物聯網與微處理機系統設計
Internet of Things and Microprocessor System Design

Lecture 06 – Networks and Cloud

Lecturer: 陳彥安 Chen, Yan-Ann

YZU CSE

Outline

- Introduction
  - IoT Architecture
  - IoT Networks
  - IoT Platform
- Network-controlled LED
  - TCP socket server
  - HTTP server
- IoT Platform
  - Ubidots
- Lab
  - Remote switch

Outline

- Introduction
  - IoT Architecture
  - IoT Networks
  - IoT Platform
- Network-controlled LED
  - TCP socket server
  - HTTP server
- IoT Platform
  - Ubidots
- Lab
  - Remote switch

IoT Architecture (1/2)

Application

- Smart City
- Smart Home
- Intelligent Transportation
- Smart Retailing
- Smart Factory
- Smart Health Care
- Smart Building
- Intelligent Logistics
- Environmental Monitoring
- Public Safety

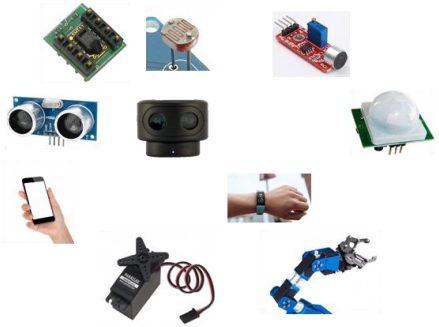
Network

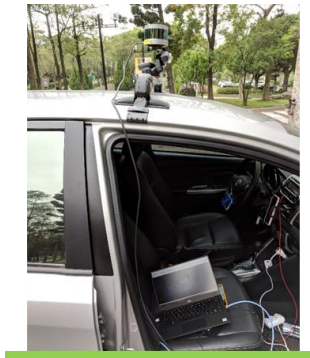
- Big-data Analysis
- Cloud Computing
- Internet
- Heterogeneous Networks
- WiFi
- ZigBee
- Bluetooth
- LoRa
- LTE
- 5G
- Ethernet
- Optical Fiber

Sensors / Actuators

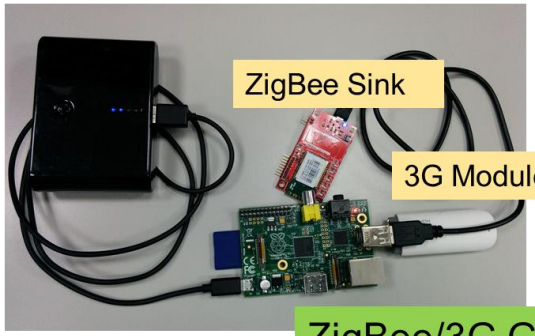
Sensing
- Environment: Temperature, Humidity, Light, PM2.5, CO_2
- Space: Pressure, distance, infrared
- Context: Acceleration, Air pressure
- Activity：smartphone, wearable device

Actuating
- Motor, Lighting, Display, Robot, Appliance

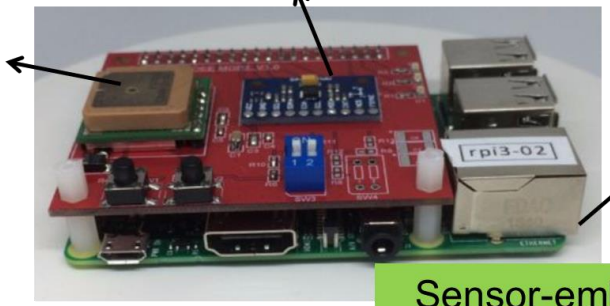




Intelligent Transportation

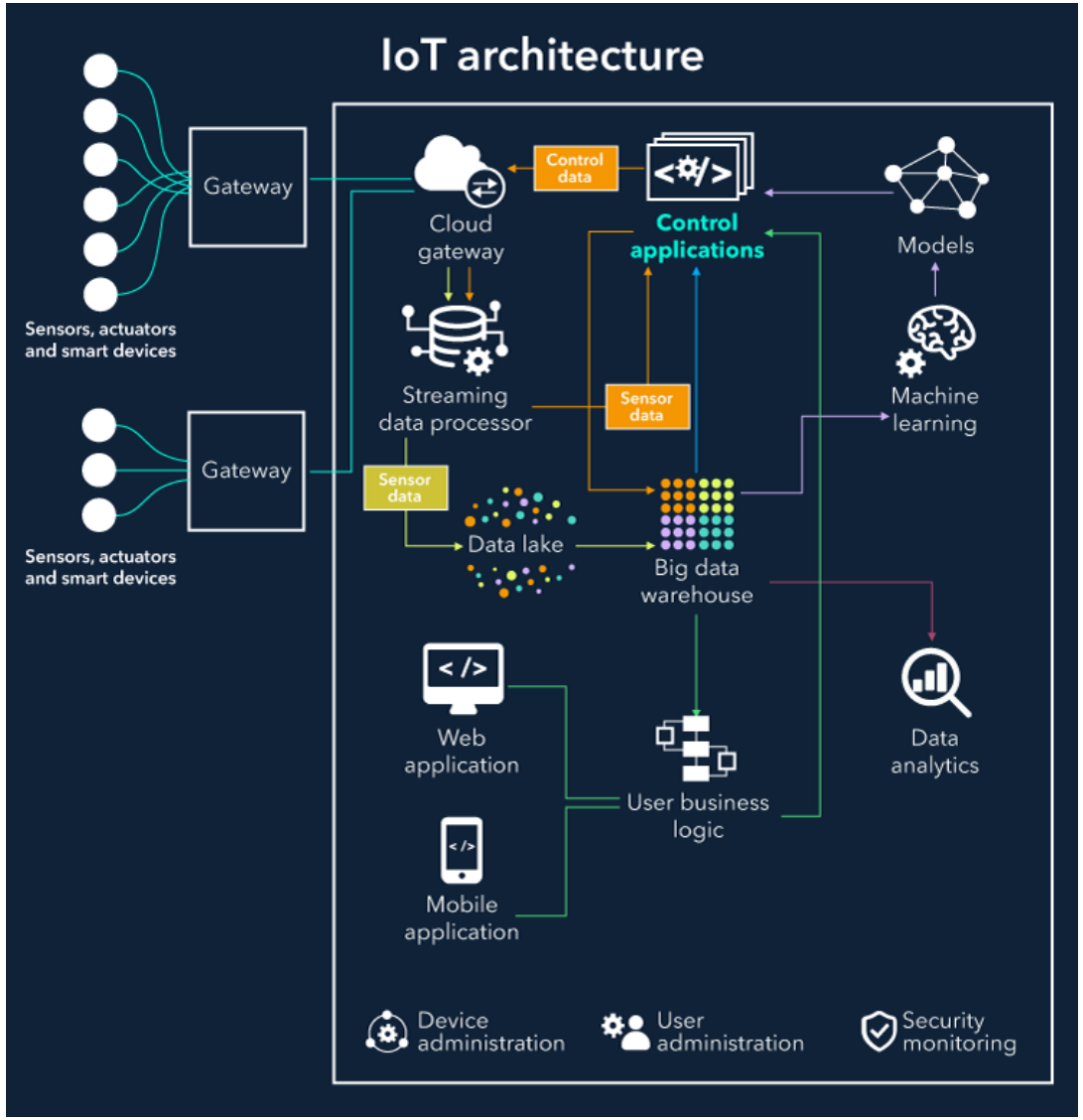


ZigBee/3G Gateway

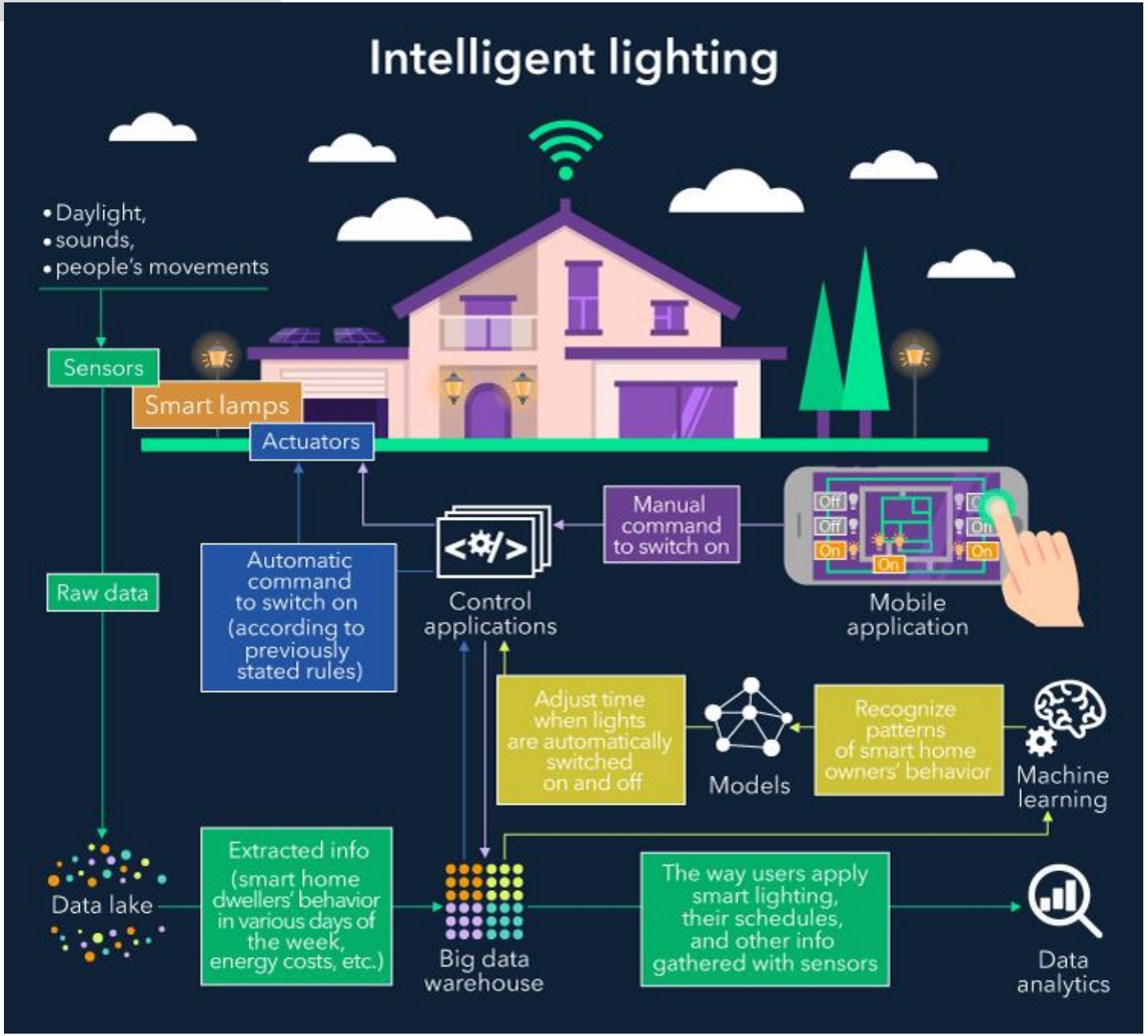


Sensor-embedded device

IoT Architecture (2/2)



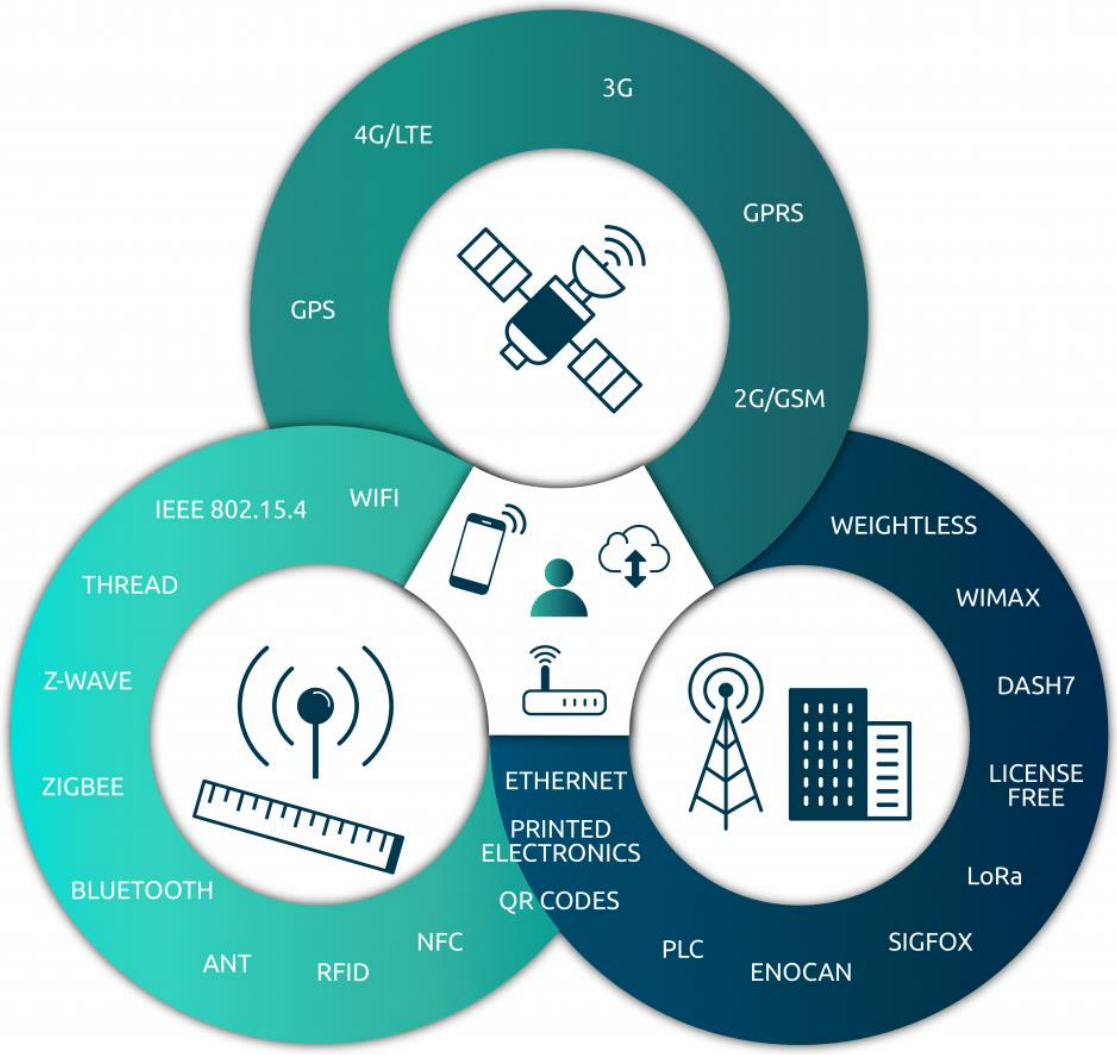
src: https://www.scnsoft.com/blog/iot-architecture-in-a-nutshell-and-how-it-works



src: https://www.scnsoft.com/blog/iot-architecture-in-a-nutshell-and-how-it-works

