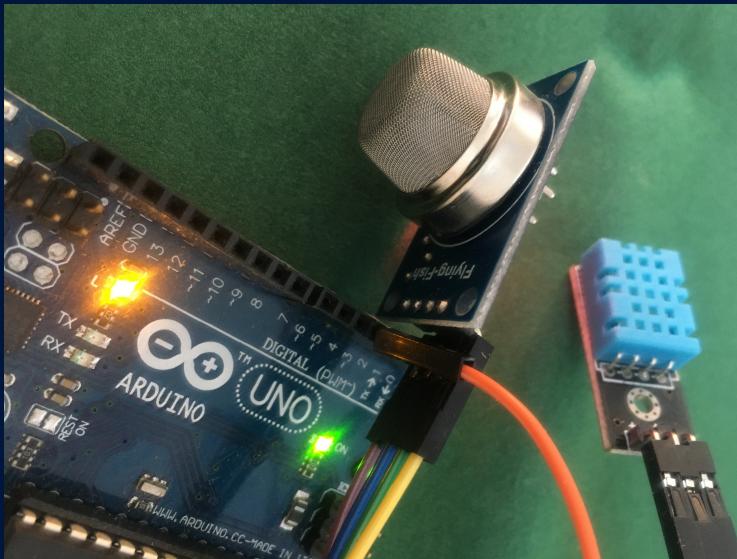


## COPENHAGEN BUSINESS ACADEMY



## DATA ENGINEERING



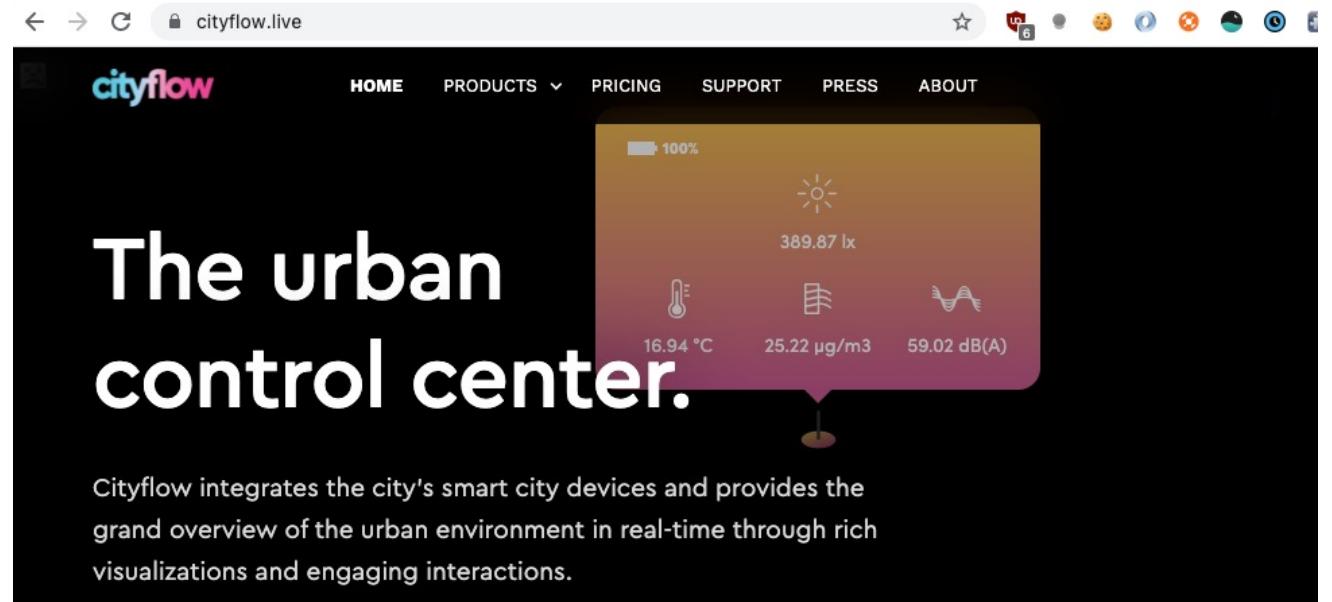
# FLOW 3 – Data Engineering

- Oplæg om de to cases i Flow 3
  - cityflow.dk – API
    - Json-dataformat
    - https request
    - python request
    - File-persistence (json, csv)
    - Database-persistence (MongoDB)
  - Bilbasen.dk – WebScraping
    - HTML/CSS/JavaScript primer
    - Scrape static (python bs4)
    - Scrape dynamic (python selenium)
- Tooling
  - GitHub / GitBash
  - Python (Anaconda) & jupyter notebook
  - AWS EC2-instance (Ubuntu med MongoDB og MySQL)
  - Arduino with serial-port driver

