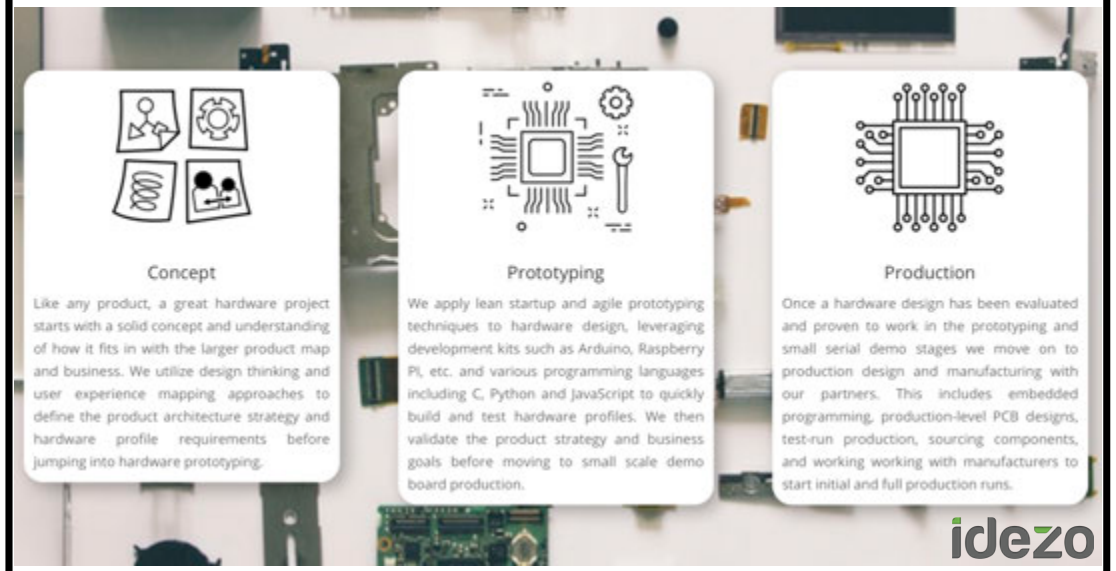
HANDS ON APPROACH TO

Data Science

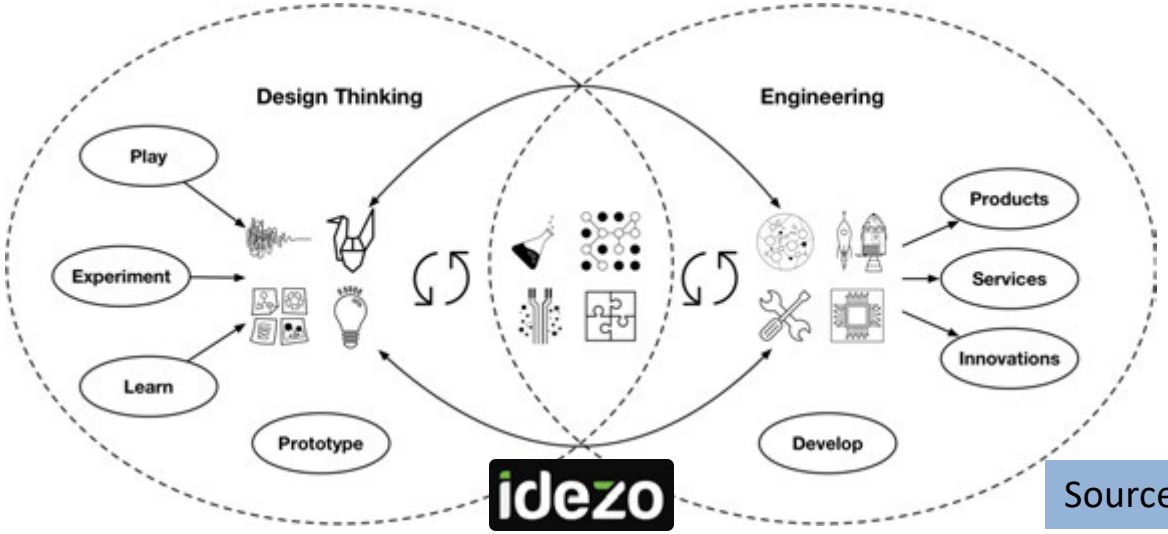
for (the)

IoT



Rob van der Willigen

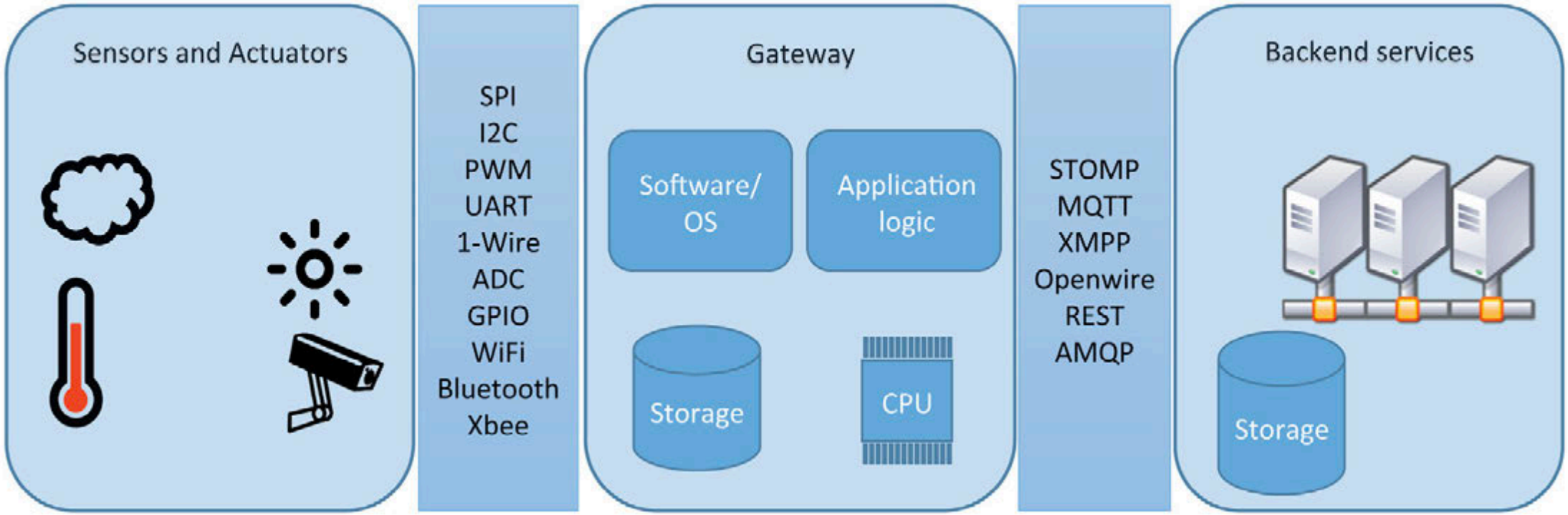
HANDS ON APPROACH TO DATA SCIENCE for (the) IoT