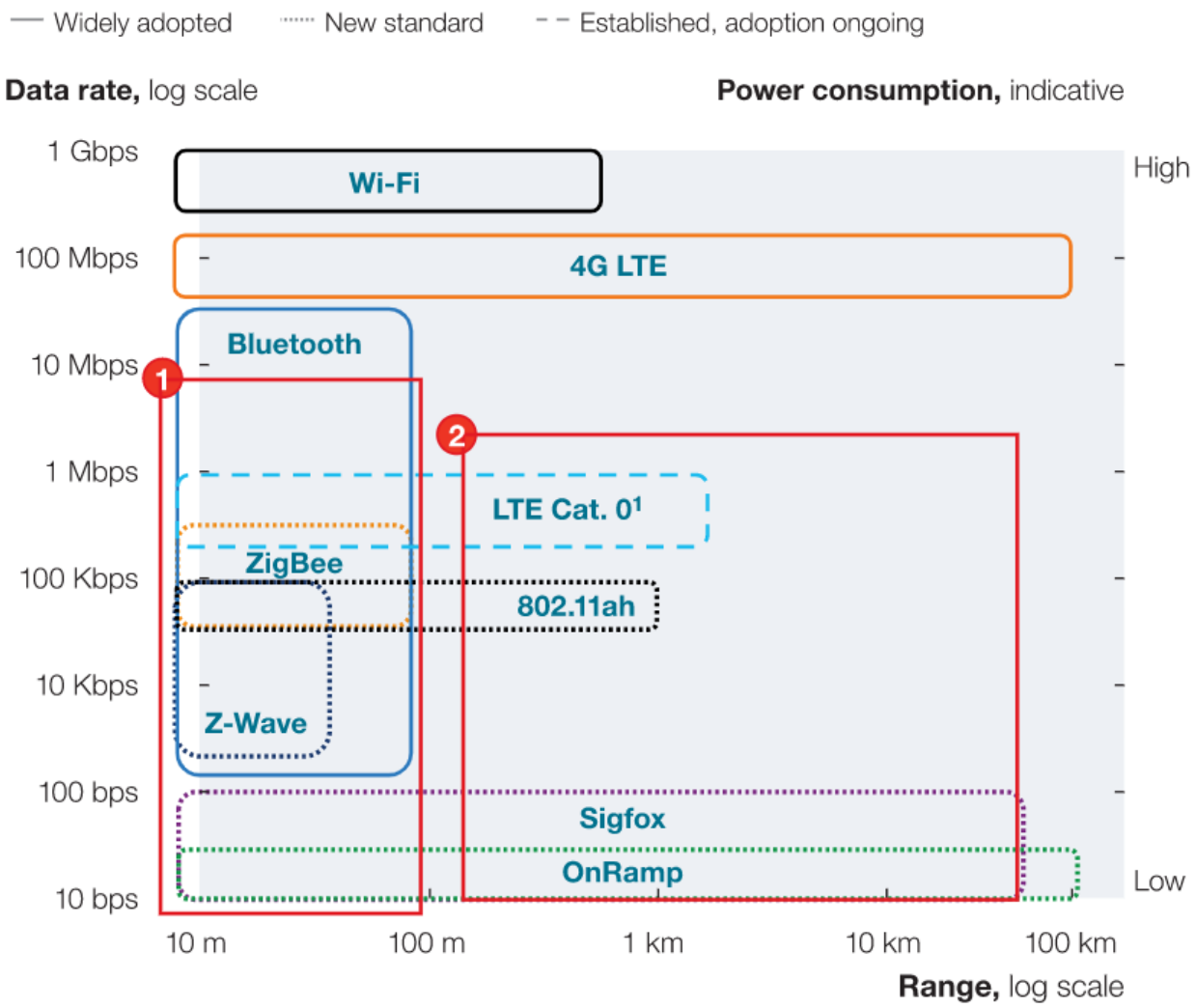
IoT Networks (1/2)

- Wireless networks
  - short-range
  - medium range
  - long range
- Category
  - LAN (Local Area Network)
  - LPWAN (Low power Wide Area Network
  - Cellular LPWAN (NB-IoT and LTE-M)
  - Cellular



src: https://axible.io/en/iot-networks-overview/

IoT Networks (2/2)



Standards for the Internet of Things (IoT) are not mature in many categories, including connectivity.

Widely adopted New standard Established, adoption ongoing

1. Many competing standards for low-range, medium-low data rate hinder growth for many IoT applications
   - Interoperability missing
   - Consortia wars might be emerging
   - Additional incompatibilities in higher communication layers, eg, 6LoWPAN vs ZigBee

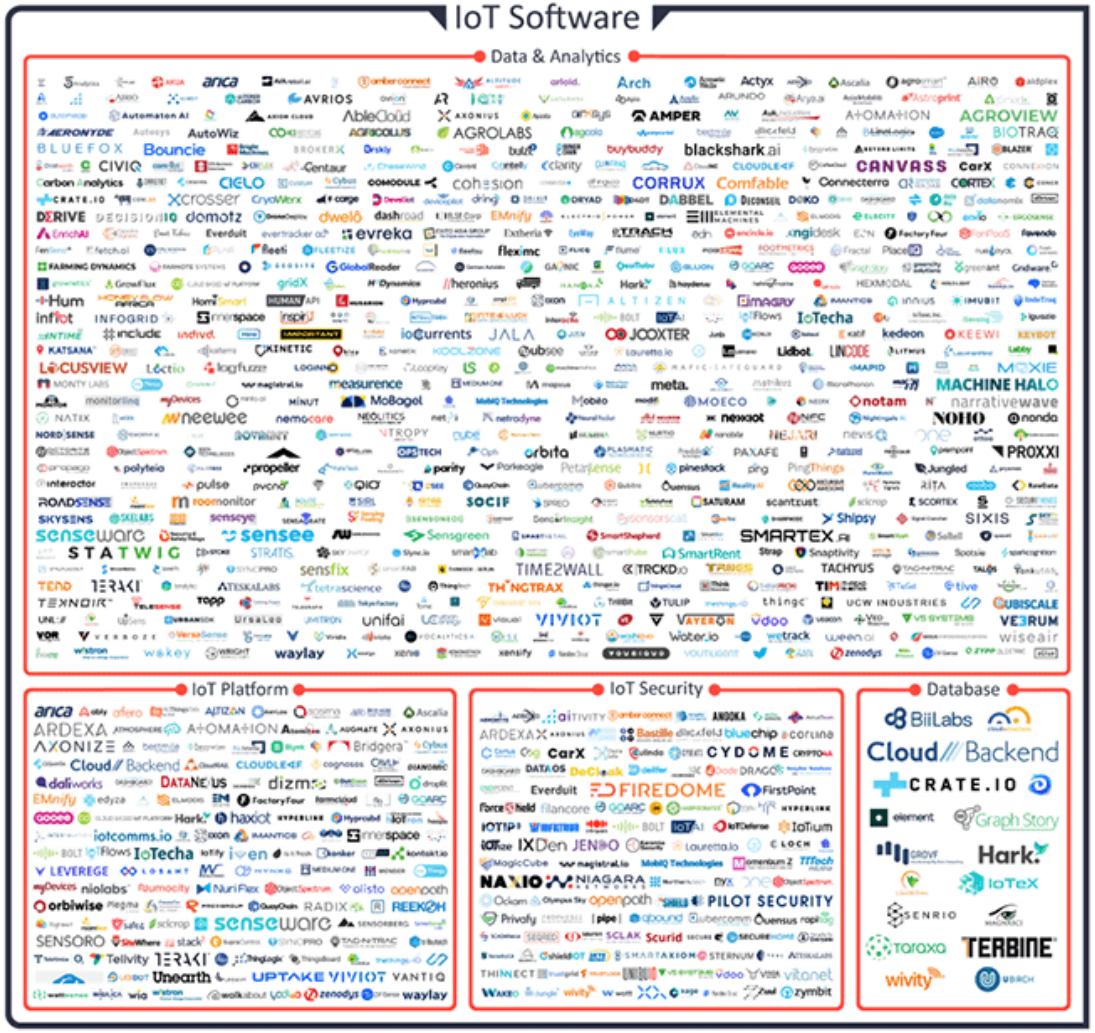
2. Standard white space for low-data-rate, low-power, high-range applications such as smart grid
   - Wi-Fi and LTE have high power consumption
   - Alternatives with low power and wide range (eg, LTE Cat. 0, 802.11ah, Sigfox, and OnRamp) are in very early stages and compete against each other