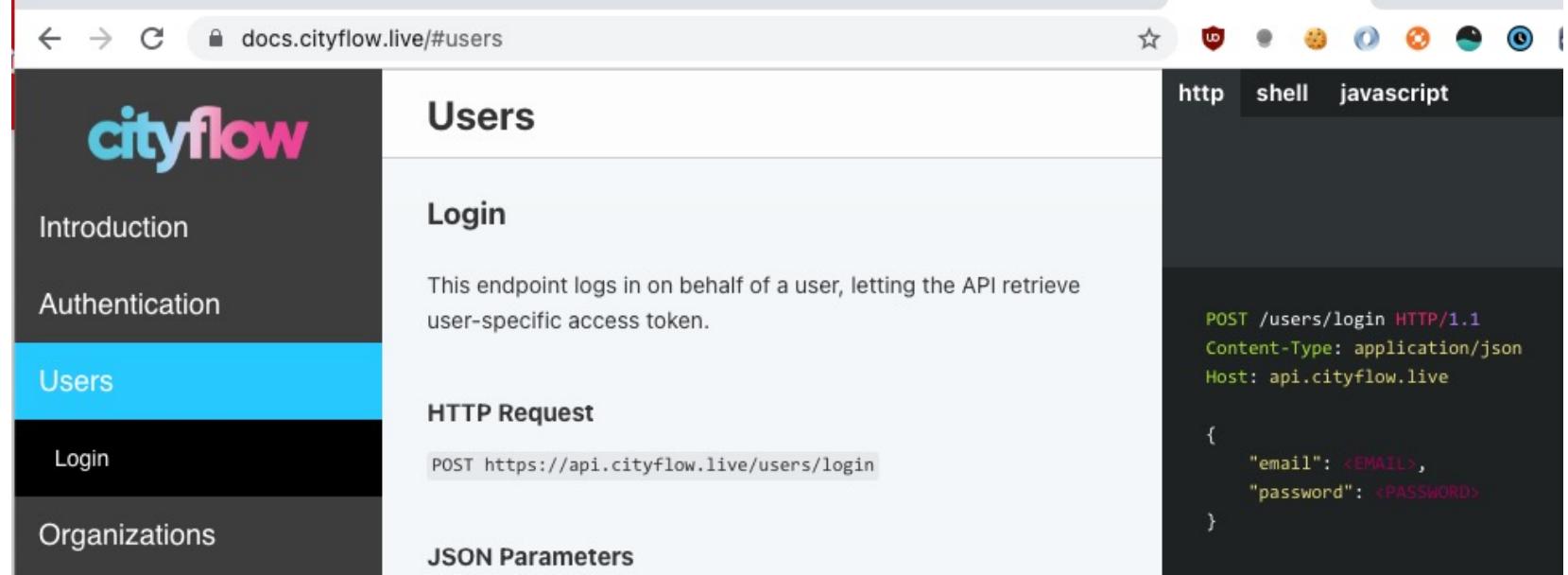
## FLOW 3 – User Stories

- Som underviser vil jeg gerne vide hvor meget den hvide bygning bliver brugt så jeg kan lægge aktiviteter i bygningen når den bliver mindst brugt
- Som studerende vil jeg gerne vide hvor det er bedst at lave en walk-and-talk i Århus i løbet af dagen på en hverdag
- Som ejer af en gammel Volvo vil jeg gerne kunne få en vurdering af hvor meget den er værd således at jeg kan planlægge hvornår jeg skal sælge den

# Cityflow



The screenshot shows the Cityflow homepage. At the top, there's a navigation bar with links for HOME, PRODUCTS (with a dropdown arrow), PRICING, SUPPORT, PRESS, and ABOUT. Below the navigation is a large, semi-transparent dark overlay containing the text "The urban control center." in white. To the right of this text is a rounded rectangular callout box with a pink-to-purple gradient background. Inside the callout box are several icons and data points: a battery icon at 100%, a sun icon, a temperature icon showing 16.94 °C, a light icon showing 389.87 lx, a pollution icon showing 25.22 µg/m³, and a sound icon showing 59.02 dB(A). Below the callout box is a descriptive paragraph: "Cityflow integrates the city's smart city devices and provides the grand overview of the urban environment in real-time through rich visualizations and engaging interactions."



The screenshot shows the Cityflow API documentation page for the "/users" endpoint. On the left, there's a sidebar with navigation links: Introduction, Authentication, **Users**, Login, and Organizations. The "Users" link is highlighted with a blue background. The main content area has a title "Users" and a sub-section "Login". It describes the endpoint: "This endpoint logs in on behalf of a user, letting the API retrieve user-specific access token." Below this is an "HTTP Request" section with a "POST" request example: `POST https://api.cityflow.live/users/login`. To the right of the main content is a sidebar with tabs: http, shell, and javascript. The "http" tab is selected. It contains a code block for a POST request to the "/users/login" endpoint:

```
POST /users/login HTTP/1.1
Content-Type: application/json
Host: api.cityflow.live

{
  "email": <EMAIL>,
  "password": <PASSWORD>
}
```

# Cityflow – REST API



- What do I see at [docs.cityflow.live](https://docs.cityflow.live)?
- Authentication
- Devices
- Measurements
- Errors
- http
- shell
- javascript

# Cityflow – REST API Data Format

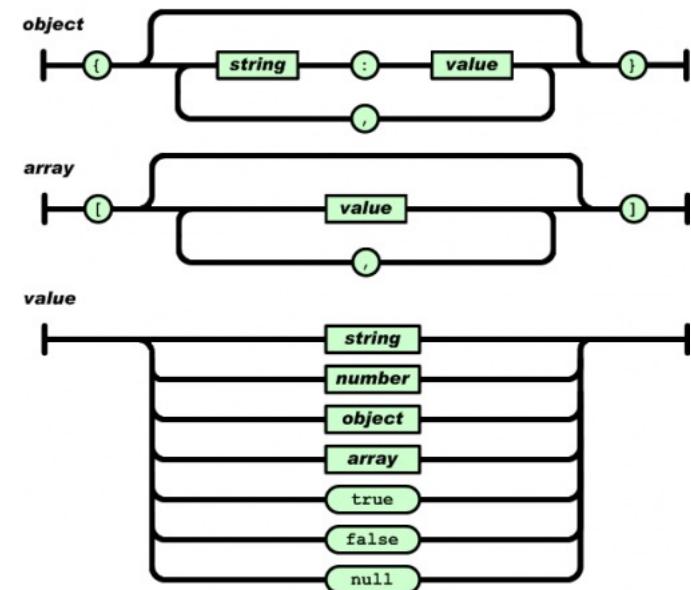
- “The above command returns JSON structured like this”
- JSON – JavaScript Object Notation