Source: https://idezo.ch

This Data Science Course was developed for keuzevak- program of the School of Communication, Media and Information Technology (CMI, at the Hogeschool Rotterdam **(Rotterdam University of Applied Sciences, RUAS)**.
If you find errors or omissions, please contact the author, Rob van der Willigen, at r.f.van.der.willigen@hr.nl. Materials of this course and code examples used will become available at:
https://github.com/robvdw/CMIDAT01K-DATA-SCIENCE-for-IOT

IoT Concepts: key Components & Protocols



SENSOR TO GATEWAY COMMUNICATION

Course Setup

Lesson 01:	**Discovering the IoT Data Science Domain**
Lesson 02:	**Defining project requirements**
	+ Cost calculation/estimate
Lesson 03	**Learn to write code**
Lesson 04	**Data Science: How to start your own IoT Project**
Lesson 05	**IoT Platforms & MiddleWare**

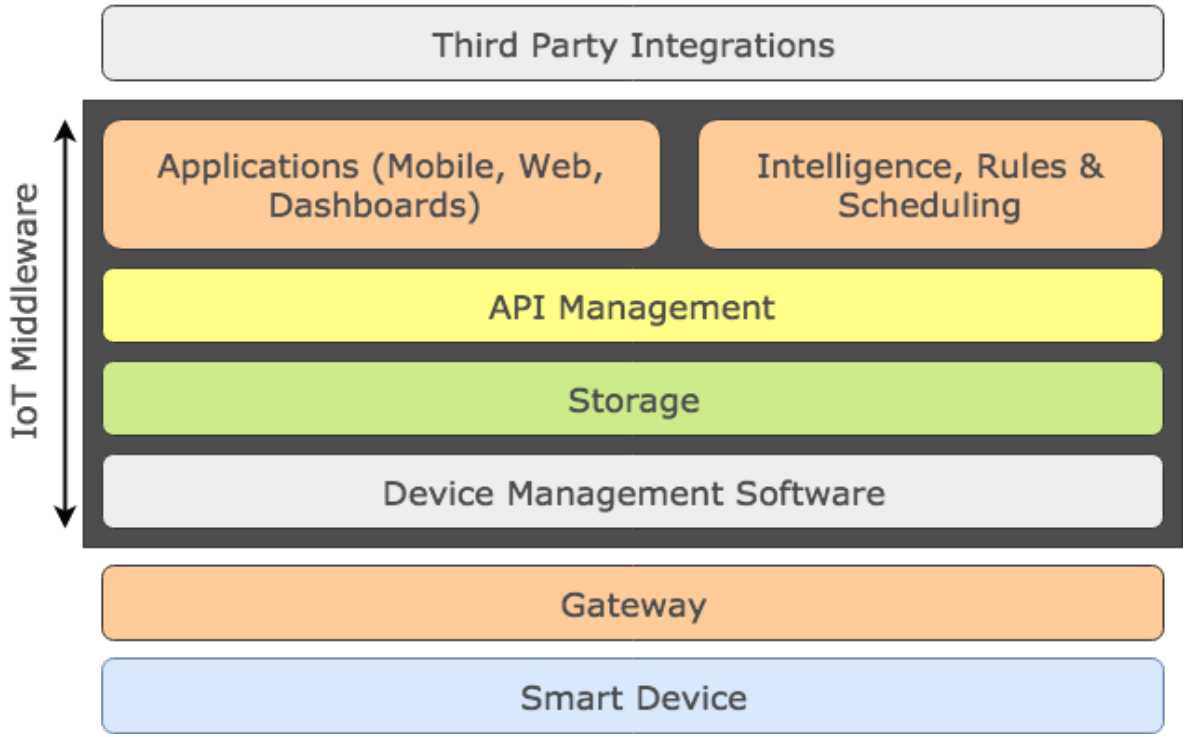
Week 09/10: FEEDBACK + GRADING

lesson five

HOGESCHOOL ROTTERDAM

Getting Started with (the) IoT

IoT Platforms == MiddleWare



IoT Platforms / MiddleWare

Why do we need these platforms? Why can't we build one of our own?

We can; however, first we have to consider the following points:

- How long does it take for one to build a piece of end-to-end, bug free IoT middleware?
- Your resources' time versus money spent on building this middleware.
- Your in-house team's ability to build an IoT middleware.
- How generic can you build it? Will this IoT middleware scale for all types of applications?

Since almost all IoT platforms have a similar set of features, using a continuously improving, community contributed platform is always better than building one on your own.