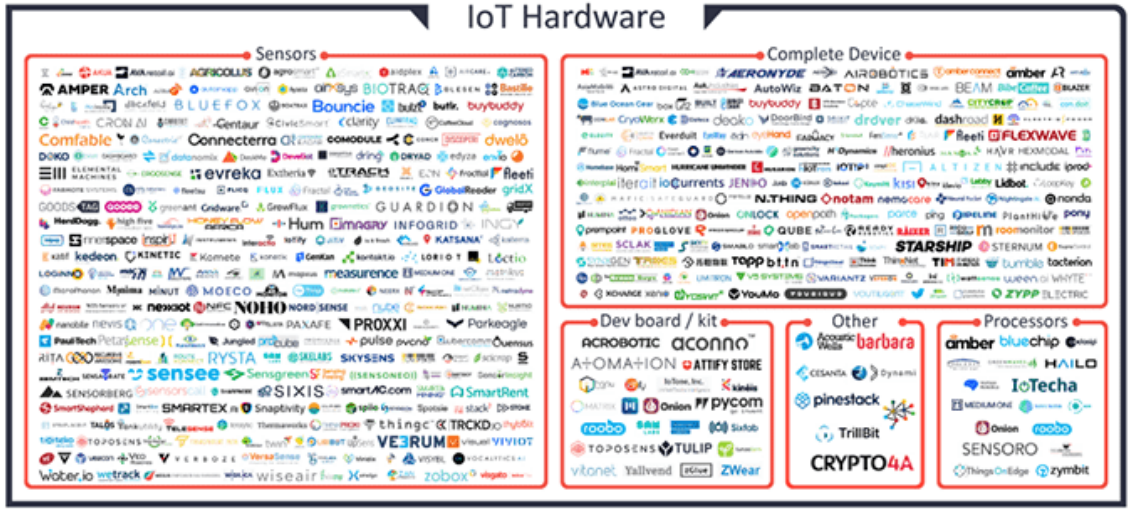
IOT ANALYTICS

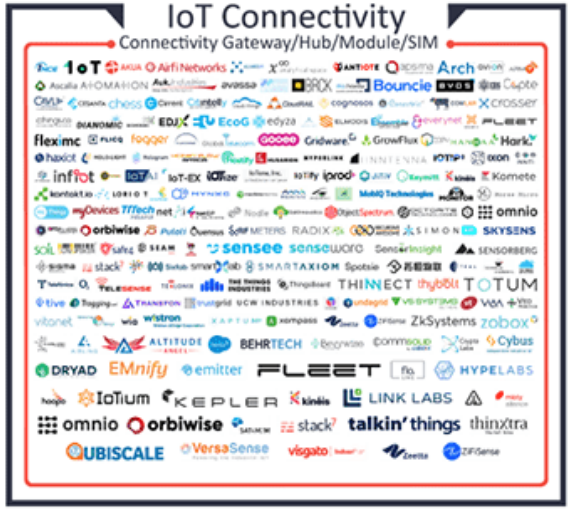
June 2021

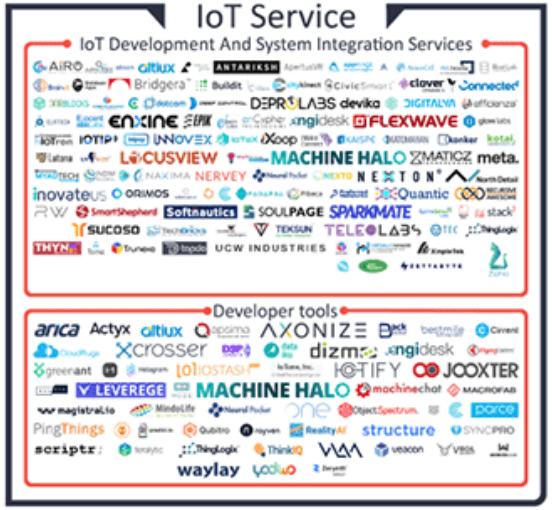
Your Global IoT Market Research Partner

IoT Startup Landscape 2021 – 1,200+ companies







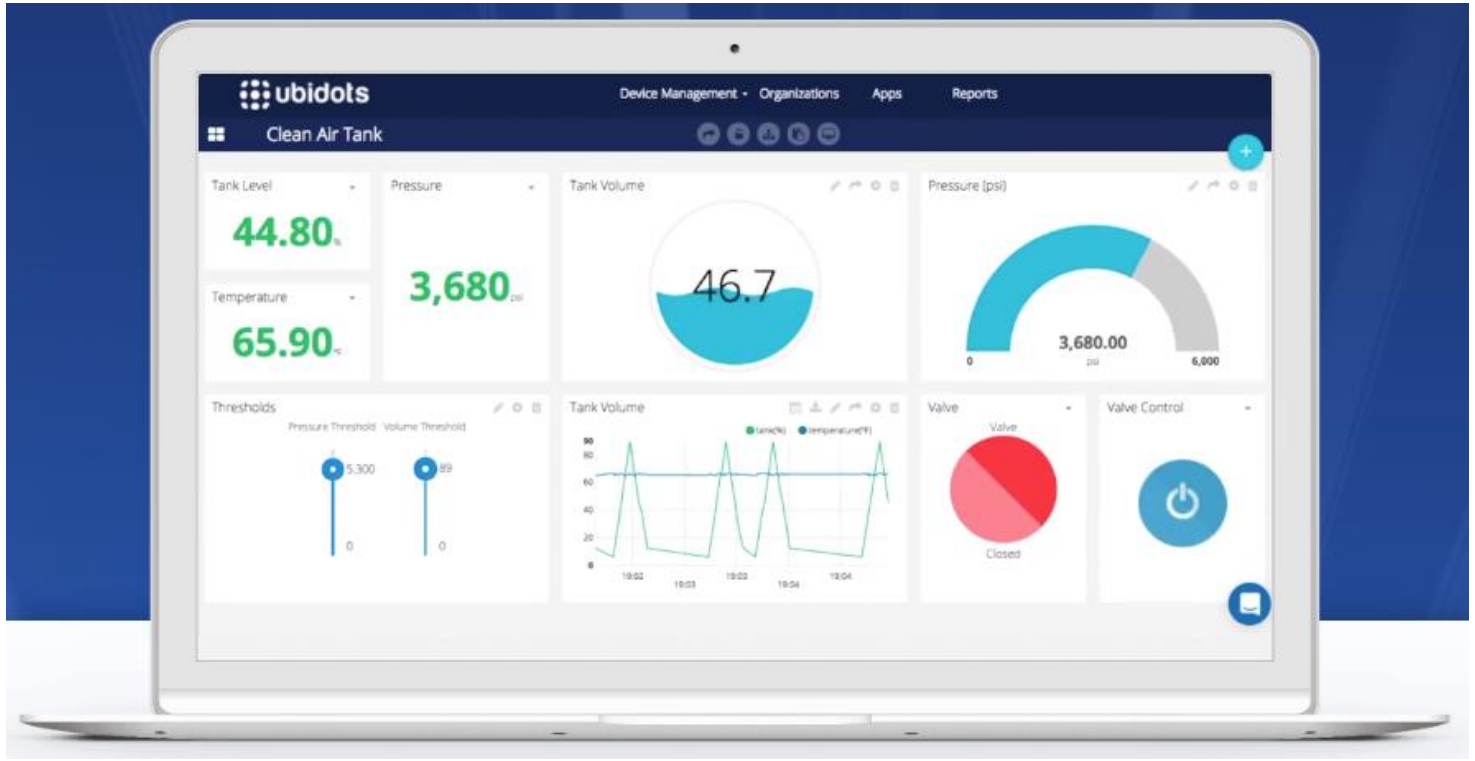


Note: For companies to be included in IoT Analytics IoT startups database they must be founded in or after January 2013, focus on building solutions for the Internet of Things and provide part of a solution focused on at least one area of the IoT technology stacks layers. Companies can offer more than one technology. This list represents a best effort to capture the entire landscape; however, it does not claim to be exhaustive, as some start-ups will have inevitably been missed

Source: IoT Analytics Research 2021, IoT Startups Report and Database 2021

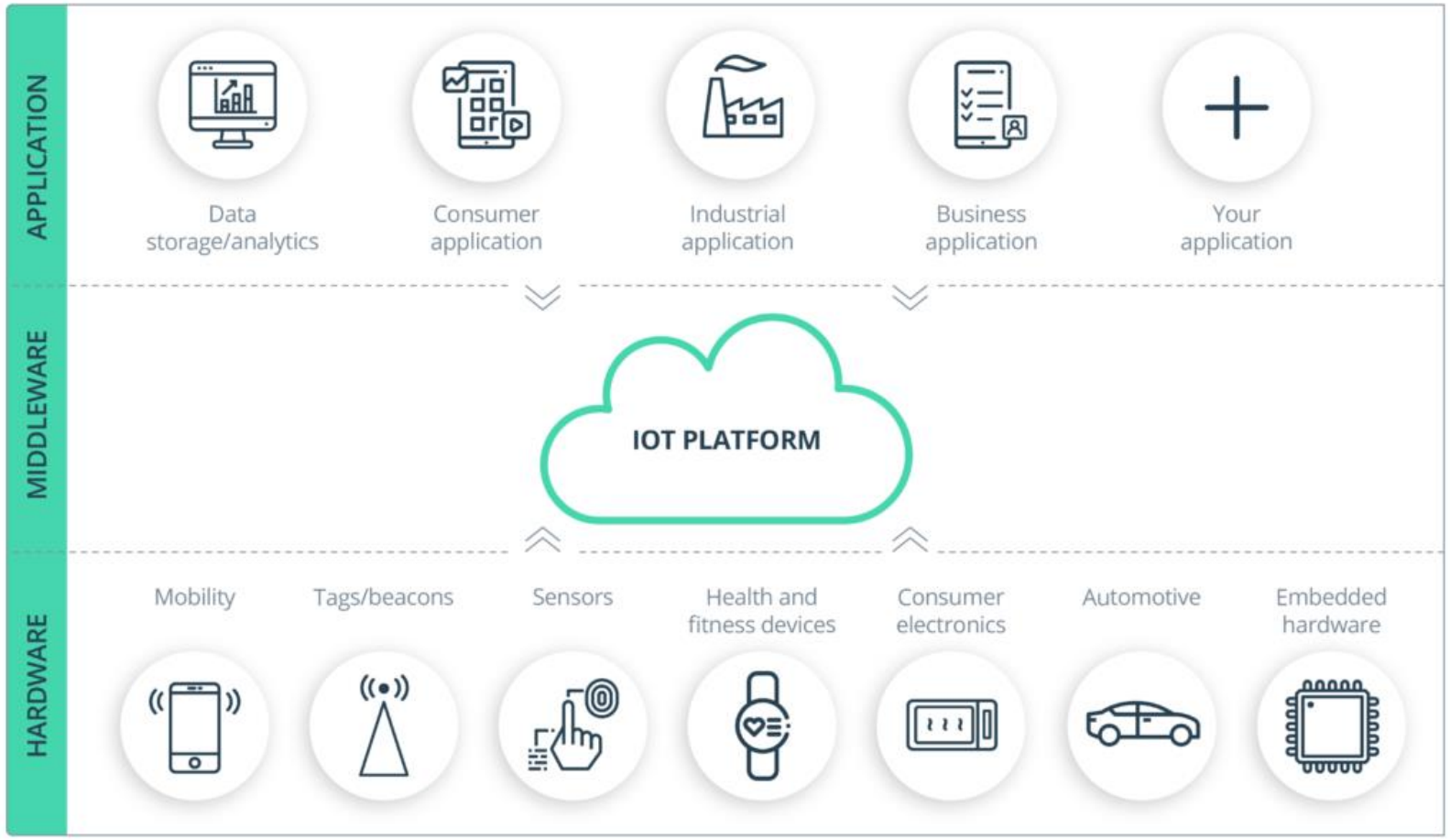
IoT Platform (1/3)

- Upload the sensory information onto an IoT platform.
- Purposes
  - Visualization
  - Storage
  - Processing
  - Notification
  - …



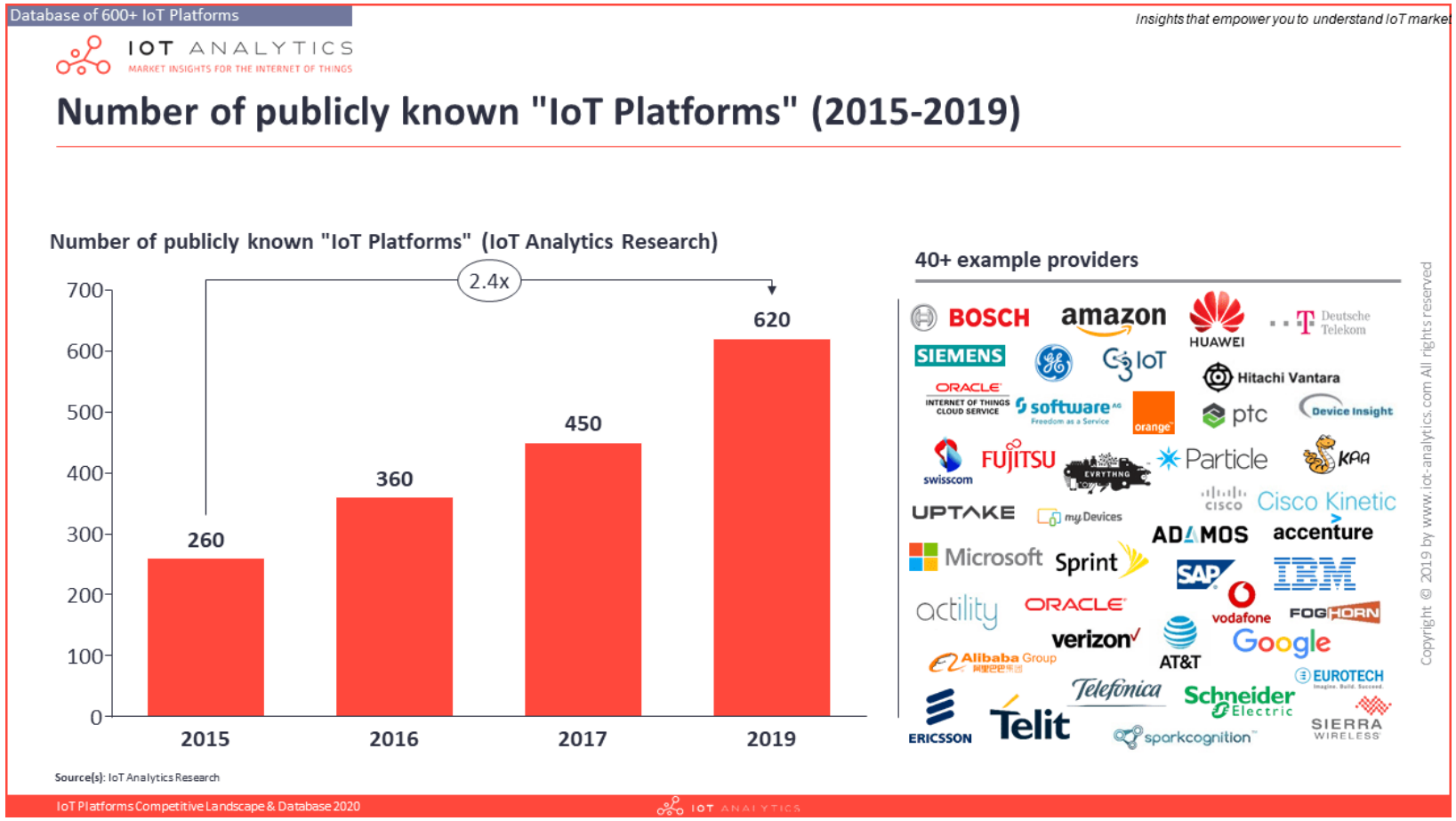
IoT Platform (2/3)

- The architecture of an IoT platform



IoT Platform (3/3)