JSON    JSONLint - The JSON Validator

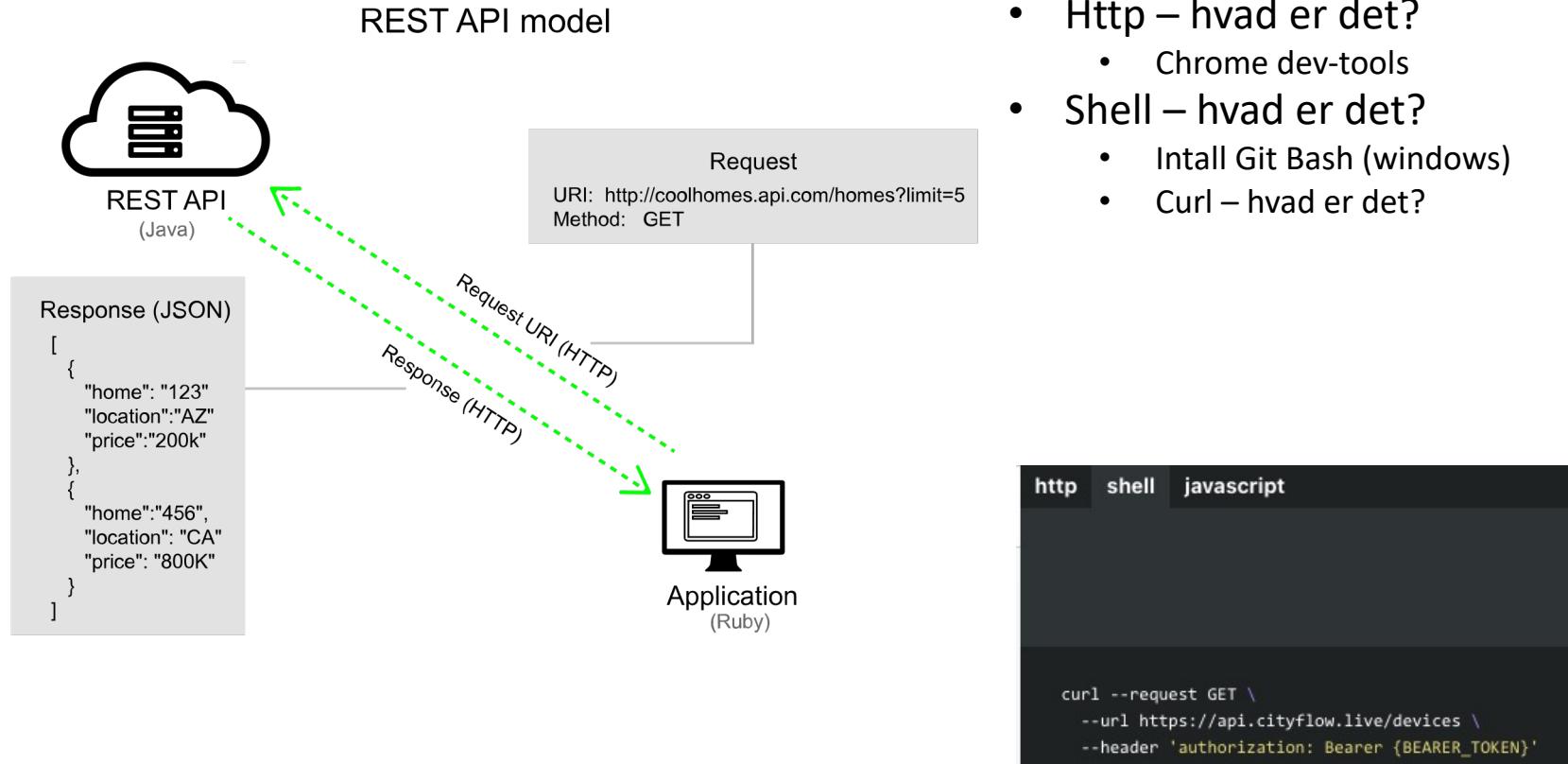
```

1 v  {
2     "id": "e00fce68c573b4acca2089ce",
3     "type": 150,
4     "location": 216,
5     "latitude": 56.1632767,
6     "longitude": 10.2105122,
7     "location_name": "Nørrebrogade",
8     "city": "Aarhus",
9     "country": "Denmark",
10    "roles": [
11        4
12    ],
13    "permissions": [],
14    "tags": [
15        "Randersvej"
16    ]
17 }

```



# Cityflow – REST API Request



- **Http – hvad er det?**
  - Chrome dev-tools
- **Shell – hvad er det?**
  - Intall Git Bash (windows)
  - Curl – hvad er det?

# Cityflow – REST API Request

1. Introduction
2. Protocols
3. Data Formats
4. Authentication, Part 1
5. Authentication, Part 2
6. API Design
7. Real-Time Communication
8. Implementation

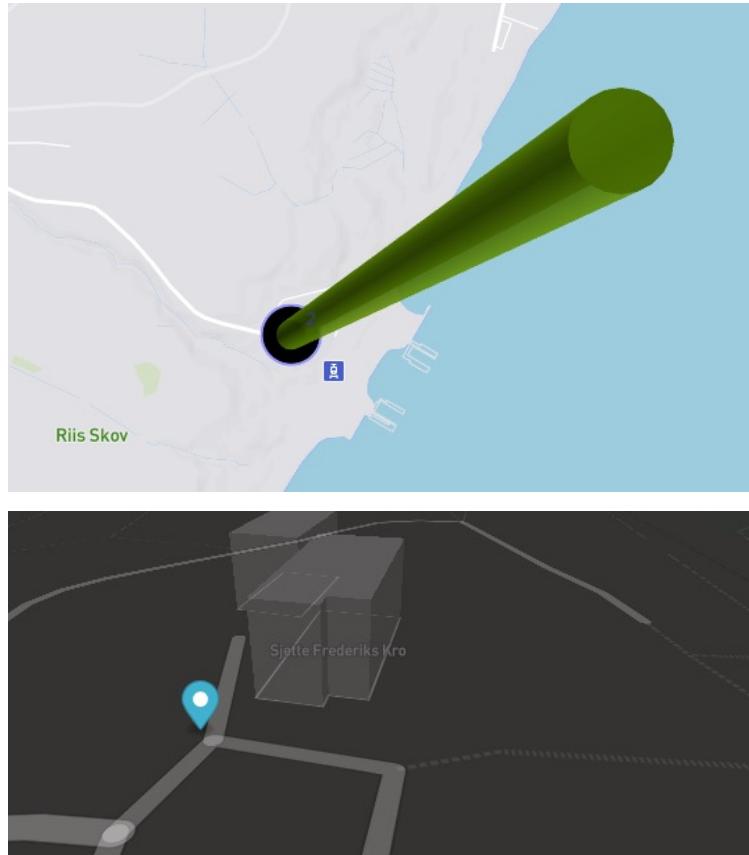
To make a valid request, the client needs to include four things:

- 1 URL (Uniform Resource Locator) [1](#)
- 2 Method
- 3 List of Headers
- 4 Body

| Navn   | Headere   | Forhåndsvisning | Svar | Iværksætter | Timing |
|--|---|-----------------|------|-------------|--------|
| <input type="checkbox"/> v1?sku=101gqxfpJUH9E&access_token=pk.eyJ1Ijoj |   |                 |      |             |        |
| <input checked="" type="checkbox"/> login                              | <b>Anmodningsheadere</b>                                      |                 |      |             |        |
| <input type="checkbox"/> login   | :authority: api.cityflow.live                                 |                 |      |             |        |
| <input type="checkbox"/> getUserDefaultLocation                        | :method: OPTIONS  |                 |      |             |        |
| <input type="checkbox"/> types   | :path: /users/login   |                 |      |             |        |
| <input type="checkbox"/> devices                                       | :scheme: https  |                 |      |             |        |
| <input type="checkbox"/> organizations                                 | accept: */*   |                 |      |             |        |
| <input type="checkbox"/> latest  | accept-encoding: gzip, deflate, br                            |                 |      |             |        |
| <input type="checkbox"/> pina  | accept-language: da-DK,da;q=0.9,en-US;q=0.8,en;q=0.7,de;q=0.6 |                 |      |             |        |
|  | access-control-request-headers: authorization,content-type    |                 |      |             |        |
|  | access-control-request-method: POST                           |                 |      |             |        |
| 43 anmodninger   1.2 MB blev overført   6.1 MB ressourcer              |   |                 |      |             |        |

| Navn   | Headere                                    | Forhåndsvisning | Svar      | Iværksætter | Tid |
|--|--|-----------------|-----------|-------------|-----|
| <input checked="" type="checkbox"/> runEnv.sh?start=rонне&destsel=ronne&date=&afg=.. | <b>Parametre for forespørgelsesstrenge</b> | se kilde        | vis indl. |             |     |
| <input type="checkbox"/> Logger.js   | start: ронне                               |                 |           |             |     |
| <input type="checkbox"/> jquery.min.js   | destsel: ronnie                            |                 |           |             |     |
| <input type="checkbox"/> SmartBomb.js  | date:                                      |                 |           |             |     |
| <input type="checkbox"/> ReferrerMonitor.js  | afg: Kurt                                  |                 |           |             |     |
| <input type="checkbox"/> CoreAPI.js  | dest: Verner                               |                 |           |             |     |
| <input type="checkbox"/> API.js  | hobby: ridning                             |                 |           |             |     |
| <input type="checkbox"/> ChromeAPI.js  | gender: F                                  |                 |           |             |     |
| <input type="checkbox"/> ComponentFactory.js   | mus: mus                                   |                 |           |             |     |
| 153 anmodninger   892 kB blev overført   1.1 MB ressourcer                           |  |                 |           |             |     |

# Cityflow – REST API Get Data I

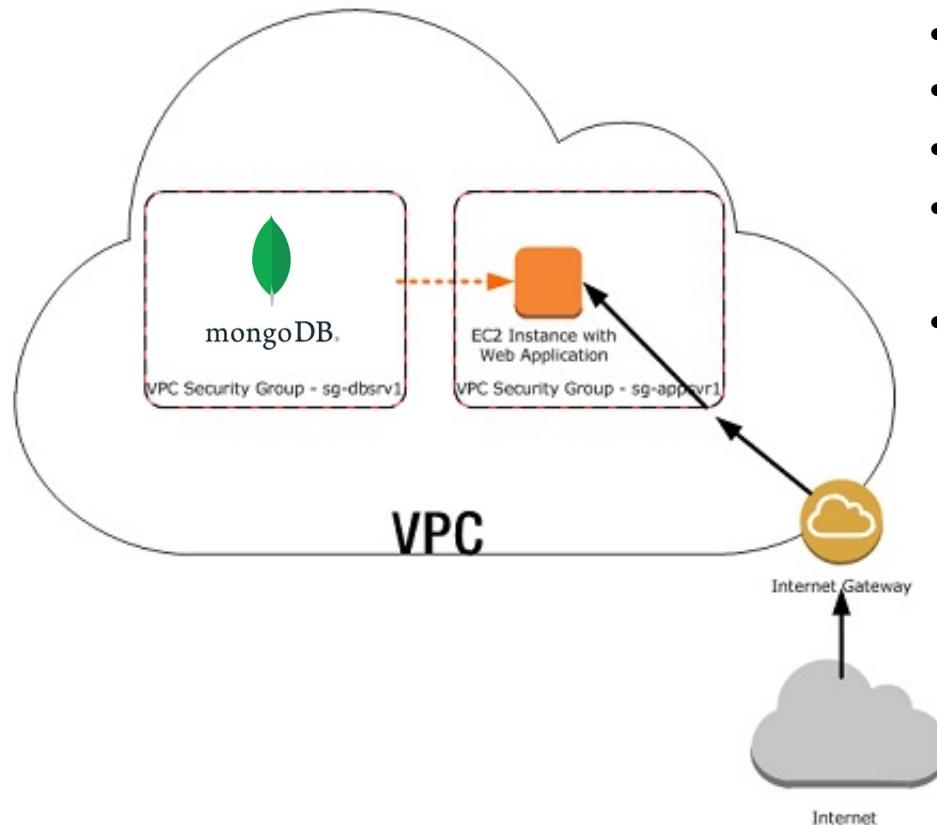


```
http shell javascript

GET /devices HTTP/1.1
Authorization: Bearer {BEARER_TOKEN}
Host: api.cityflow.live

{
  id: e00fce689f02a96799f34fc2,
  type: 150,
  location: 190,
  latitude: 56.1770897,
  longitude: 10.2296247,
  location_name: Salonvejen,
  city: Risskov,
  country: Denmark,
  roles: [
    4
  ],
  permissions: [],
  tags: [
    Risskov
  ]
},
```