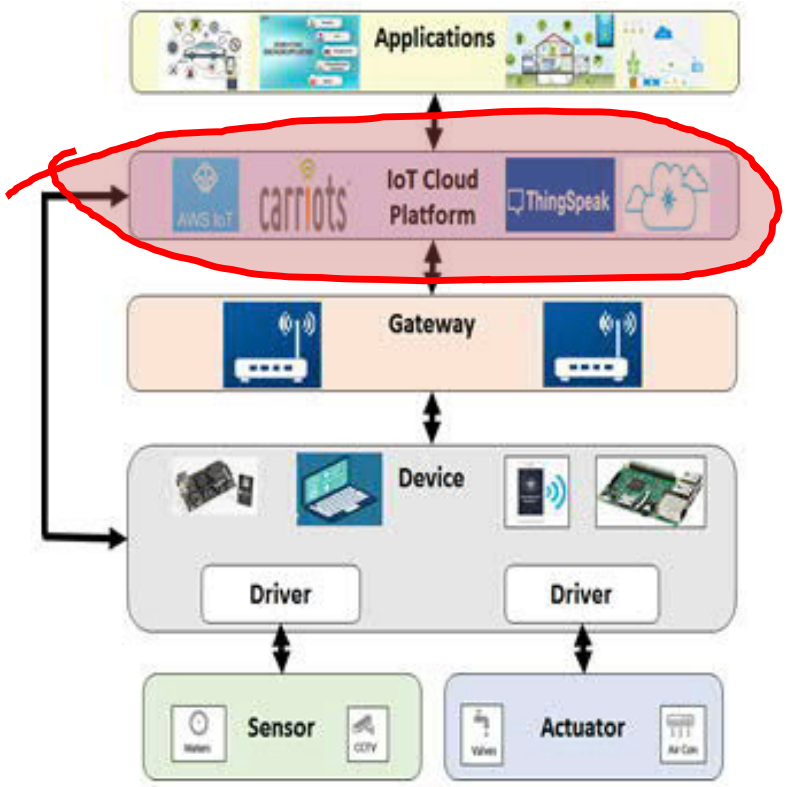
IoT platforms

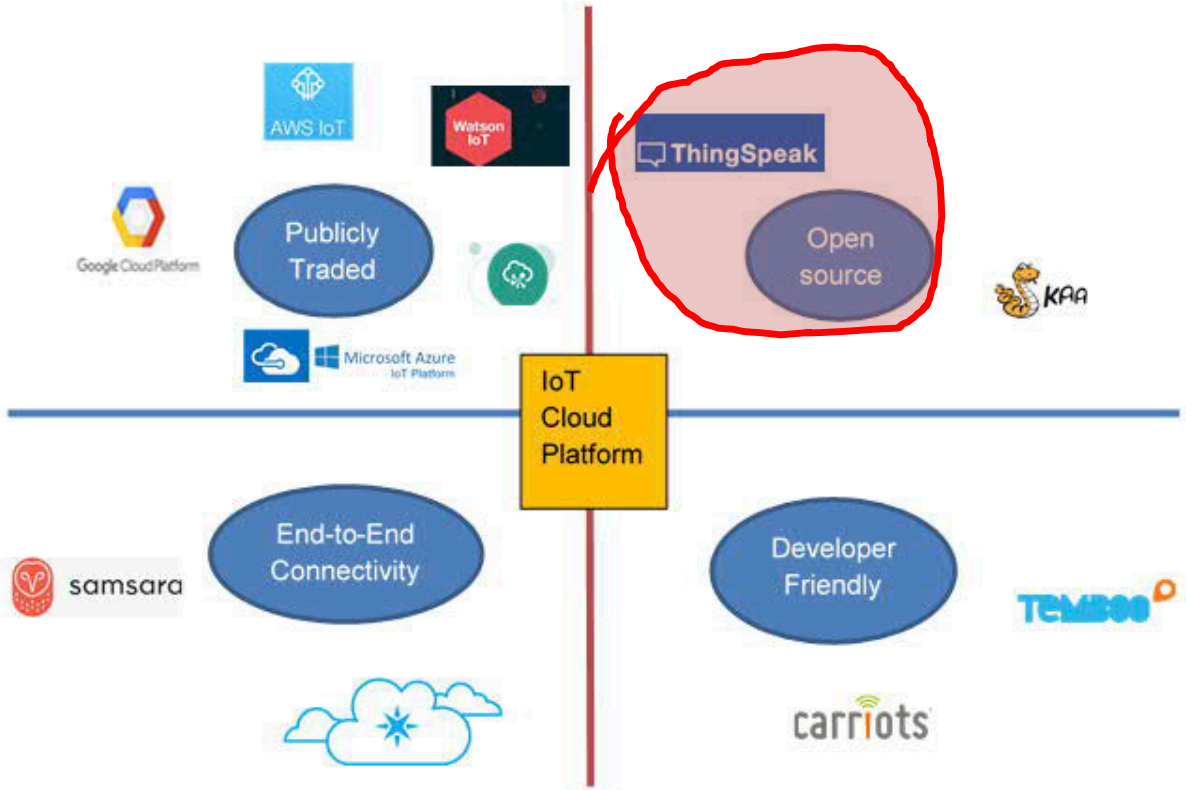
Now that we have a sense of why we are using an existing IoT platform for our IoT solutions, let us look at the options we have for these platforms. The following are a few popular IoT platforms.

You can visit the site links next to the provider to read more about the offerings:

- **AWS IoT**: https://aws.amazon.com/iot/
- **Microsoft Azure IoT**: https://www.microsoft.com/en-in/internet-of-things/azure-iot-suite
- **IBM Watson**: https://www.ibm.com/internet-of-things
- **Google Cloud IoT**: https://cloud.google.com/solutions/iot/
- **Cisco IoT Cloud Connect**: https://www.jasper.com/
- **Salesforce IoT cloud**: https://www.salesforce.com/eu/iot-cloud/
- **Bosch IoT Suite**: https://www.bosch-si.com/iot-platform/bosch-iot-suite/homepage-bosch-iot-suite.html
- **Kaa IoT**: https://www.kaaproject.org/
- **ThingSpeak**: https://thingspeak.com/
- **DeviceHive**: https://devicehive.com/

IoT Platforms / MiddleWare





http://www.thetips4you.com/iot-best-open-source-applications-open-source-industrial-iot-platform

Setting Up Your IoT Device

The Broker. The broker acts as a gateway; it receives messages from a publisher (a client) and delivers the messages to a subscriber (another client). Brokers are sometimes referred to as *servers*.

The Subscriber. The subscriber declares its topics of interest to the broker, and the broker sends messages published to those topics.

The Publisher. The publisher sends messages to the broker using a name- space or a topic name, and the broker forwards the messages to the respective subscribers.

ThingSpeak™ Channels Apps Support Commercial Use How to Buy Sign In Sign Up

Learn More About ThingSpeak

ThingSpeak™ is an IoT analytics platform service that allows you to aggregate, visualize and analyze live data streams in the cloud. ThingSpeak provides instant visualizations of data posted by your devices to ThingSpeak. With the ability to execute MATLAB® code in ThingSpeak you can perform online analysis and processing of the data as it comes in. ThingSpeak is often used for prototyping and proof of concept IoT systems that require analytics.