- Providers of the IoT platform

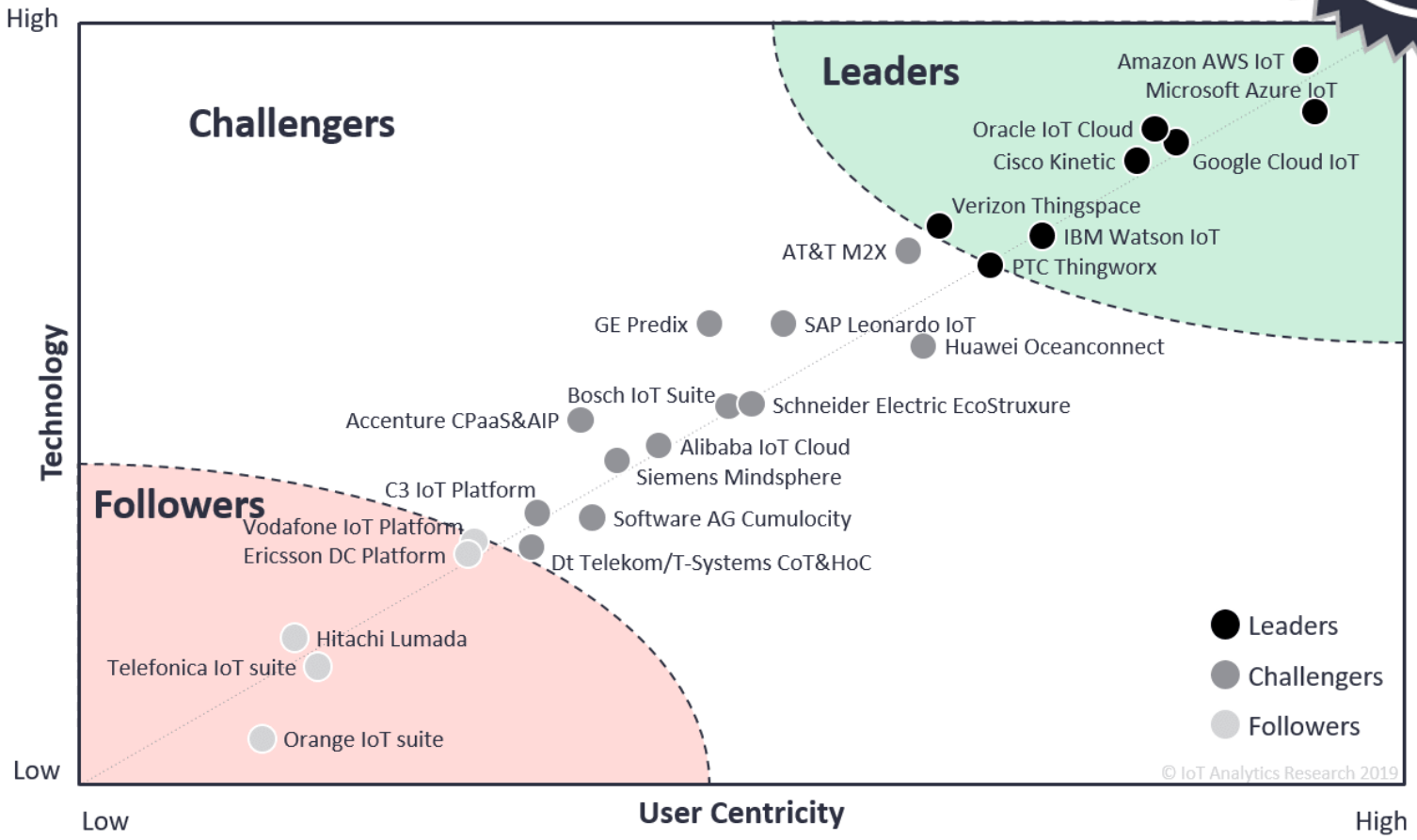


src: https://iot-analytics.com/iot-platform-companies-landscape-2020/

IOT ANALYTICS

Leading IoT Platforms 2019

Based on customer reviews



Note: The ranking is entirely based on IoT Platform end-users views. User Centricity = Usability of the platform (incl. ease of use), Support (documentation, hotline, etc.), Value for money; Technology = Maturity of cloud services, application enablement, device management, connectivity management, data management & analytics, security, interoperability, and the completeness of the total offering (each considered independently) **Source(s)**: IoT Analytics Research – July 2019

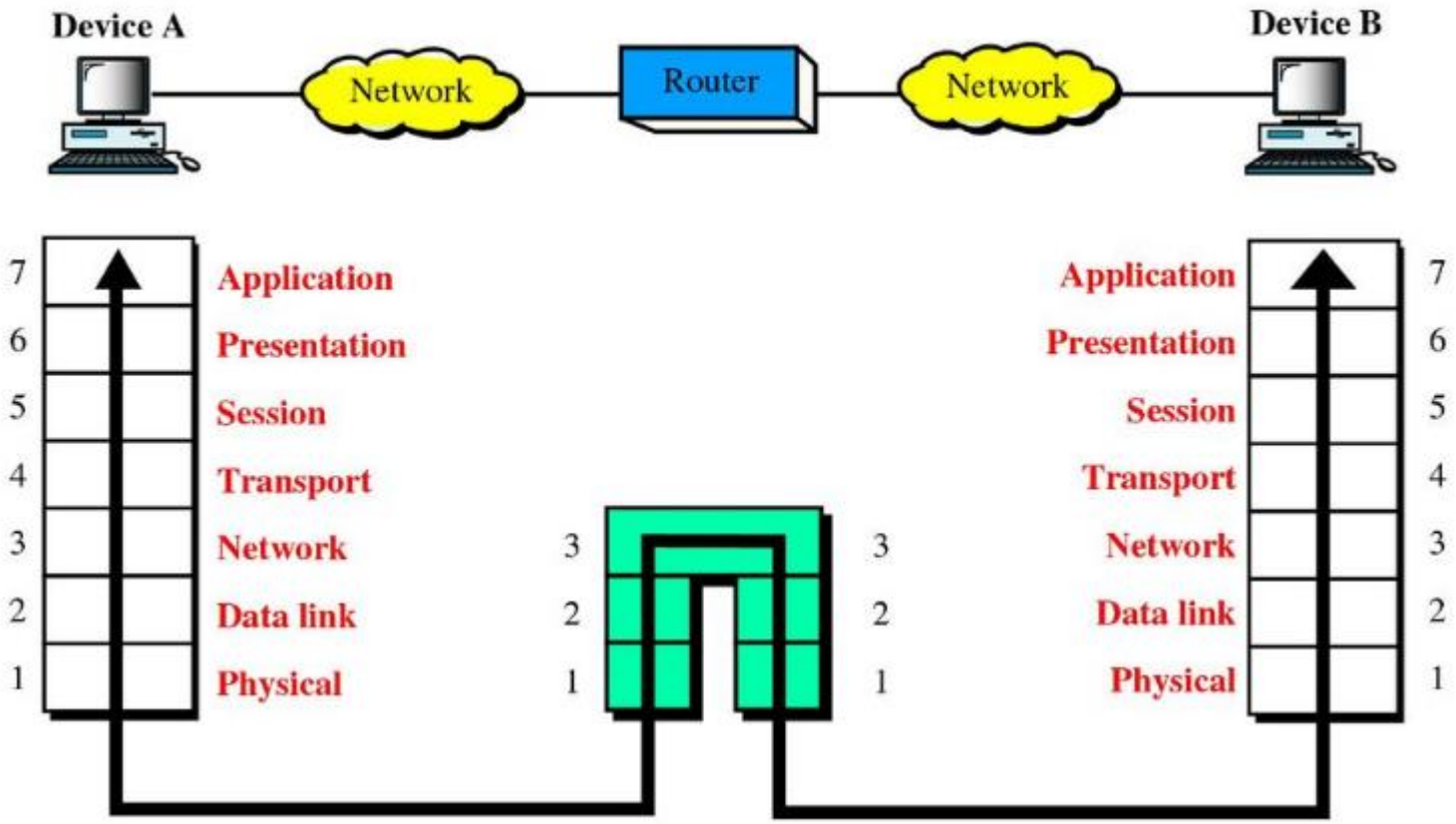
src: https://iot-analytics.com/the-25-best-iot-platforms-2019/

Outline

- Introduction
  - IoT Architecture
  - IoT Networks
  - IoT Platform
- Network-controlled LED
  - TCP socket server
  - HTTP server
- IoT Platform
  - Ubidots
- Lab
  - Remote switch

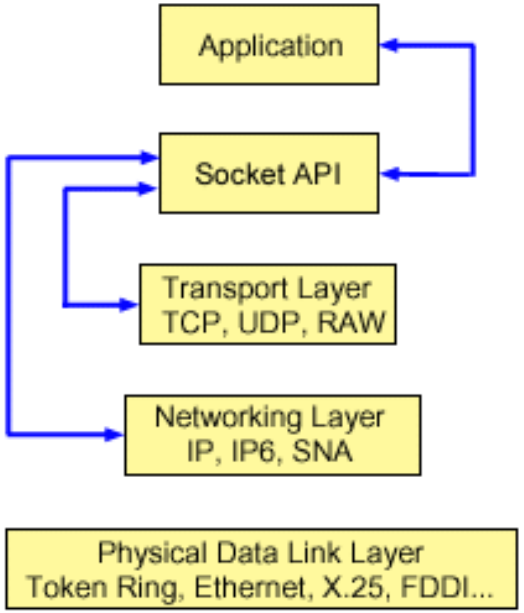
Network Model

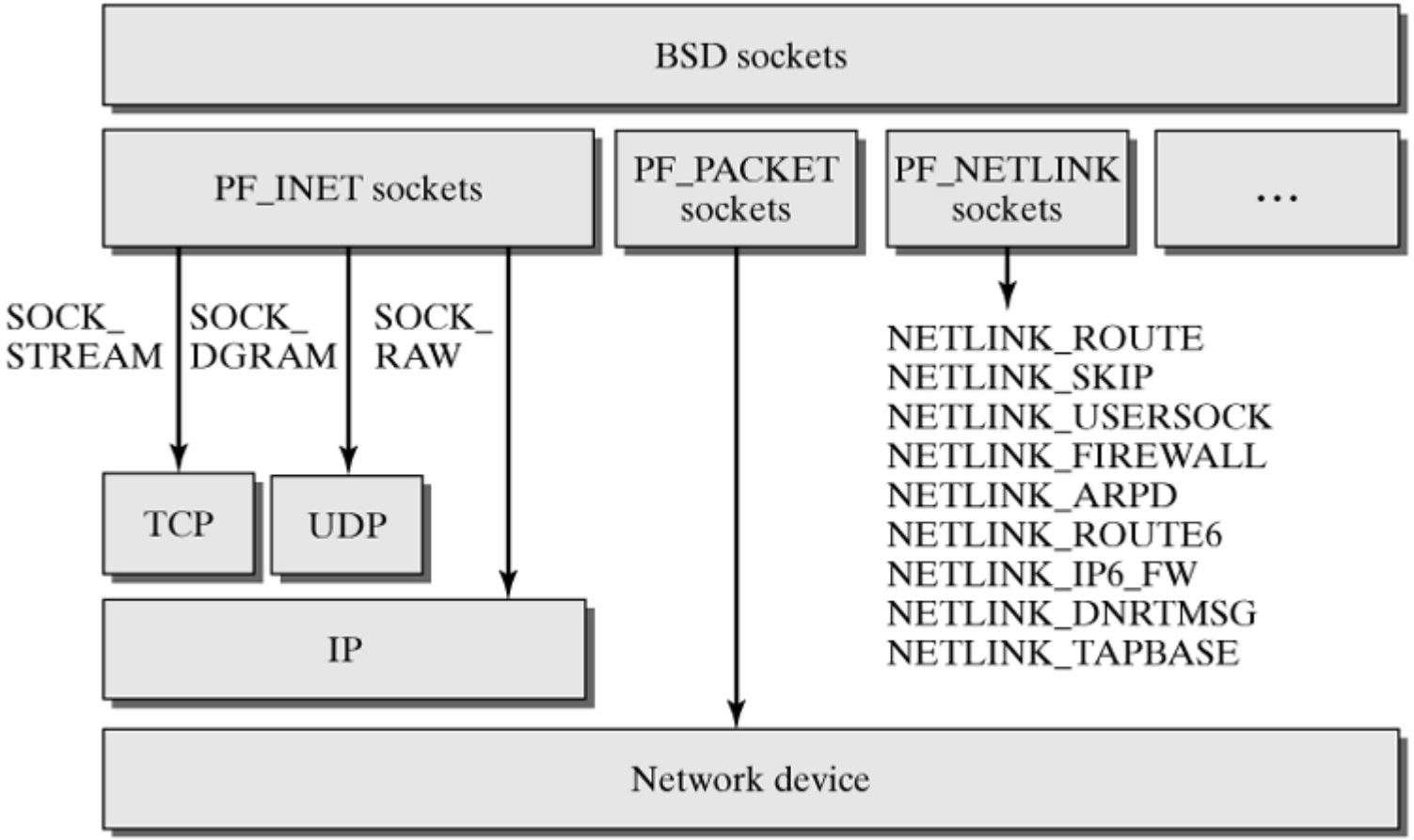
- OSI 7 Layers



Network Socket

- The API provided by OS.

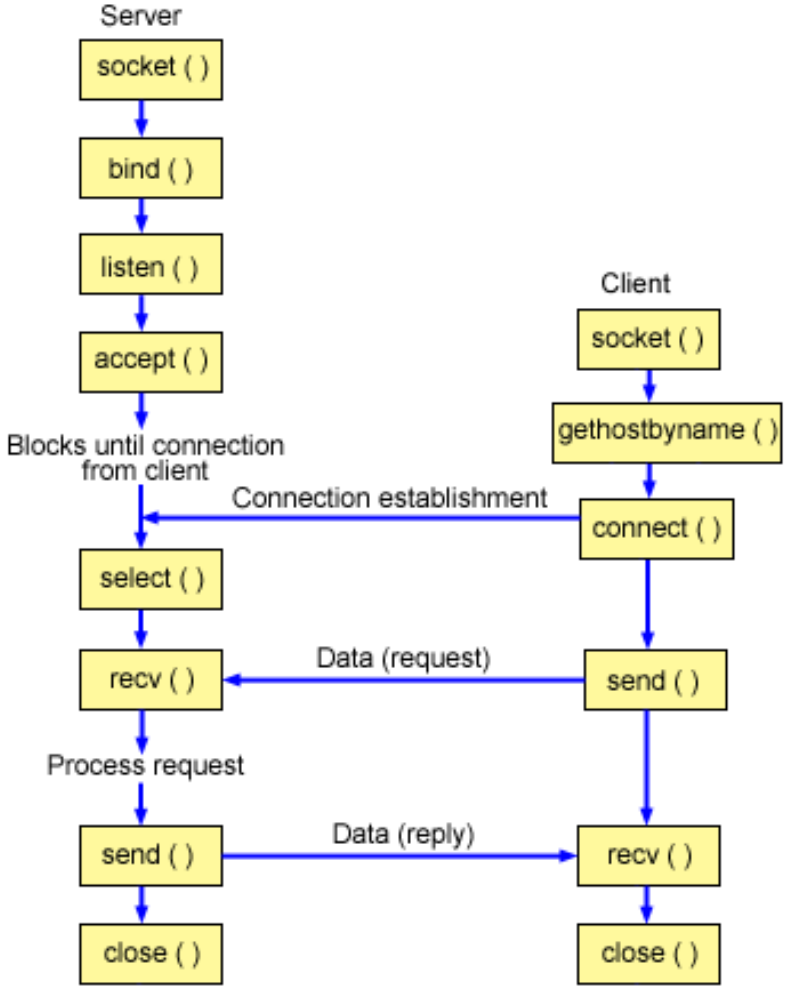




ref: https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzab6/rzab6soxoverview.htm