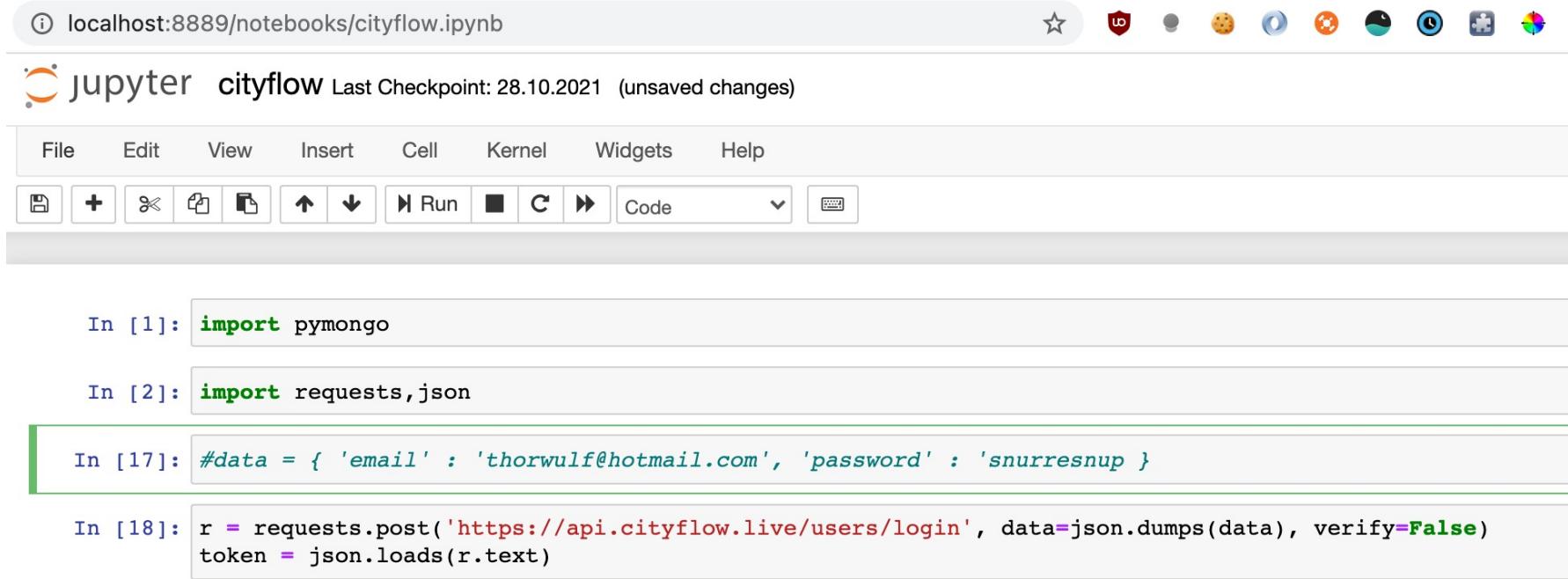
# Cityflow – Save data Infrastructure



- Create EC2 instance (ubuntu)
- Get ssh-access
- Install mongodb
- Client access
  - Robo 3T
- Dev access
  - pymongo

# Cityflow – python from jupyter notebook

localhost:8889/notebooks/cityflow.ipynb



jupyter cityflow Last Checkpoint: 28.10.2021 (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

File Cell Kernel Help

In [1]: `import pymongo`

In [2]: `import requests,json`

In [17]: `#data = { 'email' : 'thorwulf@hotmail.com', 'password' : 'snurresnup' }`

In [18]: `r = requests.post('https://api.cityflow.live/users/login', data=json.dumps(data), verify=False)`  
`token = json.loads(r.text)`

```
<class 'requests.models.Response'>
```

# Cityflow – opgave: udfyld skabelonen

importer de nødvendige moduler

In [ ]:

lav en post-request til <https://api.cityflow.live/users/login>

In [ ]:

lav en get-request der henter listen af devices

In [ ]:

lav en get-request der henter de nyeste målinger <https://api.cityflow.live/measurements/latest> Du gemmer resultatet af din request i en variabel som så er af type response-objekt.

In [ ]:

Du henter nu en datastruktur ud af response-objektet (response.json()) Undersøg strukturen af de returnerede data så du kan nå ind til listen af målinger Du får nemlig noget a la { '150': [ {'aPS': 0.69, 'b': 100,

In [ ]:

Nu kan du hente pretty-print så dit resultat står pænt

In [ ]:

```
import pprint as pp
dineMeasures=[]
pp.pprint(dineMeasures)
```

Nu looper du igennem din liste af målinger vha et for-loop

In [ ]:

```
for item in dineMeasures:
    pp.pprint(item)
```

# Cityflow – opgave: Antal SAS-fly i luften

jupyter openskyTask Last Checkpoint: 12 minutter siden (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 O

Gå til <https://opensky-network.org/apidoc/> Find via api-dokumentationen ud af hvordan du laver en get-request med en liste af alle fly. Filtrér nu listen så du kan svare på følgende spørgsmål: Hvor mange SAS-fly er der i luften lige nu?

step1: find urlen  
step2: få en respons fra dit request  
step3: hent data-strukturen ud fra respons.json() (hvis det virker)  
step4: undersøg datastrukturen (tjek api-dokumentationen for de enkelte felter)  
step5: filtrér data-sættet

In [ ]:

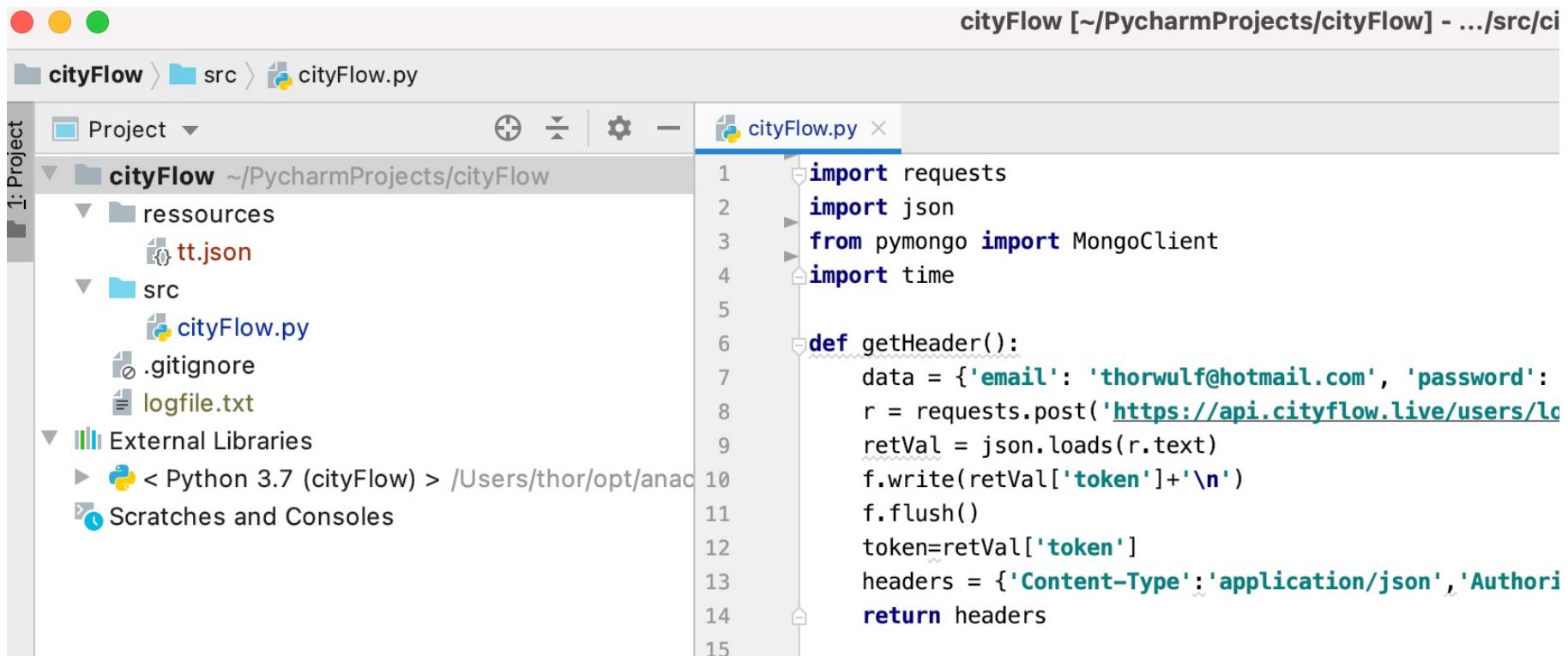
# Cityflow – python from jupyter notebook

<class 'requests.models.Response'>

## Properties and Methods

| Property/Method       | Description  |
|-----------------------|--|
| apparent_encoding     | <a href="#">Try it</a> Returns the apparent encoding   |
| close()               | <a href="#">Try it</a> Closes the connection to the server   |
| content               | <a href="#">Try it</a> Returns the content of the response, in bytes   |
| cookies               | <a href="#">Try it</a> Returns a CookieJar object with the cookies sent back from the server                                     |
| elapsed               | <a href="#">Try it</a> Returns a timedelta object with the time elapsed from sending the request to the arrival of the response  |
| encoding              | <a href="#">Try it</a> Returns the encoding used to decode r.text  |
| headers               | <a href="#">Try it</a> Returns a dictionary of response headers  |
| history               | <a href="#">Try it</a> Returns a list of response objects holding the history of request (url)                                   |
| is_permanent_redirect | <a href="#">Try it</a> Returns True if the response is the permanent redirected url, otherwise False                             |
| is_redirect           | <a href="#">Try it</a> Returns True if the response was redirected, otherwise False  |
| iter_content()        | <a href="#">Try it</a> Iterates over the response  |
| iter_lines()          | <a href="#">Try it</a> Iterates over the lines of the response   |
| json()                | <a href="#">Try it</a> Returns a JSON object of the result (if the result was written in JSON format, if not it raises an error) |
| links                 | <a href="#">Try it</a> Returns the header links  |
| next                  | <a href="#">Try it</a> Returns a PreparedRequest object for the next request in a redirection                                    |
| ok                    | <a href="#">Try it</a> Returns True if status_code is less than 400, otherwise False   |
| raise_for_status()    | <a href="#">Try it</a> If an error occur, this method returns a HTTPError object   |
| reason                | <a href="#">Try it</a> Returns a text corresponding to the status code   |
| request               | <a href="#">Try it</a> Returns the request object that requested this response   |
| status_code           | <a href="#">Try it</a> Returns a number that indicates the status (200 is OK, 404 is Not Found)                                  |
| text                  | <a href="#">Try it</a> Returns the content of the response, in unicode   |
| url                   | <a href="#">Try it</a> Returns the URL of the response   |

# Cityflow – CRUD from python IDE



cityFlow [~/PycharmProjects/cityFlow] - .../src/ci

cityFlow ~/PycharmProjects/cityFlow

ressources  
tt.json

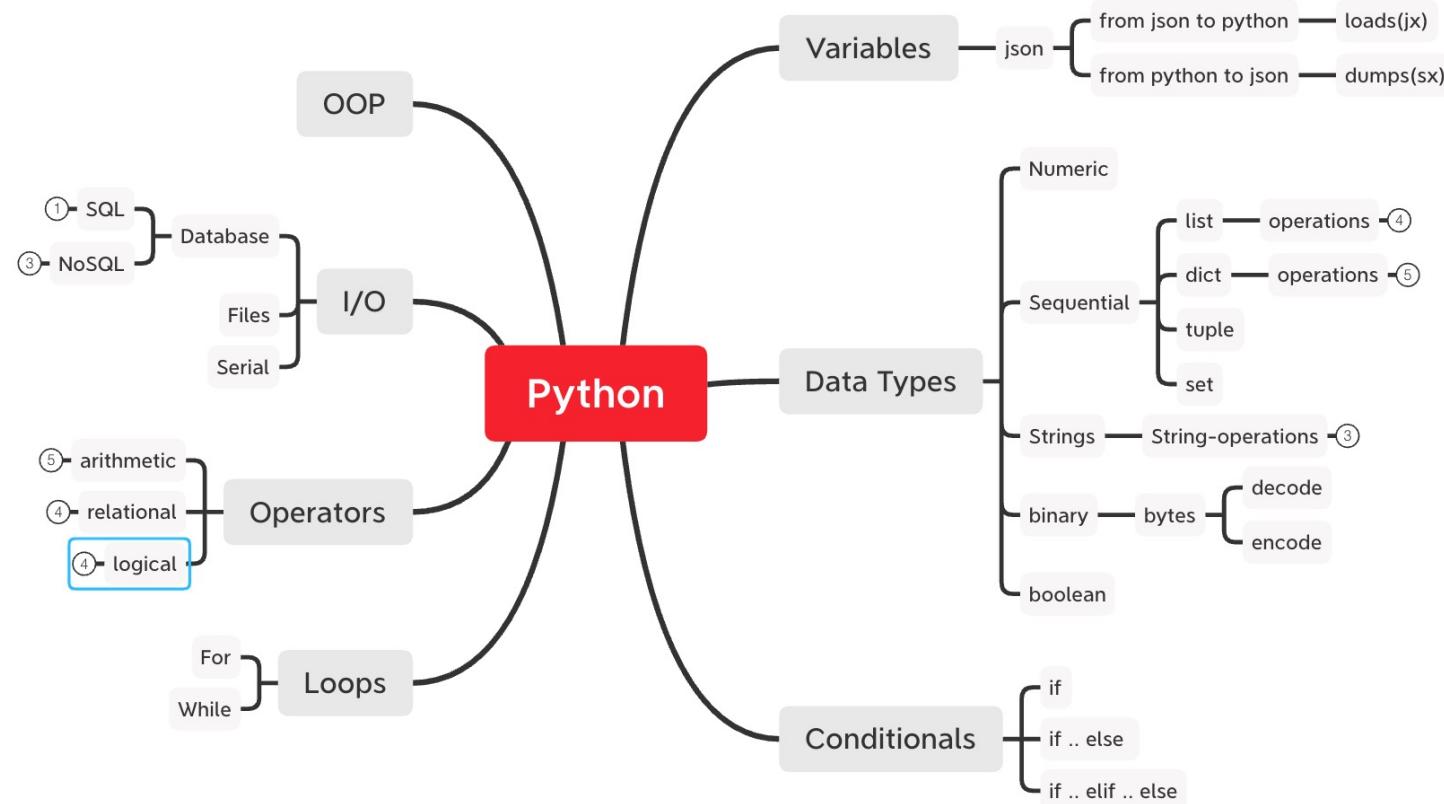
src  
cityFlow.py  
.gitignore  
logfile.txt