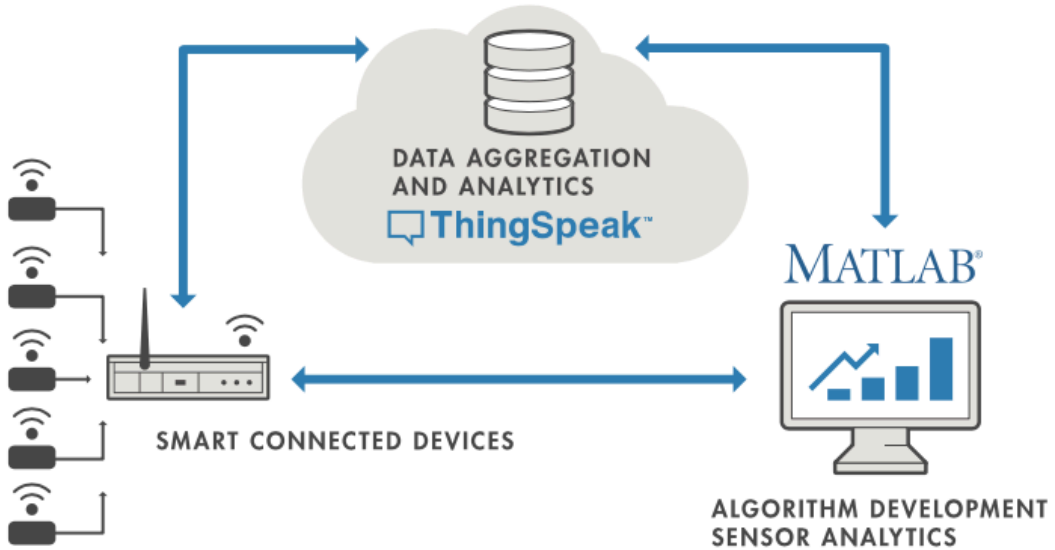
What is IoT?

Internet of Things (IoT) describes an emerging trend where a large number of embedded devices (things) are connected to the Internet. These connected devices communicate with people and other things and often provide sensor data to cloud storage and cloud computing resources where the data is processed and analyzed to gain important insights. Cheap cloud computing power and increased device connectivity is enabling this trend.

IoT solutions are built for many vertical applications such as environmental monitoring and control, health monitoring, vehicle fleet monitoring, industrial monitoring and control, and home automation.

At a high level, many IoT systems can be described using the diagram below:

https://thingspeak.com/pages/learn_more



ThingSpeak™ Channels Apps Support Commercial Use How to Buy Account Sign Out

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

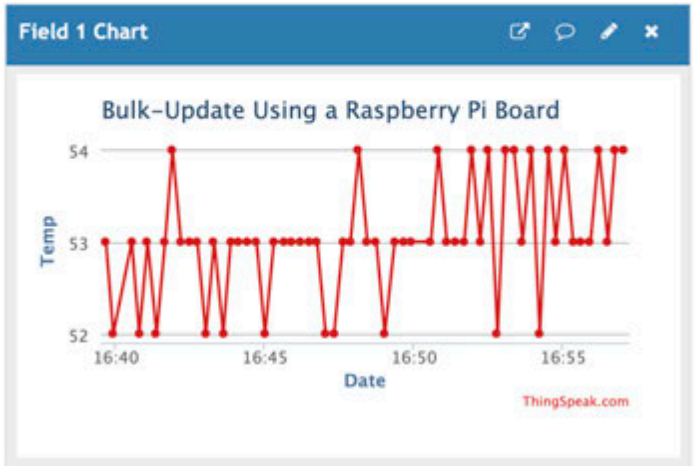
This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

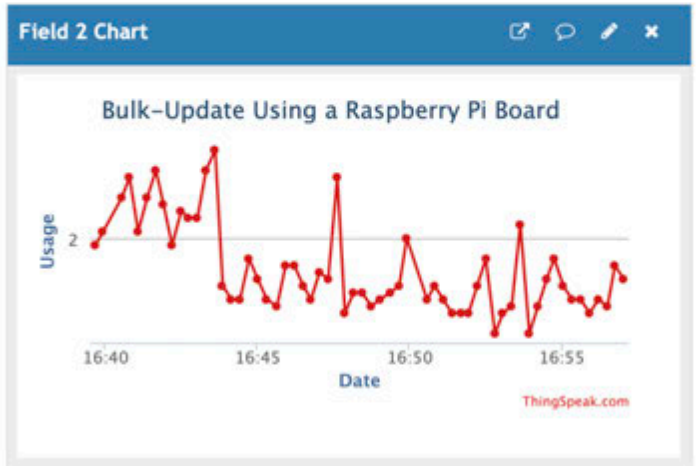
Private View | Public View | Channel Settings | Sharing | API Keys | Data Import / Export

Add Visualizations | Add Widgets | Export recent data | MATLAB Analysis | MATLAB Visualization

Channel Stats

Created: about a month ago
Last entry: less than a minute ago
Entries: 4816





To better understand the ThingSpeak platform, we will build a sample end-to-end IoT application.

Performed will be the following tasks:

- Send data to ThingSpeak via a Channel
- Integrate ThingSpeak with Raspberry Pi implemented via HTTP (LAN network)
- Visualize the sensor data

+ {Set thresholds on the data and receive alerts}

Bulk-Update Using a Raspberry Pi Board

You will learn how to Send sensor data to ThingSpeak via your own Public Channel

Shown is how to use a Raspberry Pi board that runs Python 2.7 that is connected to a Wi-Fi network to collect data.

You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak channel every 2 minutes.

Used is a Bulk-Write JSON Data API to collect sensor data as a batch and send it to your public ThingSpeak channels.

This bulk-update reduces the power usage of your devices.

Since the Raspberry Pi board does not come with a real-time clock, you can use the relative time stamp for bulk-update messages.

How to create a ThingSpeak IoT Cannel

Get Started
Learn the basics of ThingSpeak

Configure Accounts and Channels
Information on ThingSpeak channels, users, and licenses

Write Data to Channel
Use the REST and MQTT APIs to update channels with software or devices

Read Data from Channel
Use the REST and MQTT APIs to read channels using software or devices

Prepare and Analyze Data
Filter, transform, and respond to data in MATLAB

Visualize Data
Transform and visualize data in MATLAB

Act on Data
Use ThingSpeak apps to trigger an action or transform and visualize data

Specialized Analysis with MATLAB
ThingSpeak examples that show use of the advanced tools available in add-on toolboxes

API Reference
Use the REST and MQTT APIs to update ThingSpeak channels and to chart numeric data stored in channel