Network Socket

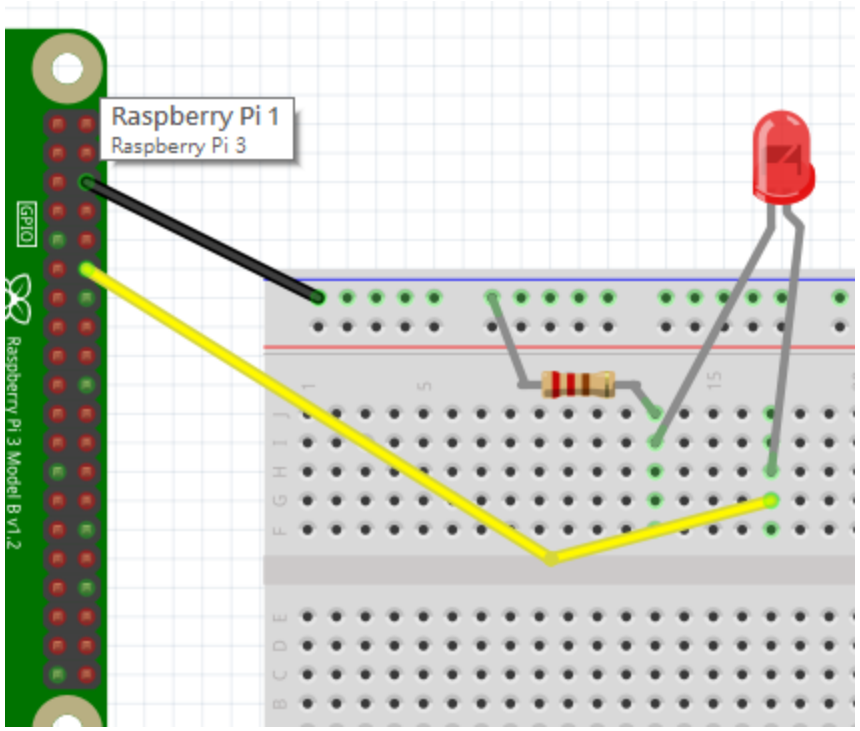
- Client/server relationship of the sockets API for a connection-oriented protocol.



ref: https://www.ibm.com/support/knowledgecenter/ssw_ibm_i_72/rzab6/rzab6soxoverview.htm

Circuit

- PIN 12 - LED(+)
- PIN 6 - LED(-)



TCP Server

- Control an LED according to the data from a TCP client.
- tcpserv.py

```python
import RPi.GPIO as GPIO
import socket

LED_PIN = 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)

HOST = '0.0.0.0'
PORT = 8000
socks = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
socks.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
socks.bind((HOST, PORT))
socks.listen(5)

print('server start at: %s:%s' % (HOST, PORT))
print('wait for connection...')
```

```python
try:
    while True:
        conn, addr = socks.accept()
        print('connected by ' + str(addr))
        while True:
            indata = conn.recv(1024)
            if len(indata) == 0: # connection closed
                conn.close()
                print('client closed connection.')
                break
            data = indata.decode("utf-8").strip()
            print('recv: %s' % data)
            if "1" in data:
                print("led on.")
                GPIO.output(LED_PIN, GPIO.HIGH)
            elif "0" in data:
                print("led off.")
                GPIO.output(LED_PIN, GPIO.LOW)
            # outdata = 'echo ' + indata.decode()
            # conn.send(outdata.encode())
except KeyboardInterrupt:
    print("Exception: KeyboardInterrupt")
finally:
    GPIO.cleanup()
```

TCP Server

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec06/tcpserv.py

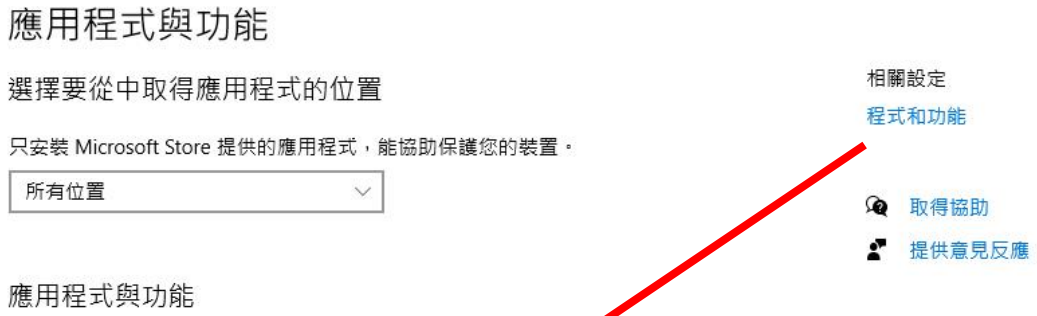
```
pi@rpi4-A00:~ $ wget https://yachentw.github.io/tmp/iot/tcpserv.py
--2020-10-24 22:20:56--  https://yachentw.github.io/tmp/iot/tcpserv.py
Resolving yachentw.github.io (yachentw.github.io)... 185.199.108.153, 185.199.10
9.153, 185.199.110.153, ...
Connecting to yachentw.github.io (yachentw.github.io)|185.199.108.153|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 1183 (1.2K) [application/octet-stream]
Saving to: 'tcpserv.py'

tcpserv.py          100%[===================>]   1.16K  --.-KB/s    in 0s

2020-10-24 22:20:57 (6.65 MB/s) - 'tcpserv.py' saved [1183/1183]
```

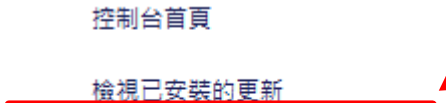
$ python3 tcpserv.py

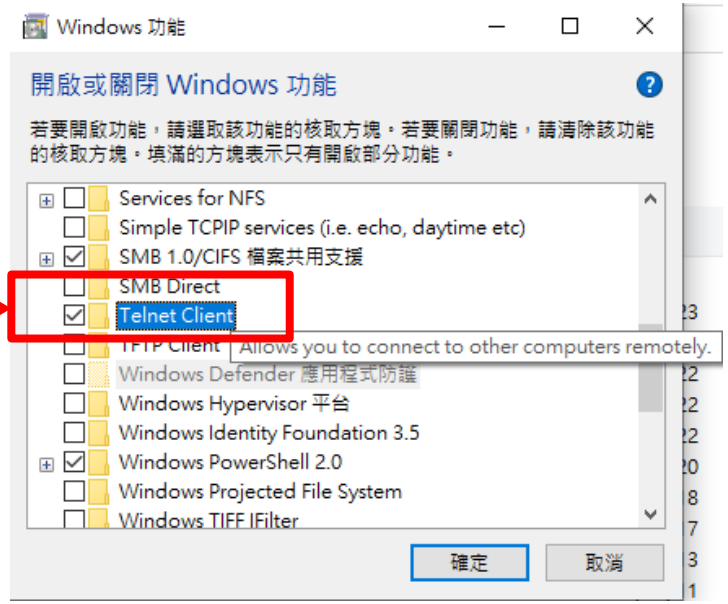
```
pi@rpi4-A00:~ $ python3 tcpserv.py
server start at: 0.0.0.0:8000
wait for connection...
```

TCP Client

Mouse right click







Network-controlled LED

- Client on PC

  > telnet 140.138.145.158 8xxx

  (Replace xxx with your IP's last 3 digits)

- Server on RPI

  $ python3 tcpserv.py

命令提示字元

```
Microsoft Windows [版本 10.0.18363.1139]
(c) 2019 Microsoft Corporation. 著作權所有，並保留一切權利。

C:\Users\cya>telnet 140.138.145.158 8xxx
```

Telnet 192.168.55.111

Type 1 or 0

```
pi@rpi4-A00:~/iot/lec06 $ python3 tcpserv.py
server start at: 0.0.0.0:8000
wait for connection...
connected by ('192.168.55.213', 9465)
recv: 1
led on.
recv: 0
led off.
recv: 1
led on.
recv: 0
led off.
recv: 1
led on.
recv: 0
led off.
```

HTTP

- Hypertext Transfer Protocol (HTTP) is an application layer protocol which runs on TCP socket.

```
pi@rpi4-A00:~/iot/lec06 $ python3 tcpserv.py
server start at: 0.0.0.0:8000
wait for connection...
connected by ('192.168.55.213', 9499)
recv: GET / HTTP/1.1
Host: 192.168.55.111:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/86.0.4240.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-TW,zh;q=0.9,en-US;q=0.8,en;q=0.7
led on.
```

HTTP

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec06/httpserv.py

$ python3 httpserv.py

```
pi@rpi4-A00:~ $ python3 httpserv.py
 * Serving Flask app "httpserv" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
```

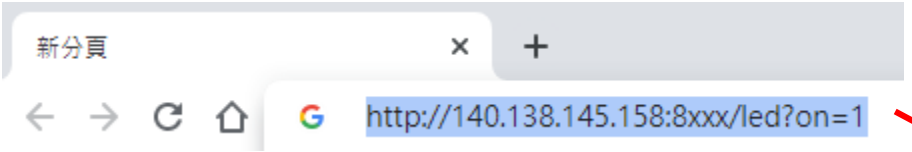
```python
from flask import Flask, request
import RPi.GPIO as GPIO

LED_PIN = 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)

app = Flask(__name__)

def led(cmd):
    if cmd == 1:
        print("led on.")
        GPIO.output(LED_PIN, GPIO.HIGH)
    elif cmd == 0:
        print("led off.")
        GPIO.output(LED_PIN, GPIO.LOW)
```

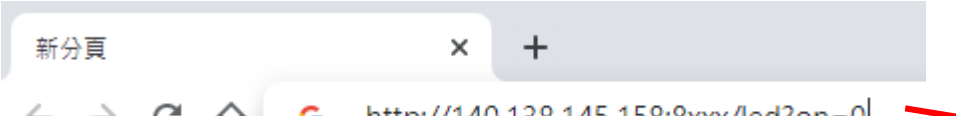
```python
@app.route('/')
def index():
    return "Index Page"
@app.route('/cmd/<int:num>')
def cmd(num):
    led(num)
    return "cmd: %d" % (num)

@app.route('/led', methods=['GET'])
def ledcmd():
    if request.method == 'GET':
        cmd = request.args.get('on')
        led(int(cmd))
        return "LED on: %s" % cmd

if __name__ == '__main__':
    try:
        app.run(host='0.0.0.0', port=8000)
    except KeyboardInterrupt:
        print("Exception: KeyboardInterrupt")
    finally:
        GPIO.cleanup()
```

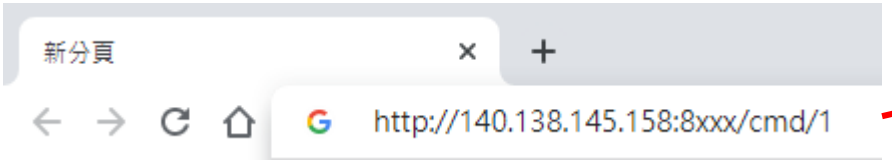
HTTP Request

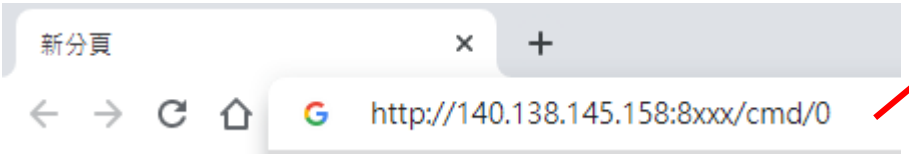
- Use HTTP request from browser to control the LED

(Replace xxx with your IP's last 3 digits)









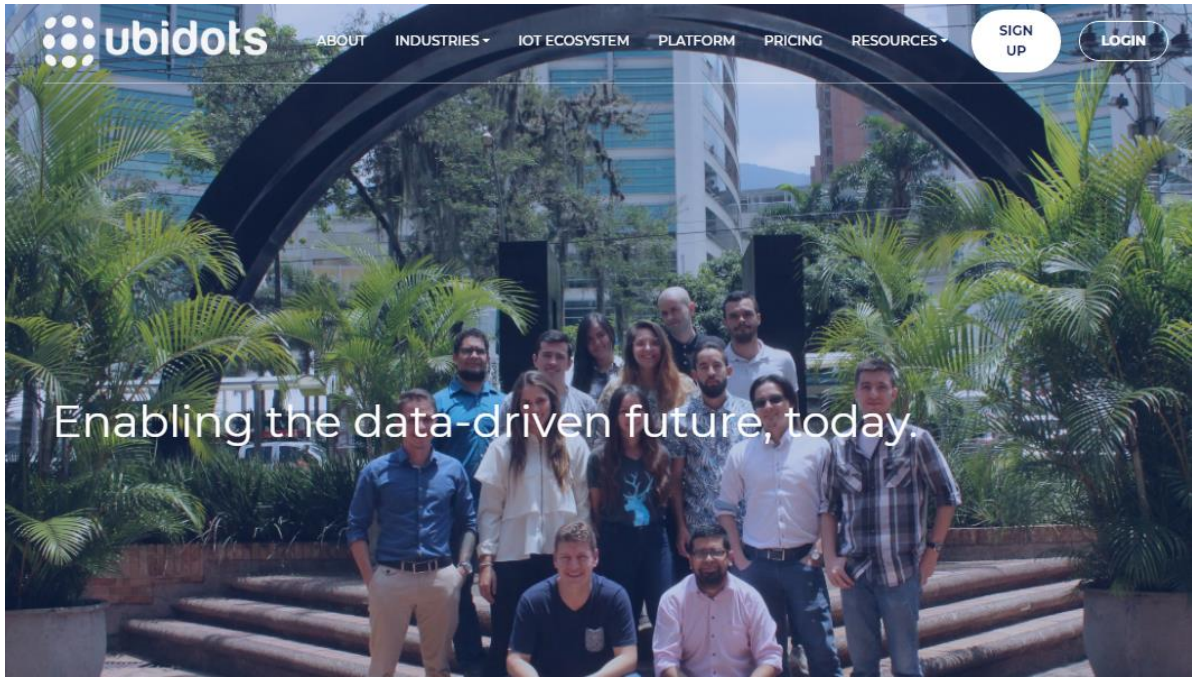

```
pi@rpi4-A00:~/iot/lec06 $ python3 httpserv.py
 * Serving Flask app "httpserv" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:8000/ (Press CTRL+C to quit)
led on.
192.168.55.213 - - [24/Oct/2020 17:56:04] "GET /led?on=1 HTTP/1.1" 200 -
led off.
192.168.55.213 - - [24/Oct/2020 17:56:09] "GET /led?on=0 HTTP/1.1" 200 -
led on.
192.168.55.213 - - [24/Oct/2020 17:56:14] "GET /led?on=1 HTTP/1.1" 200 -
led on.
192.168.55.213 - - [24/Oct/2020 17:56:40] "GET /cmd/1 HTTP/1.1" 200 -
led off.
192.168.55.213 - - [24/Oct/2020 17:56:43] "GET /cmd/0 HTTP/1.1" 200 -
```

