to use the HTTP GET method [27]. In case the channel is not public (and by default it is not), retrieving data requires a read key, which can be acquired in the channel view tab.

It is also possible to retrieve just the latest entry through modifying the URI below; where the information in the brackets will be replaced with the channel id of the user, the field id and the type of data to be retrieved:

```
http://api.thingspeak.com/channels/
(channel_id)/field/(field_id)/last.(format)
```

The channel supports GET formats including JSON, XML, and CSV formats for integration into other (third party) applications.

One example of how data retrieval can be automated is by using *Python*, which can be run under most modern operating systems, including *Raspbian* on the Raspberry Pi. We have included a full code example below which continuously checks a given channel for updates and prints a message every time an update takes place.

```
import urllib, json
import datetime, time
import threading

channel = "10101" # channel id
read_api = "01ABC01" # API key
check_interval = 5;  # in seconds

next_call = time.time()
url = "http://api.thingspeak.com/channels/" + channel +
"/feed.json?key=" + read_api
response = urllib.urlopen(url);
data = json.loads(response.read())
channeldata = data[u'channel']
channeldata = channeldata[u'last_entry_id']
cache = channeldata

print "RUNNING Doorbell Script"
print "CTRL+C to end"

def checkChannel():
        global next_call
        global cache
        global channel
        global read_api

        url = "http://api.thingspeak.com/channels/" +
channel + "/feed.json?key=" + read_api
        response = urllib.urlopen(url);
        data = json.loads(response.read())
        channeldata = data[u'channel']
        channeldata = channeldata[u'last_entry_id']

        if cache != channeldata:
                print "CHANNEL UPDATE!"
                cache = channeldata

        next_call = next_call + check_interval
        threading.Timer( next_call - time.time(),
checkChannel ).start()

checkChannel();
```

Table 3. Python code for basic notifications

Presentation

As previously mentioned, aside from getting the data and converting it to a graphic yourself, *ThingSpeak* can do this automatically [5]. The API allows for numeric data processing such as time scaling, averaging, median, summing, and rounding. For this you use the chart functionality that is a part of the *ThingSpeak* API. This allows you to make visualizations of the data which can be updated in real time. There is also a short script provided to add it to your own website, shown in Table 4.

```
<iframe width="450" height="250" style="border:
1px solid #cccccc;"
src="http://thingspeak.com/channels/
(channel_id)/charts/(field_id)"></iframe>
```

Table 4. HTML code to display channel charts

As you can see, it is not difficult to get started with ThingSpeak with a little programming proficiency.

FINAL THOUGHTS

The IoT can provide a great number of benefits to our modern society. As we have seen, there is much potential for both individuals and organizations to connect objects to the IoT. The vision of "smart cities" which allow for maximum efficiency is at the forefront of the European Digital Agenda.

The IoT was coined prior to the turn of the millennium, and since then there are many open source APIs that one can use if they wish to connect any object to the IoT. The *ThingSpeak* API in particular is an excellent starting point,

as it provides a free platform for data exchange. This is particularly useful for students, as there is no limit of channels that can be used for connectivity without additional costs. That said, the *ThingSpeak* web service is more useful than the API alone, since it also provides web-server maintenance (with additional services for popular programming languages) and the ability to peek at other projects through open channels.

When considering the future, it is not difficult to predict an increasing miniaturization of technology. As such, almost everything can potentially become capable of data exchange. Micro-controllers will shrink and disappear into the environment, leading to the IoT becoming ubiquitous and omnipresent. As this happens, when will it stop being called the "Internet of Things"? Instead, the normalcy of intelligent, interconnected objects will no longer warrant a special distinction for their connectivity.

Caution is also necessary when uploading data to the IoT. One should not be overly insouciant with their personal data. Consequently, trust is still a major issue. It is important to consider if one can entrust personal data with organizations. However, if the past few decades are any indication, the perfunctory approach to protecting personal information is lost online. Social networks encourage to shed anonymity and an entire generation is consequently raised to share with abandon. The current laissez faire approach to data may produce a more personalized, efficient and technologically driven future with the IoT. Or perhaps, one not so favorable. All in all though, the IoT is surely an interesting web technology that has the potential to shape the future – and *ThingSpeak* is an easy way to get acquainted with it.