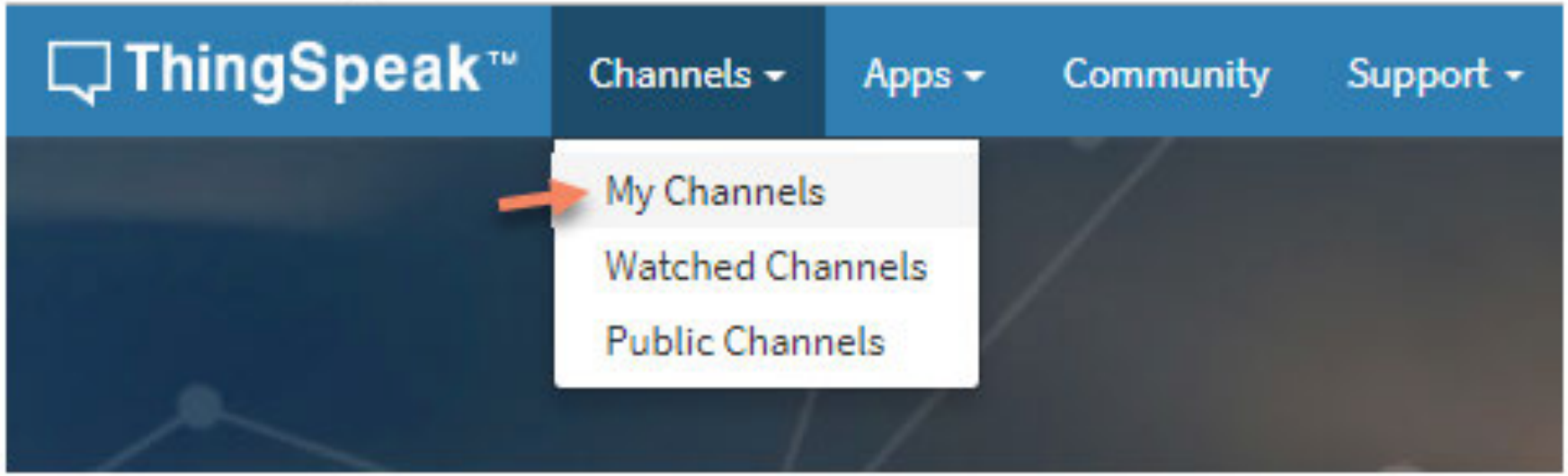
https://nl.mathworks.com/help/thingspeak/index.html?s_tid=CRUX_lftnav

Creating an TingSpeak account

If you do not have an account already, you can visit https://thingspeak.com/users/sign_up to create one.
Once you have created the account, activate it, and log in, you will see an interface like this:

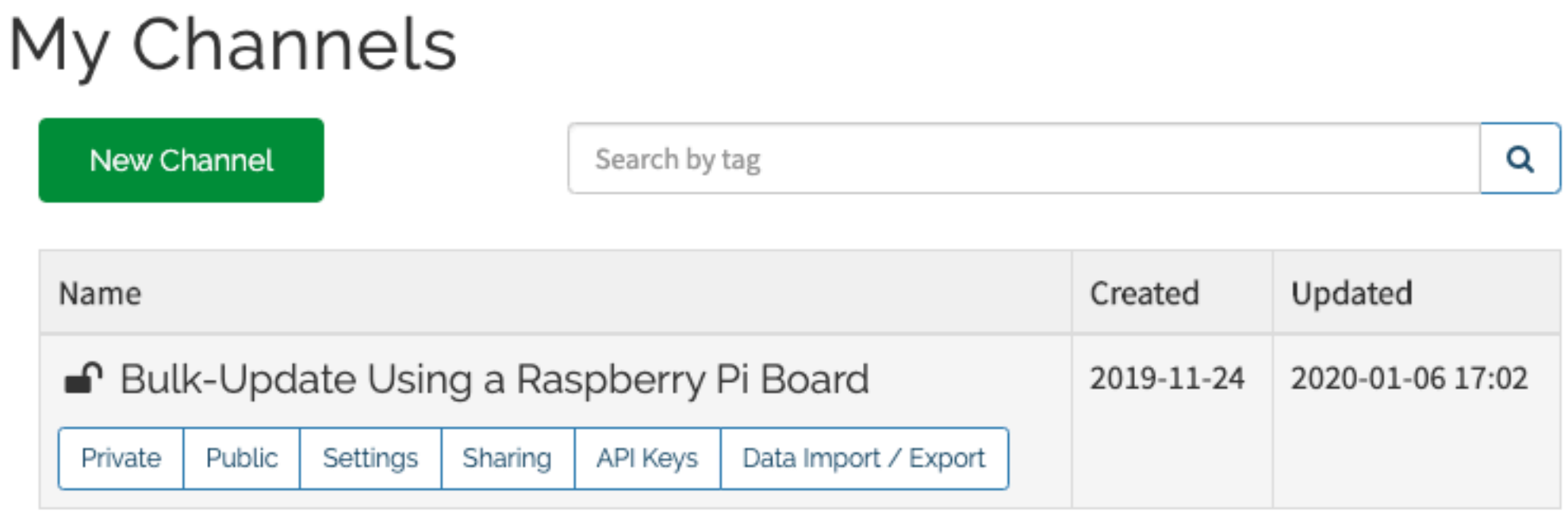
Create a Channel

1. Sign In to ThingSpeak™ using your MathWorks® Account, or create a new MathWorks account.
2. Click **Channels** > **MyChannels**.



Collect Data in a Public Channel called

“Bulk-Update Using a Raspberry Pi Board”



https://www.mathworks.com/help/thingspeak/collect-data-in-a-new-channel.html

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

https://nl.mathworks.com/help/thingspeak/continuously-collect-data-and-bulk-update-a-thingspeak-channel-using-a-raspberry-pi-board.html

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

Channel Settings

Percentage complete 50%

Channel ID 920549

Name: Bulk-Update Using a Raspberry Pi Board

Description: This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a

Field 1: Temp ☑

Field 2: Usage ☑

Field 3: ☐

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.

Bulk-Update Using a Raspberry Pi Board

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

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

Write API Key

Key: X200JB02Z2LN5W8E

Generate New Write API Key

Read API Keys

Key: 2CBID06MIN1M06KZ

Note:

Save Note | Delete API Key

Add New Read 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=X200JB02Z2LN5W8E
```

Read a Channel Feed

```
GET https://api.thingspeak.com/channels/920549/feeds.json?resu
```

Read a Channel Field

```
GET https://api.thingspeak.com/channels/920549/fields/1.json?r
```

Read Channel Status Updates