Outline

- Introduction
  - IoT Architecture
  - IoT Networks
  - IoT Platform
- Network-controlled LED
  - TCP socket server
  - HTTP server
- IoT Platform
  - Ubidots
- Lab
  - Remote switch

ubidots

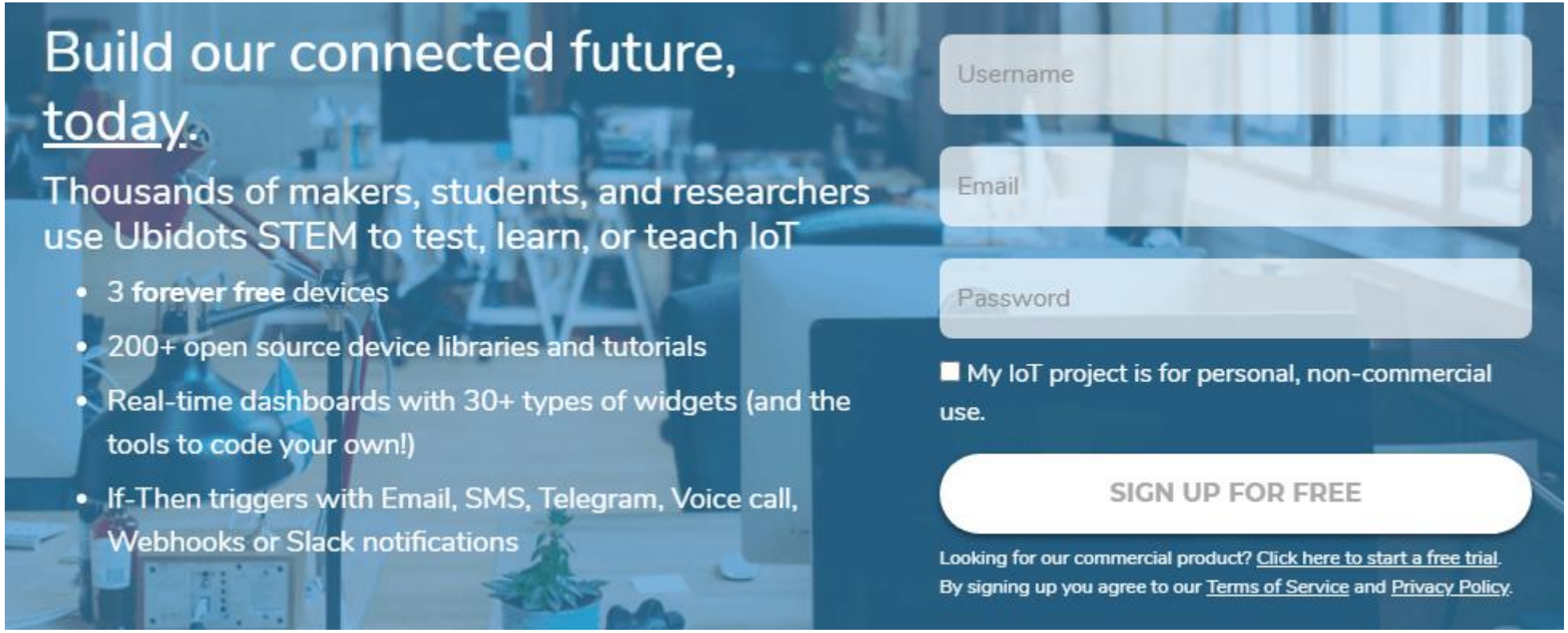
- Ubidots is an IoT Platform empowering innovators and industries to prototype and scale IoT projects to production.



ubidots

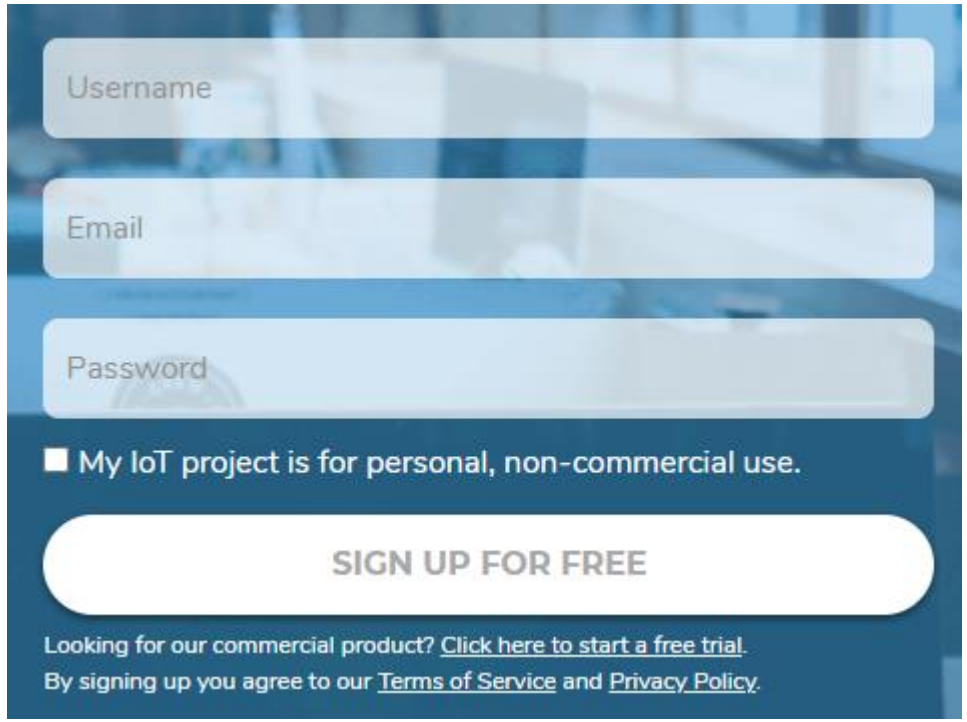
- ubitdots STEM (Science, Technology, Engineering and Mathematics)

https://ubidots.com/stem/

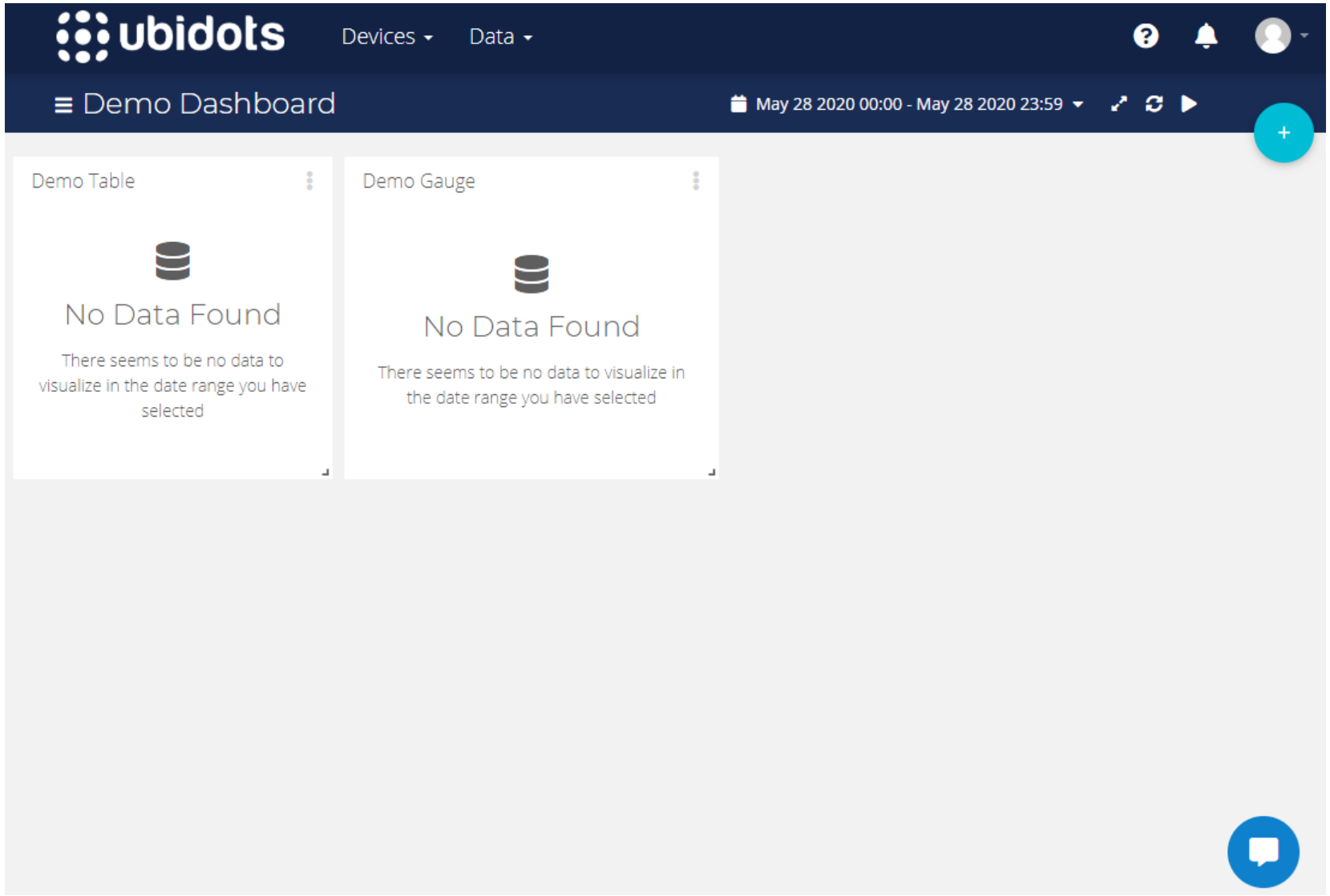


Sign Up

- https://ubidots.com/stem/



Sign In

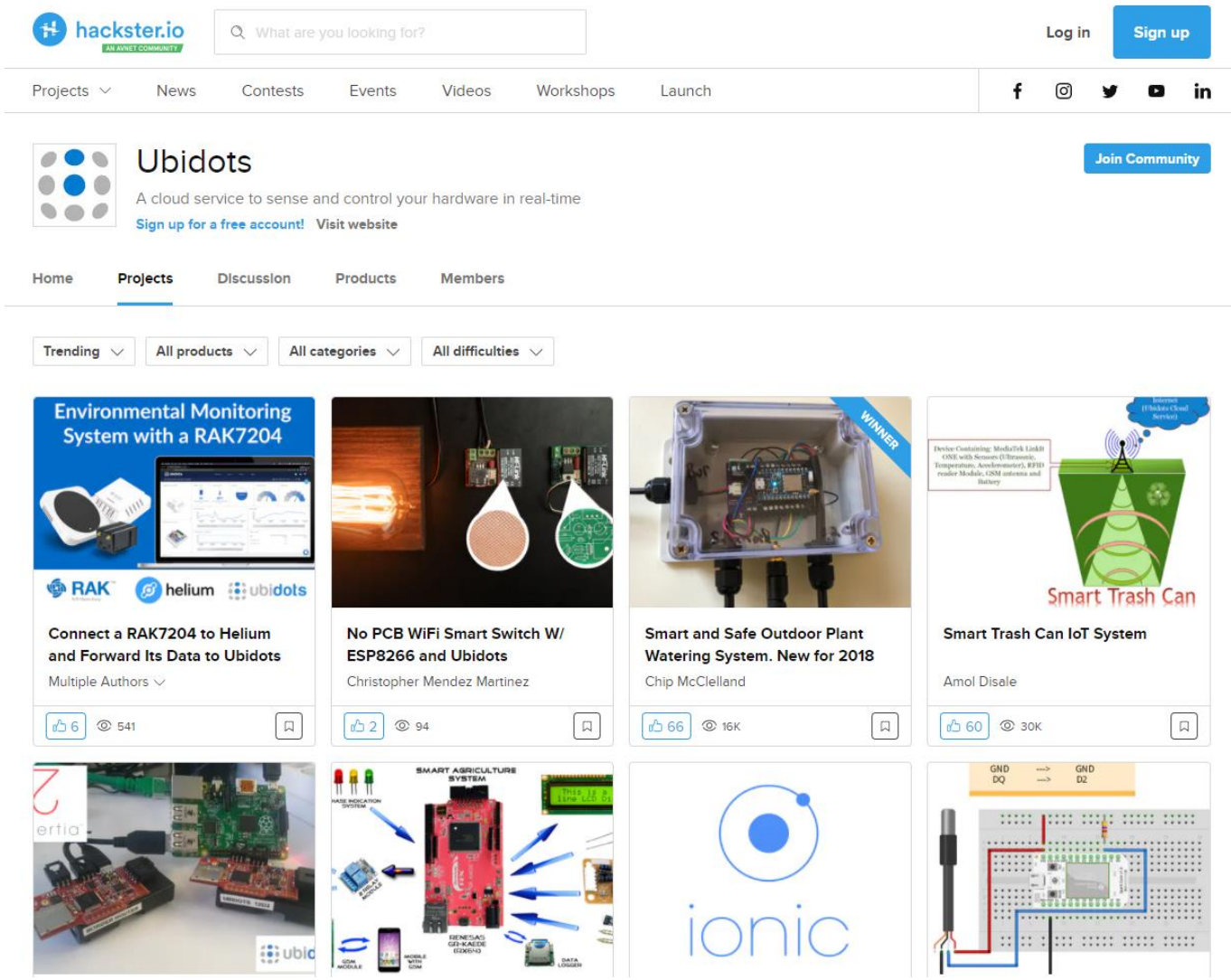

ubidots
Devices
Data
Demo Dashboard
May 28 2020 00:00 - May 28 2020 23:59
Demo Table
No Data Found
There seems to be no data to visualize in the date range you have selected
Demo Gauge
No Data Found
There seems to be no data to visualize in the date range you have selected