External Libraries  
> Python 3.7 (cityFlow) > /Users/thor/opt/anaconda3/lib/python3.7/site-packages

Scratches and Consoles

```
1 import requests
2 import json
3 from pymongo import MongoClient
4 import time
5
6 def getHeader():
7     data = {'email': 'thorwulf@hotmail.com', 'password': '123456'}
8     r = requests.post('https://api.cityflow.live/users/login')
9     retVal = json.loads(r.text)
10    f.write(retVal['token']+'\n')
11    f.flush()
12    token=retVal['token']
13    headers = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + token}
14    return headers
15
```

# Cityflow – python primer



# Cityflow – PyMongo CRUD primer

C

- insert()

R

- find()

U

- update()

D

- remove()

```
import pymongo
```

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
```

```
for x in mycol.find({}, { "address": 0 }):
    print(x)
```

```
import pymongo
```

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
```

```
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
```

```
for x in mycol.find():
    print(x)
```

```
import pymongo
```

```
myclient = pymongo.MongoClient("mongodb://localhost:27017/")
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
```

```
mydict = { "name": "John", "address": "Highway 37" }

x = mycol.insert_one(mydict)
```

```
import pymongo
```

```
myclient = pymongo.MongoClient("mongodb://",
mydb = myclient["mydatabase"]
mycol = mydb["customers"]
```

```
myquery = { "address": { "$gt": "S" } }
```

```
mydoc = mycol.find(myquery)
```

```
for x in mydoc:
    print(x)
```

# Cityflow – mongodb CRUD primer



# Cityflow – inspect data from Mongo



The screenshot shows a MongoDB shell interface. On the left is a sidebar with database and collection names: aws (4), System, admin, local, cityflow (with Collections (1) containing observations), Functions, and Users. The main area displays a command line with a query and its execution results.

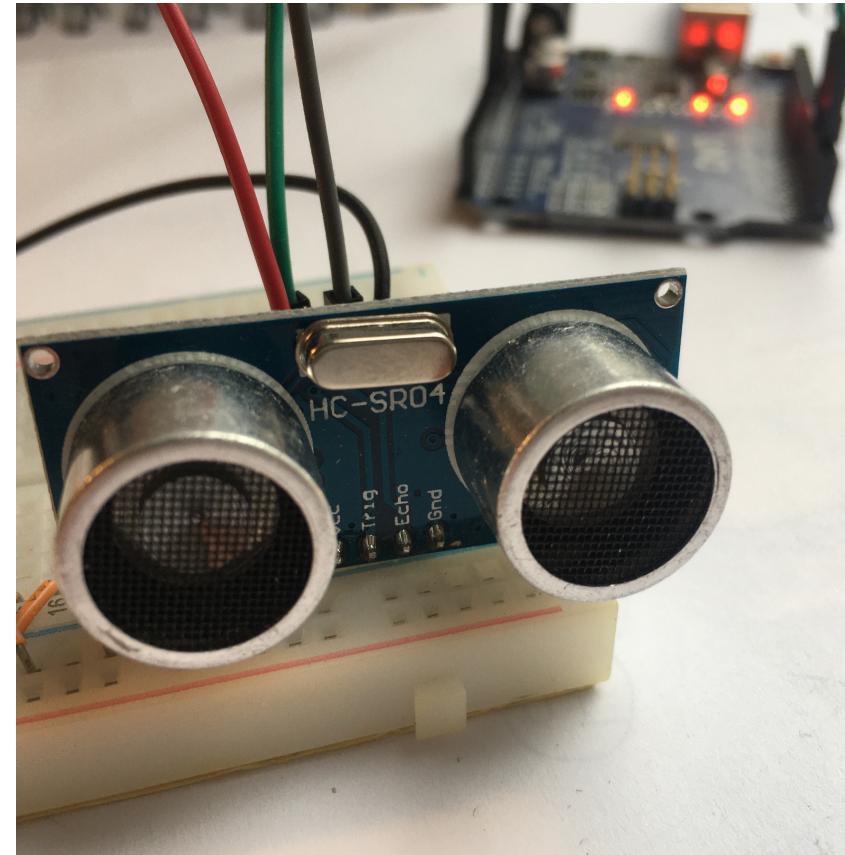
```
* db.getCollection('observations').find({device_id: "e00fce683182b4876452e2c7"}, {time:1, t:1, _id:0})  
aws 3.145.0.163:27017 cityflow  
db.getCollection('observations').find({device_id: "e00fce683182b4876452e2c7"}, {time:1, t:1, _id:0})  
observations 0.208 sec.  
Key | Value  
(1) | { 2 fields }  
time | 2021-11-04T12:01:02.000Z  
t | 8.84
```