REFERENCES

1. About Twitter: *https://about.twitter.com/*. Accessed: 2014-06-08.
2. Ashton, K. 2009. That “internet of things” thing. *RFiD Journal*. 22, (2009), 97–114.
3. Automatic Thermostat Control Based on Location and Weather: 2010. *http://iamshadowlord.com/2010/09/automatic-thermostat-control-based-on-location-and-weather.html*. Accessed: 2014-06-08.
4. Carriots - Internet of Things Platform: *https://www.carriots.com/*. Accessed: 2014-06-08.
5. Chart API: *http://community.thingspeak.com/documentation/api/#charts*. Accessed: 2014-06-08.
6. CheerLights - ThingSpeak: *https://thingspeak.com/channels/1417*. Accessed: 2014-06-08.
7. Digital Agenda for Europe - European Commission: *http://ec.europa.eu/digital-agenda/*. Accessed: 2014-06-08.
8. Doorbell on Twitter: *https://twitter.com/doorbelldaan*. Accessed: 2014-06-08.
9. EMA White Paper 2013. Cloud 2.0: Delivering Value to the Enterprise.
10. Energy Monitor – ThingSpeak: *https://thingspeak.com/channels/10867*. Accessed: 2014-06-08.
11. Features - ThingSpeak: *https://thingspeak.com/pages/features*. Accessed: 2014-06-08.
12. Holiday Innovation: Tweet At Your Christmas Tree To Light It Up: 2013. *http://www.npr.org/blogs/alltechconsidered/2013/12/24/256888326/holiday-innovation-you-can-tweet-at-your-christmas-tree*. Accessed: 2014-06-08.
13. Höller, J. et al. 2014. *From Machine-to-Machine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press.
14. Inferno Photovoltaic - ThingSpeak: *https://thingspeak.com/channels/10958*. Accessed: 2014-06-08.
15. Internet-of-Things Architecture - Updated Reference Model: *http://www.iot-a.eu/arm/d1.3*. Accessed: 2014-06-08.
16. Is the Cloud Really Just the Return of Mainframe Computing?: 2011. *http://sqlmag.com/cloud/cloud-really-just-return-mainframe-computing*. Accessed: 2014-06-08.
17. Jamy - Smart Toaster: 2011. *http://legrandours.com/3924/647021/gallery/jamy-smart-toaster*. Accessed: 2014-06-08.
18. Kyriazis, D. et al. 2013. Sustainable smart city IoT applications: Heat and electricity management & Eco-conscious cruise control for public transportation. *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a* (2013), 1–5.
19. Magrassi, P. and Berg, T. 2001. A world of smart objects: The role of auto identification technologies. *Strategic Analysis Report, Gartner*. (2001).
20. Mattern, F. and Floerkemeier, C. 2010. From the Internet of Computers to the Internet of Things. *From active data management to event-based systems and more*. Springer. 242–259.
21. My house power consumption - ThingSpeak: *https://thingspeak.com/channels/2567*. Accessed: 2014-06-08.
22. OGC SensorThings API: *http://ogc-iot.github.io/ogc-iot-api/*. Accessed: 2014-06-08.
23. React: *http://community.thingspeak.com/documentation/apps/react/*. Accessed: 2014-06-08.
24. Skynet: *https://www.npmjs.org/package/skynet*. Accessed: 2014-06-08.
25. SmartObject: *https://github.com/mjkoster/SmartObject*. Accessed: 2014-06-08.

26. Srivastava, L. 2011. The Internet of Things—Back to the Future. *Proceedings of IoT 2011 Conference 16th May* (2011).
27. ThingSpeak Channels: *http://community.thingspeak.com/tutorials/thingspeak-channels/*. Accessed: 2014-06-08.
28. ThingSpeak Community Forum: *http://community.thingspeak.com/forum/*. Accessed: 2014-06-08.
29. Water Bed - ThingSpeak: *https://thingspeak.com/channels/277*. Accessed: 2014-06-08.
30. What is Internet of Things (IoT)? - Definition from WhatIs.com: 2013. *http://whatis.techtarget.com/definition/Internet-of-Things*. Accessed: 2014-06-08.
31. What is machine-to-machine (M2M)?: 2010. *http://whatis.techtarget.com/definition/machine-to-machine-M2M*. Accessed: 2014-06-08.
32. Introduction to the “Internet of Things” and ThingSpeak. *ThingSpeak Community*.