```
GET https://api.thingspeak.com/channels/920549/status.json
```

Bulk-Update Using a Raspberry Pi Board

Watch | Like 0 | Share

Channel ID: 920549
Author: robfvdw
Access: Public

This example shows how to use a Raspberry Pi™ board that runs Python® 2.7 that is connected to a Wi-Fi® network to collect data. You can continuously collect CPU temperature and percentage of CPU utilization over 15 seconds and bulk-update a ThingSpeak™ c

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

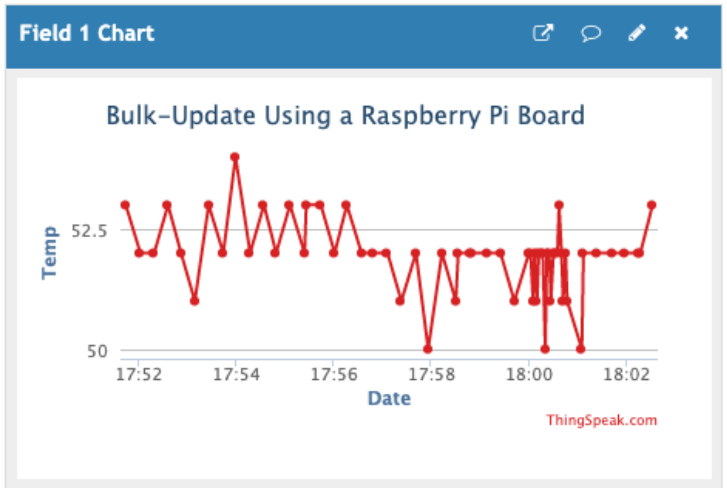
Add Visualizations | Add Widgets | Export recent data

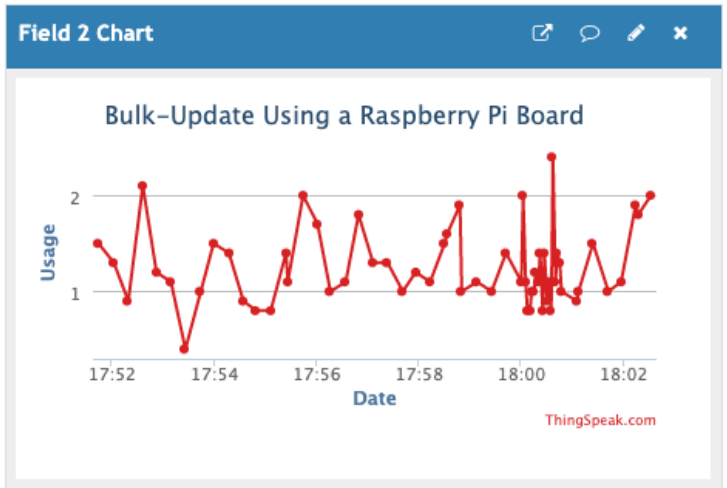
MATLAB Analysis | MATLAB Visualization

Channel Stats

Created: about a month ago
Last entry: about 2 hours ago
Entries: 5150

https://thingspeak.com/channels/920549/





Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#1) Create a channel as shown in Collect Data in a Public Channel called "Bulk-Update Using a Raspberry Pi Board"

#2) Import the necessary libraries for the script.
import json
import time
import os
import psutil
import urllib2 as ul

#3) Define global variables that track the last connection time and last update time. Also, define time intervals to update the data, and post the data to ThingSpeak.
lastConnectionTime = time.time() # Track the last connection time
lastUpdateTime = time.time() # Track the last update time
postingInterval = 120 # Post data once every 2 minutes
updateInterval = 15 # Update once every 15 seconds
```

```
#4) Define your ThingSpeak channel settings such as write API key and channel ID along with ThingSpeak server settings.
writeAPIkey = "YOUR-CHANNEL-WRITEAPIKEY" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "YOUR-CHANNELID" # Replace YOUR-CHANNELID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []
```

```
writeAPIkey = "X2O0JB02Z2LN5W8E" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "920549" # Repla                    LID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []
```

Explorer (⇧⌘E)

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#5) Define the function httpRequest to send data to ThingSpeak and to print the response code from the server. A response code 202 indicates tha
def httpRequest():
    '''Function to send the POST request to
    ThingSpeak channel for bulk update.'''

    global messageBuffer
    data = json.dumps({'write_api_key':writeAPIkey,'updates':messageBuffer}) # Format the json data buffer
    req = ul.Request(url = url)
    requestHeaders = {"User-Agent":"mw.doc.bulk-update (Raspberry Pi)","Content-Type":"application/json","Content-Length":str(len(data))}
    for key,val in requestHeaders.iteritems(): # Set the headers
        req.add_header(key,val)
    req.add_data(data) # Add the data to the request
    # Make the request to ThingSpeak
    try:
        response = ul.urlopen(req) # Make the request
        print response.getcode() # A 202 indicates that the server has accepted the request
    except ul.HTTPError as e:
        print e.code # Print the error code
    messageBuffer = [] # Reinitialize the message buffer
    global lastConnectionTime
    lastConnectionTime = time.time() # Update the connection time
```

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#6) Define the function getData that returns the CPU temperature in Celsius along with the CPU utilization as a percentage.def getData():
'''Function that returns the CPU temperature and percentage of CPU utilization'''
    cmd = '/opt/vc/bin/vcgencmd measure_temp'
    process = os.popen(cmd).readline().strip()
    cpuTemp = process.split('=')[1].split("'")[0]
    cpuUsage = psutil.cpu_percent(interval=2)
    return cpuTemp,cpuUsage
```

Wrting Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#7) Define the function updatesJson to continuously update the message buffer every 15 seconds.
def updatesJson():
    '''Function to update the message buffer
    every 15 seconds with data. And then call the httpRequest
    function every 2 minutes. This examples uses the relative timestamp as it uses the "delta_t" parameter.
    If your device has a real-time clock, you can also provide the absolute timestamp using the "created_at" parameter.
    '''

    global lastUpdateTime
    message = {}
    message['delta_t'] = int(round(time.time() - lastUpdateTime))
    Temp,Usage = getData()
    message['field1'] = Temp
    message['field2'] = Usage
    global messageBuffer
    messageBuffer.append(message)
    # If posting interval time has crossed 2 minutes update the ThingSpeak channel with your data
    if time.time() - lastConnectionTime >= postingInterval:
        httpRequest()
    lastUpdateTime = time.time()
```

Writing Python code to collect data in a public channel called "Bulk-Update Using a Raspberry Pi Board"

```
#8) Run an infinite loop to continuously call the function updatesJson once every 15 seconds.

if __name__ == "__main__":  # To ensure that this is run directly and does not run when imported
    while 1:
        # If update interval time has crossed 15 seconds update the message buffer with data
        if time.time() - lastUpdateTime >= updateInterval:
            updatesJson()
```