Ubidots & Python

- Use a python program to upload generated values onto the platform.
- Ubidots documents
  - https://ubidots.com/docs/

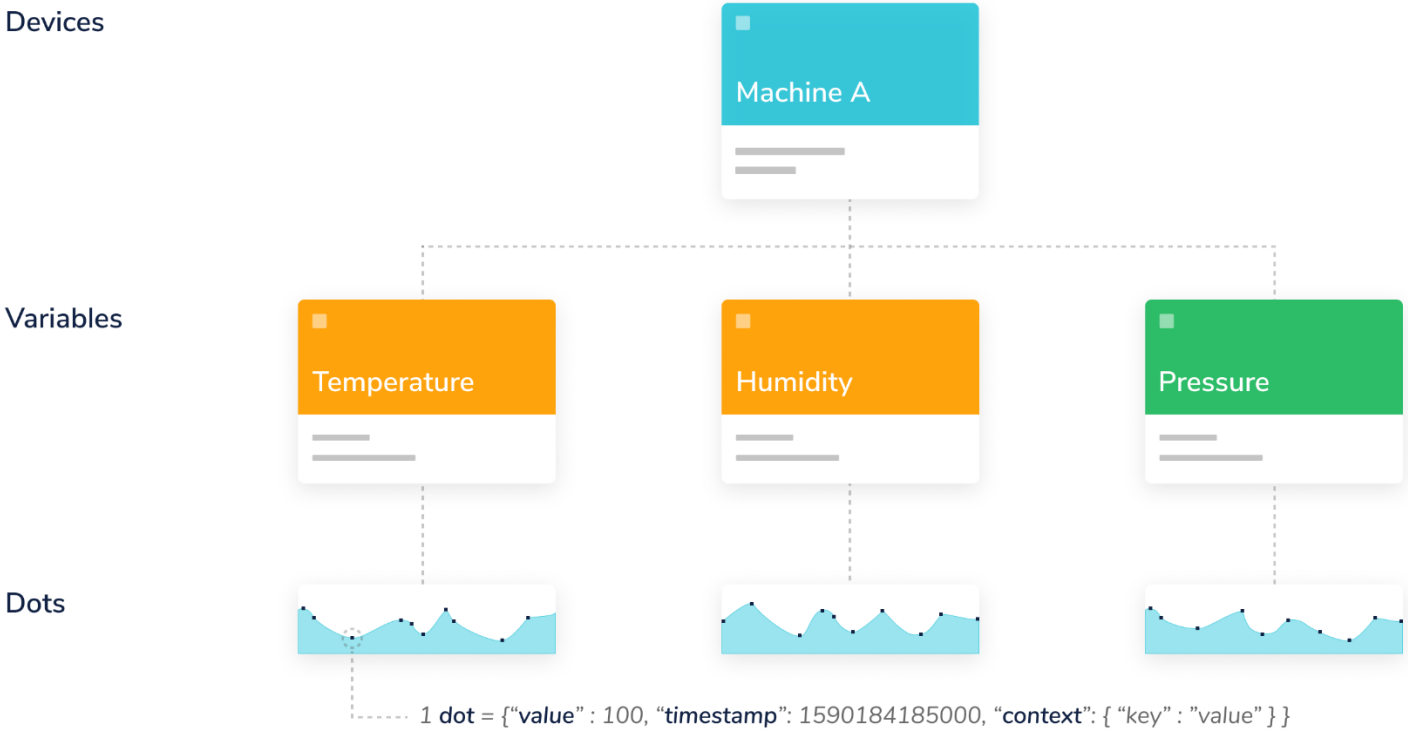
Projects

- For your reference
- https://www.hackster.io/ubidots/projects



IoT Friendly API

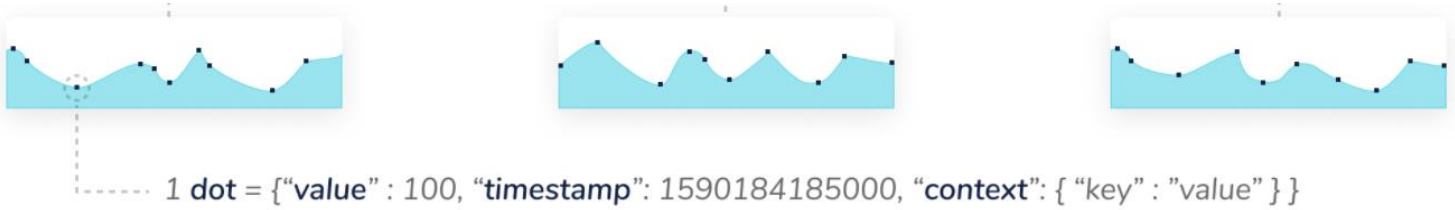
- https://ubidots.com/docs/hw/
- Send data to or retrieve data from your hardware devices using supported communication protocols: HTTP, MQTT and TCP/UDP.



Each dot contains these items:

Item	Description	Mandatory
value	A numerical value. Ubidots accepts up to 16 floating-point length numbers.	Yes
timestamp	Unix Epoch time, in milliseconds. If not specified, then our servers will assign one upon reception.	No
context	An arbitrary collection of key-value pairs. Mostly used to store the latitude and longitude coordinates of GPS devices.	No

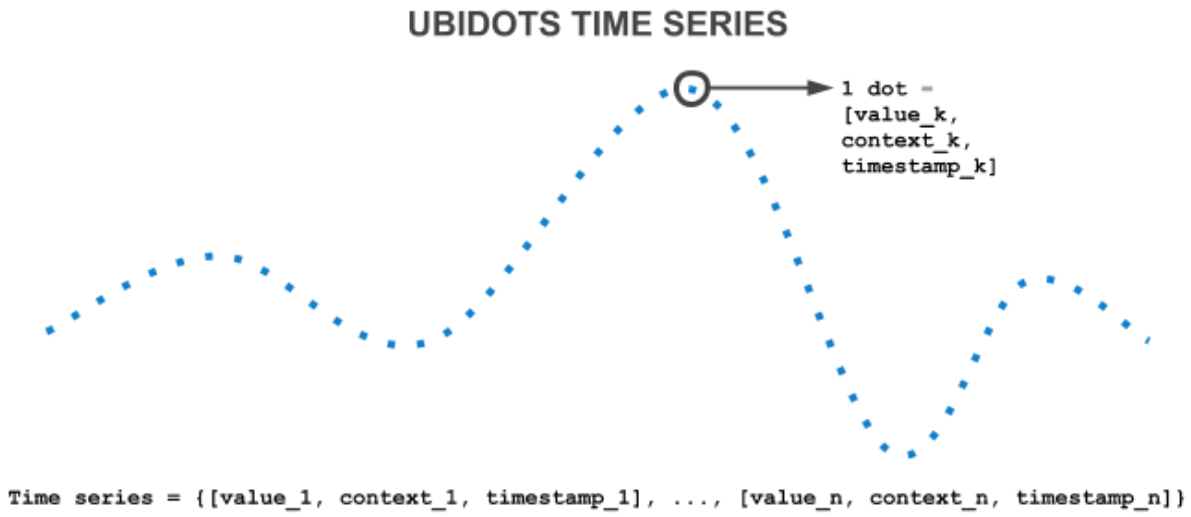
Items in a Dot



- Values
  - A numerical value. Ubidots accepts up to 16 floating-point length numbers.
  - Ex: { "value" : 34.87654974 }
- Timestamps
  - A way to track time as a running total of seconds.
  - This count starts at the Unix Epoch on January 1st, 1970 at UTC.
  - In ubidots, timestamp is in milliseconds.
  - Ex: "timestamp" : 1537453824000
    Thursday, September 20, 2018 2:30:24 PM.
- Context
  - A key-value object that allows you to store not only numerical but also string values.
  - Ex: "context" : {"status" : "on", "weather" : "sunny"}
  - For GPS coordinates, you just need to send a single dot with the coordinates values in the variable context
  - Ex: "context" : {"lat":-6.2, "lng":75.4, "weather" : "sunny"}

Time Series

- Ubidots time series



- No matter what hardware device you are using, you are able to interact with ubidots through at least one of these protocols:
  - HTTP
  - MQTT
  - TCP/UDP

HTTP Requests

- https://docs.ubidots.com/v1.6/reference/http

Ubidots API docs [hardware engineers]

Introduction
How Ubidots Works?
Time Series
HTTP
API URLs
HTTP Authentication
Send data
Retrieve data
HTTP Examples
Response codes
MQTT
Broker URLs
MQTT Authentication
Send data (Publish)
Retrieve data (Subscribe)
MQTT Examples
TCP / UDP
TCP Authentication
TCP URLs
Send Data
Retrieve Data
Utils

enabling you to replicate it within your own firmware. Because of this, we avoid using our custom libraries in all the examples.

HTTP requests

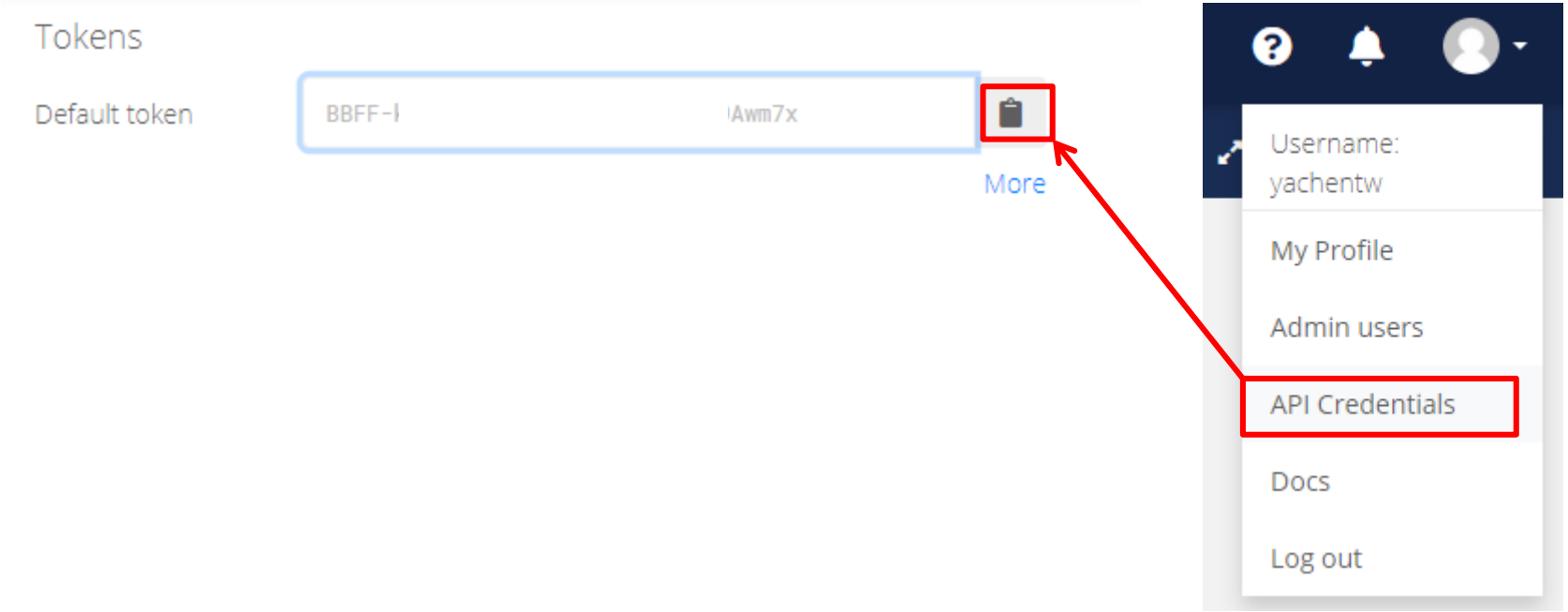
The following methods are specified within the *HTTP* standard:

HTTP Method	Description
`GET`	Used to retrieve information
`POST`	Used to create a new element
`PATCH`	Used to update existing elements
`DELETE`	Used to delete existing elements

HTTP is a **request/response** protocol, this means that every request that you make is answered by the server. This response includes a number (response code) and a body. For example, when you make a request to retrieve a file on a webpage *"(e.g. "Get me the file 'webside.html'")"*, you build a GET request. If the request is correct, the server will typically return a 200 response code, along with the file requested (body).

HTTP Authentication

- Every request requires a token. A token is an unique key that authorizes your device to ingest data inside your Ubidots account.
- Copy your token.