# Cityflow – CRUD-operations

|   |            |
|---|------------|
| C | • insert() |
| R | • find()   |
| U | • update() |
| D | • remove() |

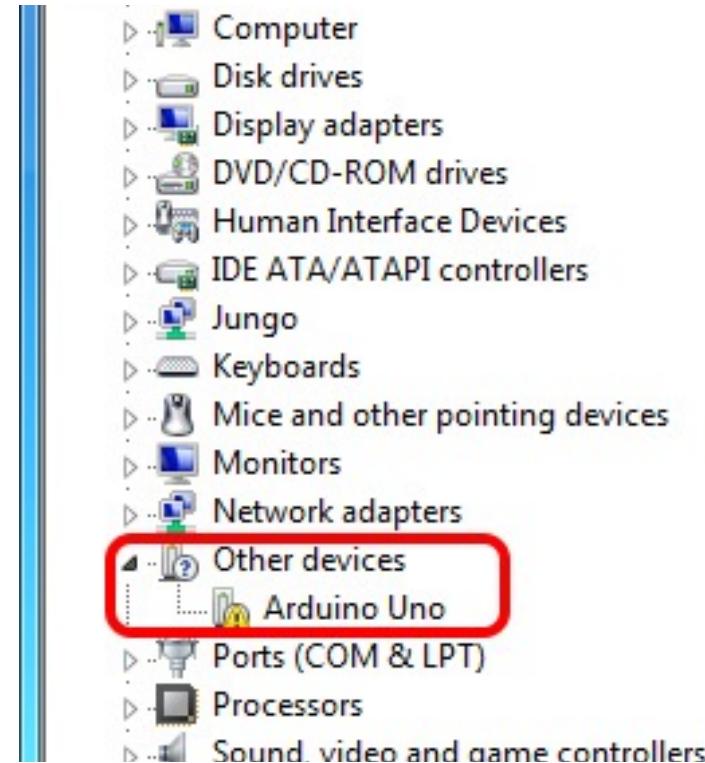
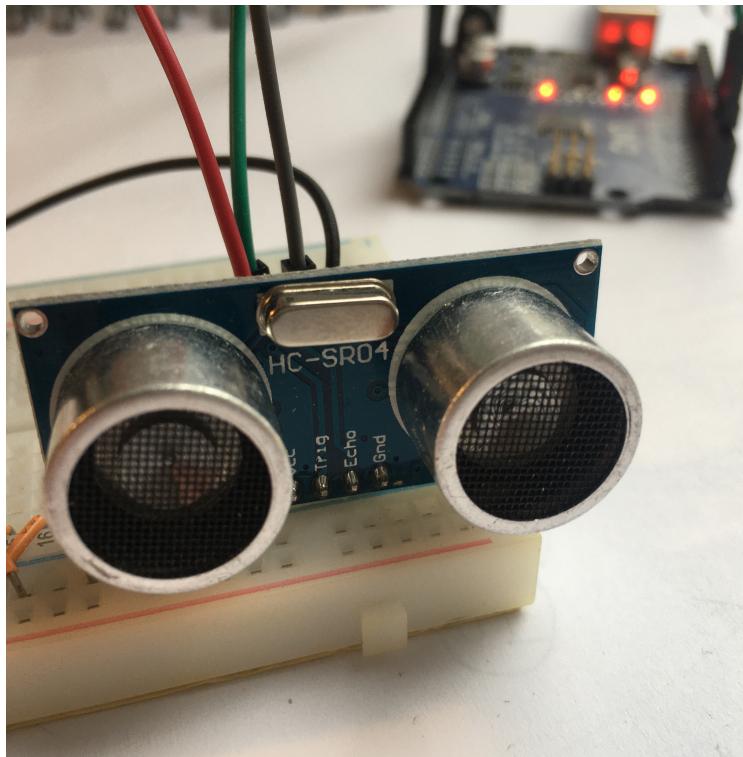
# Den hvide bygning – Data-retrieval

- Design af IO-device
- Kodning af device
  - Arduino
  - Python
- Analyse af data
- Design et API til forespørgslen: hvor mange er i bygningen <yyyy-mm-dd hh:mm>

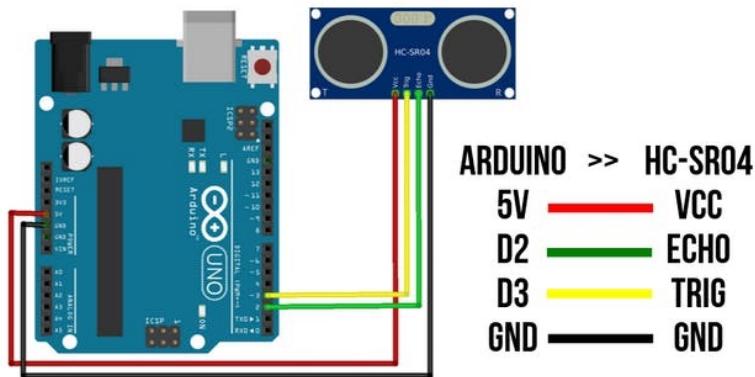


# Den hvide bygning – Data-retrieval via USB

31



# HC-SR04



- Installér Arduino
- Plug boardet
- Tjek at din device dukker op i enhedshåndteringen

# HC-SR04 - PySerial

localhost:8929/notebooks/pyserial.ipynb#

Jupyter pyserial (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [1]:

```
import serial
import serial.tools.list_ports
import pymongo
import time
```

In [13]:

```
serial_speed = 9600
#serial_port = '/dev/cu.usbmodem145101'
serial_port = '/dev/tty.usbserial-14610'
timeout = time.time() + 60*5 # 5 minutes from now
```

In [14]:

```
ports=serial.tools.list_ports.comports()
[print(port.name) for port in ports]
```

cu.Bluetooth-Incoming-Port  
cu.JabraElite75t-SPPDev  
cu.URT2  
cu.URT1  
cu.usbserial-14610

Out[14]: [None, None, None, None, None]

In [15]:

```
ser = serial.Serial(serial_port, 9600, timeout=1)
```