```python
import json
import time
import os
import psutil
import urllib2 as ul

lastConnectionTime = time.time() # Track the last connection time
lastUpdateTime = time.time() # Track the last update time
postingInterval = 120 # Post data once every 2 minutes
updateInterval = 15 # Update once every 15 seconds

writeAPIkey = "X2O0JB02Z2LN5W8E" # Replace YOUR-CHANNEL-WRITEAPIKEY with your channel write API key
channelID = "920549" # Replace YOUR-CHANNELID with your channel ID
url = "https://api.thingspeak.com/channels/"+channelID+"/bulk_update.json" # ThingSpeak server settings
messageBuffer = []

def httpRequest():
    '''Function to send the POST request to
    ThingSpeak channel for bulk update.'''

    global messageBuffer
    data = json.dumps({'write_api_key':writeAPIkey,'updates':messageBuffer}) # Format the json data buffer
    req = ul.Request(url = url)
    requestHeaders = {"User-Agent":"mw.doc.bulk-update (Raspberry Pi)","Content-Type":"application/
    json","Content-Length":str(len(data))}
    for key,val in requestHeaders.iteritems(): # Set the headers
        req.add_header(key,val)
    req.add_data(data) # Add the data to the request
    # Make the request to ThingSpeak
    try:
        response = ul.urlopen(req) # Make the request
        print response.getcode() # A 202 indicates that the server has accepted the request
    except ul.HTTPError as e:
        print e.code # Print the error code
    messageBuffer = [] # Reinitialize the message buffer
    global lastConnectionTime
    lastConnectionTime = time.time() # Update the connection time

def getData():
    '''Function that returns the CPU temperature and percentage of CPU utilization'''
    cmd = '/opt/vc/bin/vcgencmd measure_temp'
    process = os.popen(cmd).readline().strip()
    cpuTemp = process.split('=')[1].split("'")[0]
    cpuUsage = psutil.cpu_percent(interval=2)
    return cpuTemp,cpuUsage
```

```python
def updatesJson():
    '''Function to update the message buffer
    every 15 seconds with data. And then call the httpRequest
    function every 2 minutes. This examples uses the relative timestamp as it uses the "delta_t" parameter.
    If your device has a real-time clock, you can also provide the absolute timestamp using the "created_at"
    parameter.
    '''

    global lastUpdateTime
    message = {}
    message['delta_t'] = int(round(time.time() - lastUpdateTime))
    Temp,Usage = getData()
    message['field1'] = Temp
    message['field2'] = Usage
    global messageBuffer
    messageBuffer.append(message)


# If posting interval time has crossed 2 minutes update the ThingSpeak channel with your data
if time.time() - lastConnectionTime >= postingInterval:
        httpRequest()
        lastUpdateTime = time.time()

if __name__ == "__main__":  # To ensure that this is run directly and does not run when imported
    while 1:
        # If update interval time has crossed 15 seconds update the message buffer with data
        if time.time() - lastUpdateTime >= updateInterval:
            updatesJson()

            httpRequest()
```

HOGESCHOOL ROTTERDAM

OPDRACHT WEEK 6:

- Create a repository entry in GitHub where you describe al the step necessary to create a Raspberry Pi based IoT project using ThinkSpeak as middleware

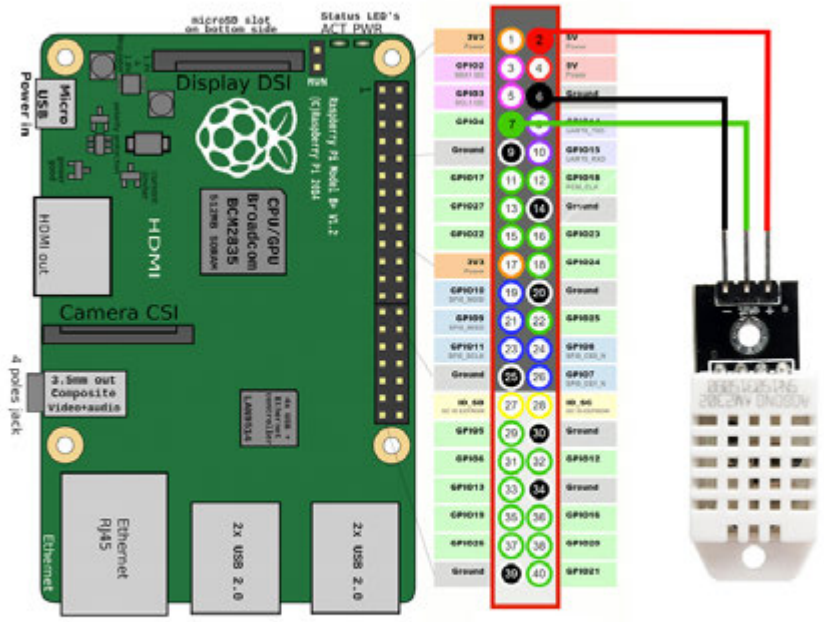
- Send {your sensor} data to ThingSpeak via your own public Channel as described in this tutorial, provide evidence on Git

Raspberry Pi Sensor to ThingsBoard using Constrained Application Protocol

HOGESCHOOL ROTTERDAM

Installation

Connecting the sensor



To connect the DHT22 sensor to the RPi:

1. Connect the `-` output on the sensor the pin 6 on the RPi.
2. Connect the `+` input on the sensor to pin 2 on the RPi.
3. Connect the `output` pin on the sensor to pin 7 on the RPi.

https://github.com/Silver292/rpi-coap

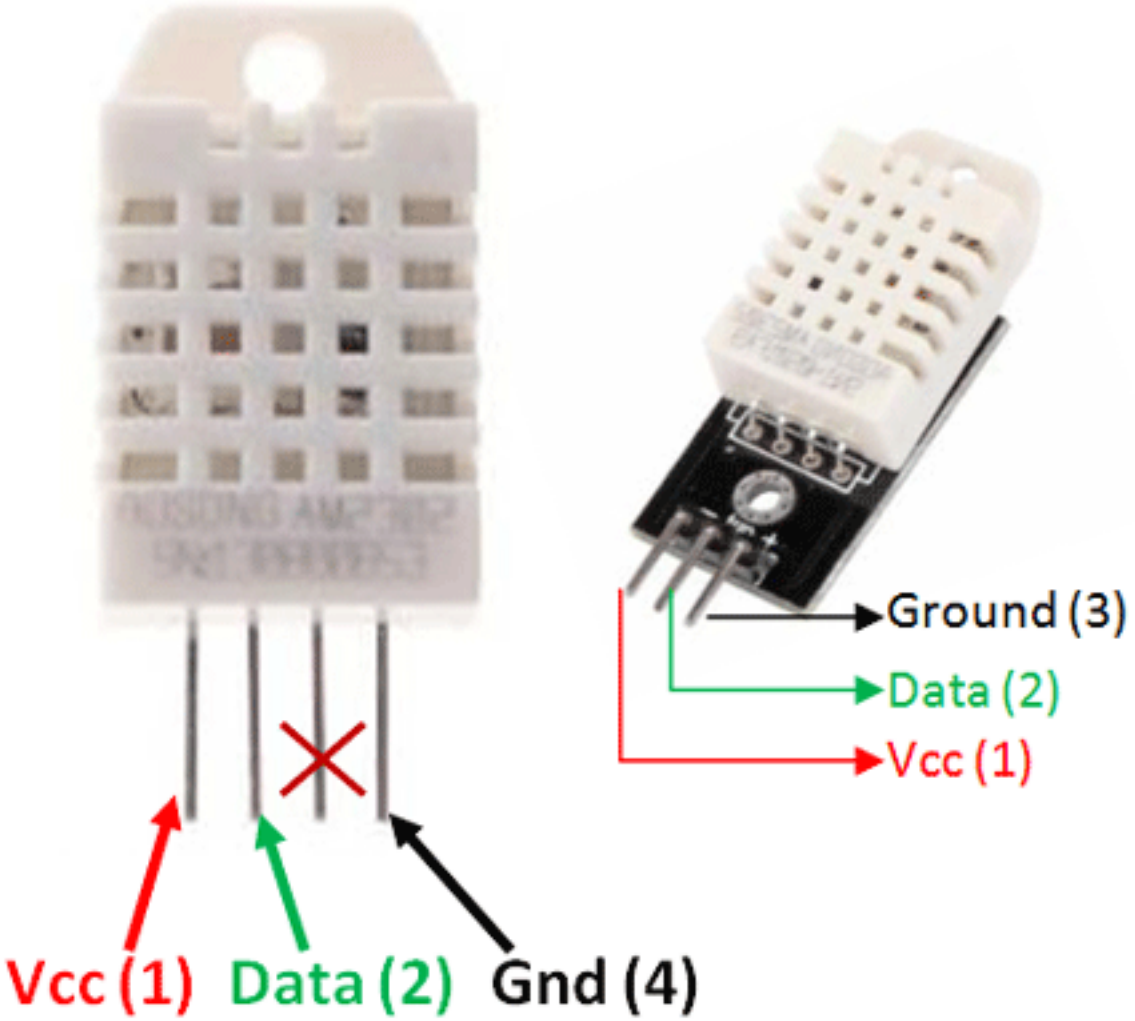
Introduction

This script is designed to be run on Raspberry Pi 3 hardware running the Raspbian 9.6 operating system.
It is also assumed that a DHT22 temperature and humidity sensor is connected to the Raspberry Pi (RPi) using GPIO 4 pin.
The process of connecting the sensor is described here.

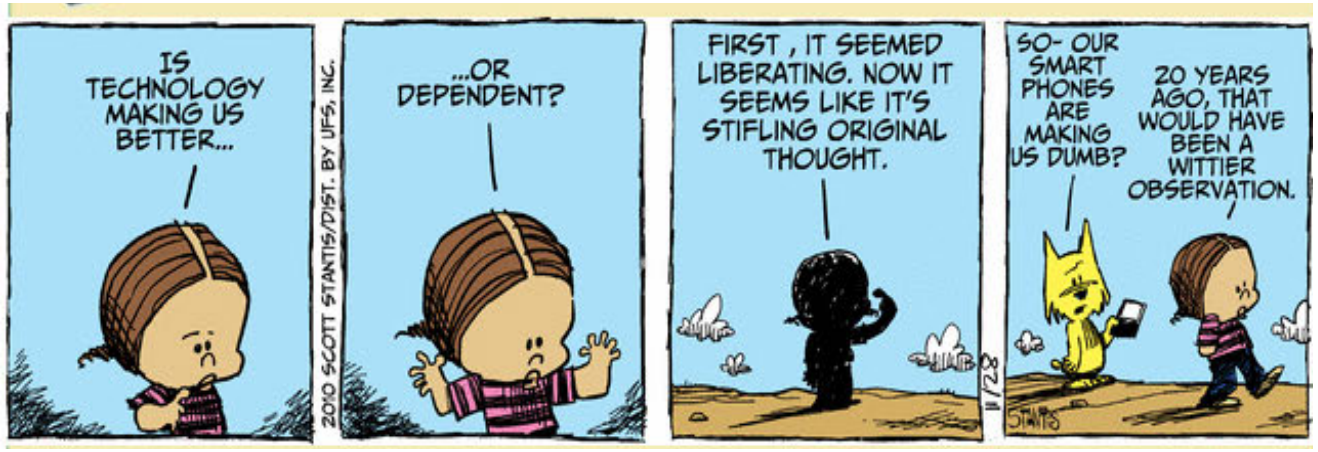
The script creates a Constrained Application Protocol (CoAP) client on the RPi and repeatedly polls the attached DHT22 sensor for temperature and humidity data.

This data is then formatted to JavaScript Object Notation (JSON) and is sent to a CoAP endpoint provided by the ThingsBoard Cloud Platform.

This data is then displayed on the ThingsBoard dashboard that shows the current temperature and humidity as well as historical readings in a graph form.


Ground (3)
Data (2)
Vcc (1)
Vcc (1)
Data (2)
Gnd (4)

HOGESCHOOL ROTTERDAM



This lesson was developed by:
Rob van der Willigen
CMD, Hogeschool Rotterdam
NOV 2019

http://creativecommons.org/licenses/by-nc-sa/3.0/

This lesson is licensed under a Creative Commons Attribution-Share-Alike license. You can change it, transmit it, show it to other people. Just always give credit to RFvdW.

Creative Commons License Types

	Can someone use it commercially?	Can someone create new versions of it?
Attribution	👍	👍
Share Alike	👍	Yup, AND they must license the new work under a Share Alike license.
No Derivatives	👍	👎
Non-Commercial	👎	Yup, AND the new work must be non-commercial, but it can be under any non-commercial license.
Non-Commercial Share Alike	👎	Yup, AND they must license the new work under a Non-Commercial Share Alike license.
Non-Commercial No Derivatives	👎	👎

SOURCE
http://www.masternewmedia.org/how-to-publish-a-book-under-a-creative-commons-license/

http://creativecommons.org/licenses/

http://empoweringthenatives.edublogs.org/2012/03/15/creative-commons-licenses/