Send Data

- https://docs.ubidots.com/v1.6/reference/sending-data
- Examples

URL

```
POST https://industrial.api.ubidots.com/api/devices/{DEVICE_LABEL}
```

Request

Request

```
POST /api/v1.6/devices/{DEVICE_LABEL} HTTP/1.1<CR><LN>
Host: {Endpoint}<CR><LN>
User-Agent: {USER_AGENT}<CR><LN>
X-Auth-Token: {TOKEN}<CR><LN>
Content-Type: application/json<CR><LN>
Content-Length: {PAYLOAD_LENGTH}<CR><LN><CR><LN>
{PAYLOAD}
<CR><LN>
```

Response

Response

```
HTTP/1.1 200 OK<CR><LN>
Server: nginx<CR><LN>
Date: Tue, 04 Sep 2018 22:35:06 GMT<CR><LN>
Content-Type: application/json<CR><LN>
Transfer-Encoding: chunked<CR><LN>
Vary: Cookie<CR><LN>
Allow: GET, POST, HEAD, OPTIONS<CR><LN><CR><LN>
{PAYLOAD_LENGTH_IN_HEXADECIMAL}<CR><LN>
{"{VARIABLE_LABEL}": [{"status_code": 201}]}<CR><LN>
0<CR><LN>
```

```python
import requests
import random
import time

'''
global variables
'''

ENDPOINT = "things.ubidots.com"
DEVICE_LABEL = "weather-station"
VARIABLE_LABEL = "temperature"
TOKEN = "..."
DELAY = 1  # Delay in seconds


def post_var(payload, url=ENDPOINT, device=DEVICE_LABEL, token=TOKEN):
    try:
        url = "http://{}/api/v1.6/devices/{}".format(url, device)
        headers = {"X-Auth-Token": token, "Content-Type": "application/json"}

        attempts = 0
        status_code = 400

        while status_code >= 400 and attempts < 5:
            print("[INFO] Sending data, attempt number: {}".format(attempts))
            req = requests.post(url=url, headers=headers,
                                json=payload)
            status_code = req.status_code
            attempts += 1
            time.sleep(1)

        print("[INFO] Results:")
        print(req.text)
    except Exception as e:
        print("[ERROR] Error posting, details: {}".format(e))
```

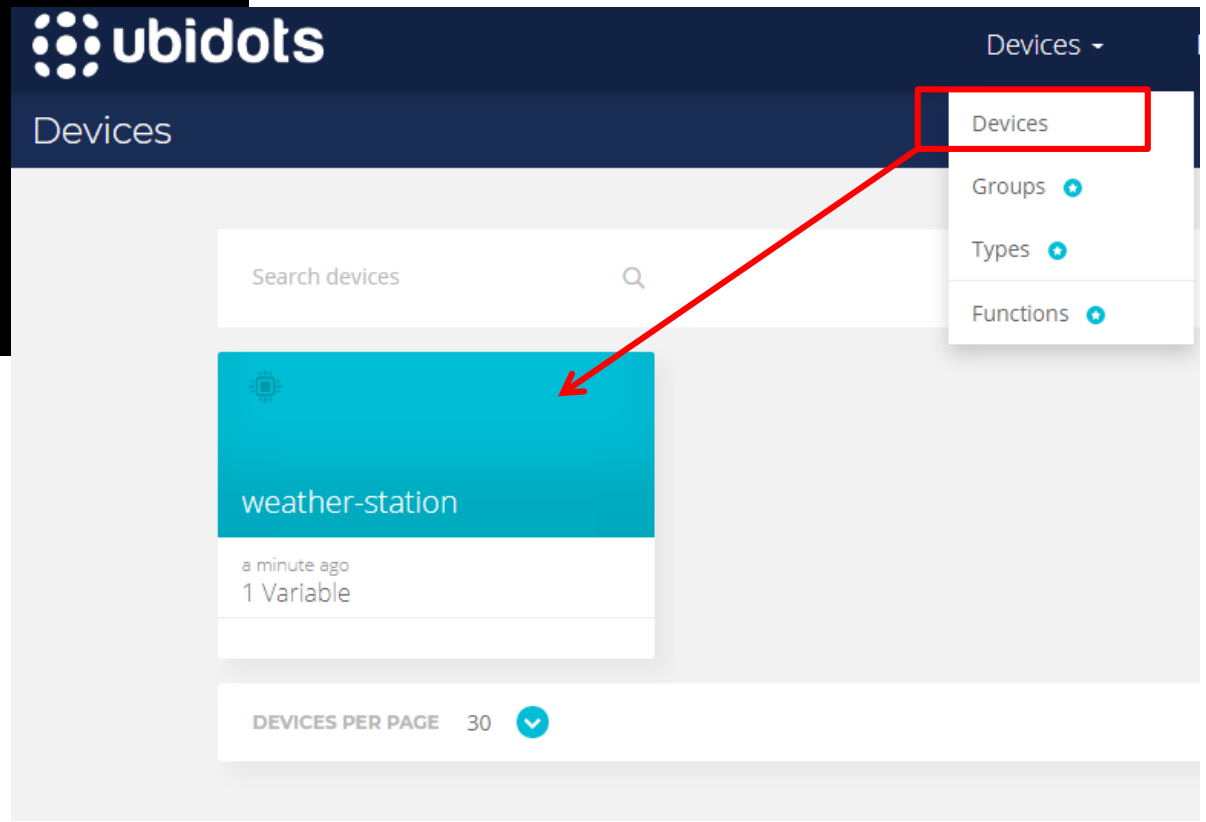

```python
def main():
    # Simulates sensor values
    sensor_value = random.random() * 100

    # Builds Payload and topic
    payload = {VARIABLE_LABEL: sensor_value}

    # Sends data
    post_var(payload)


if __name__ == "__main__":
    while True:
        main()
        time.sleep(DELAY)
```

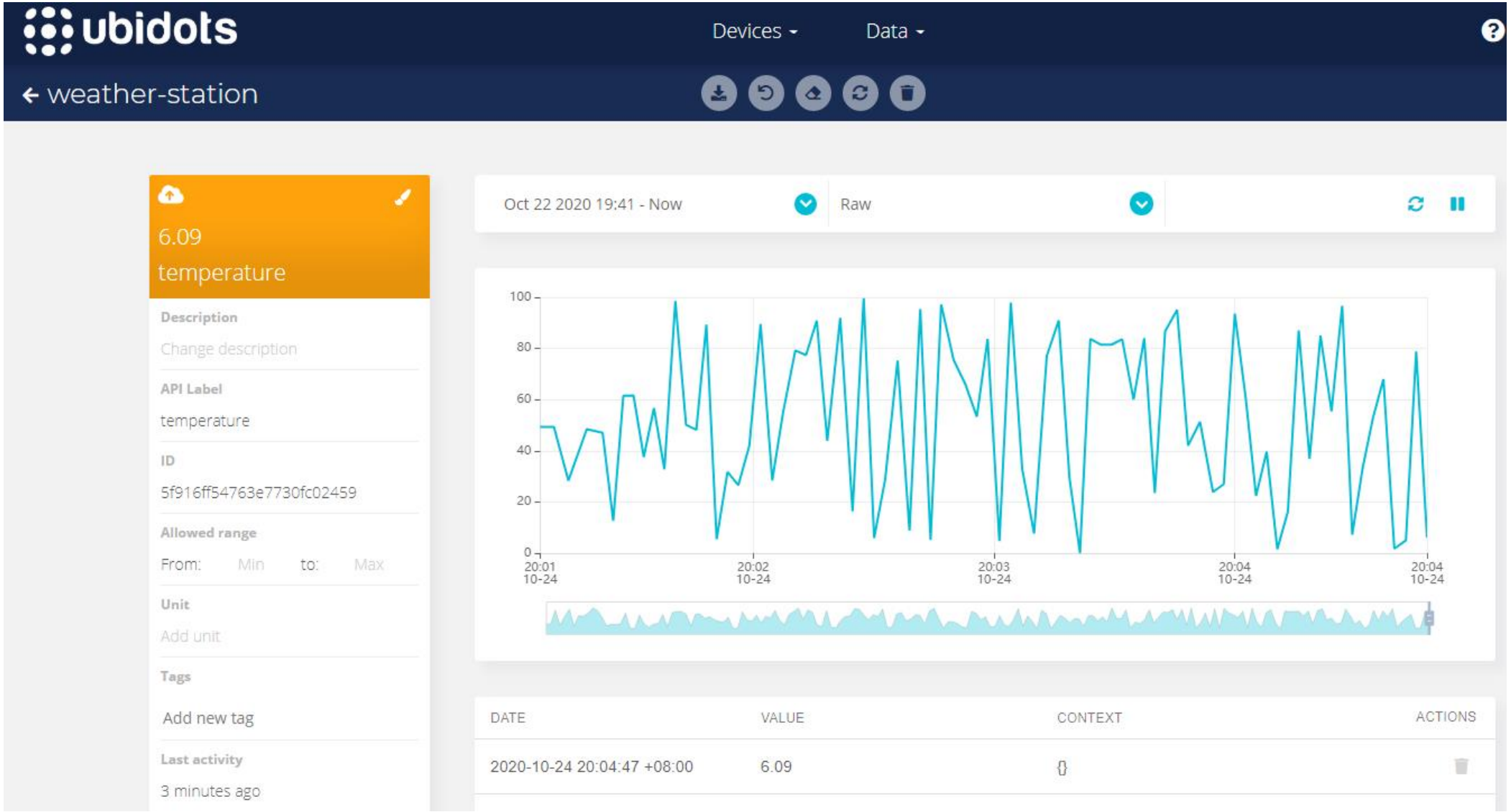
Results

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec06/ubidots_http_send.py

$ python3 ubidots_http_send.py

```
pi@rpi4-A00:~/iot/lec06 $ python3 ubidots_http_send.py
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
```




ubidots
Devices
Data
weather-station
6.09
temperature
Description
Change description
API Label
temperature
ID
5f916ff54763e7730fc02459
Allowed range
From: Min to: Max
Unit
Add unit
Tags
Add new tag
Last activity
3 minutes ago
Oct 22 2020 19:41 - Now
Raw
100
80
60
40
20
0
20:01 10-24
20:02 10-24
20:03 10-24
20:04 10-24
20:04 10-24
DATE
VALUE
CONTEXT
ACTIONS
2020-10-24 20:04:47 +08:00
6.09
{}

Retrieve Data

- https://docs.ubidots.com/v1.6/reference/retrieving-data
- Examples
  - Retrieving Multiple Values from a Variable

Request

Request

```
GET /api/v1.6/devices/{DEVICE_LABEL}/{VARIABLE_LABEL}/values HTTP/1.1<CR><LN>
Host: {Endpoint}<CR><LN>
User-Agent: {USER_AGENT}<CR><LN>
X-Auth-Token: {TOKEN}<CR><LN>
Content-Type: application/json<CR><LN><CR><LN>
```

Response

Response

```
HTTP/1.1 200 OK<CR><LN>
Server: nginx<CR><LN>
Date: Tue, 04 Sep 2018 22:35:06 GMT<CR><LN>
Content-Type: application/json<CR><LN>
Transfer-Encoding: chunked<CR><LN>
Vary: Cookie<CR><LN>
Allow: GET, POST, HEAD, OPTIONS<CR><LN><CR><LN>
{PAYLOAD_LENGTH_IN_HEXADECIMAL}<CR><LN>
{"count": true, "previous": null, "results": [{PAYLOAD_WITH_VALUES}], "next": {URL_WITH_ADDITIONAL_VALUE
0<CR><LN>
```

```python
import requests
import random
import time

'''
global variables
'''

ENDPOINT = "things.ubidots.com"
DEVICE_LABEL = "weather-station"
VARIABLE_LABEL = "temperature"
TOKEN = "..."
DELAY = 1  # Delay in seconds


def get_var(url=ENDPOINT, device=DEVICE_LABEL, variable=VARIABLE_LABEL,
            token=TOKEN):
    try:
        url = "http://{}/api/v1.6/devices/{}/{}/values/?page_size=2".format(url,
                                                                            device,
                                                                            variable)

        headers = {"X-Auth-Token": token, "Content-Type": "application/json"}

        attempts = 0
        status_code = 400

        while status_code >= 400 and attempts < 5:
            print("[INFO] Retrieving data, attempt number: {}".format(attempts))
            req = requests.get(url=url, headers=headers)
            status_code = req.status_code
            attempts += 1
            time.sleep(1)

        print("[INFO] Results:")
        print(req.text)
    except Exception as e:
        print("[ERROR] Error posting, details: {}".format(e))
```

add “values”

```python
if __name__ == "__main__":
    while True:
        get_var()
        time.sleep(DELAY)
```

Last 2 Values

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec06/ubidots_http_get.py

$ python3 ubidots_http_get.py

```
pi@rpi4-A00:~/iot/lec06 $ python3 ubidots_http_get.py
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
{"count": true, "next": "http://things.ubidots.com/api/v1.6/devices/weather-stat
ion/temperature/values/?page_size=2&page=2", "previous": null, "results": [{"tim
estamp": 1603542164689, "value": 62.58, "context": {}, "created_at": 16035421646
89}, {"timestamp": 1603542161984, "value": 54.88, "context": {}, "created_at": 1
603542161984}]}
```

```python
import requests
import random
import time

'''
global variables
'''

ENDPOINT = "things.ubidots.com"
DEVICE_LABEL = "weather-station"
VARIABLE_LABEL = "temperature"
TOKEN = "..."
DELAY = 1  # Delay in seconds


def get_var(url=ENDPOINT, device=DEVICE_LABEL, variable=VARIABLE_LABEL,
            token=TOKEN):
    try:
        url = "http://{}/api/v1.6/devices/{}/{}/lv".format(url,
                                                           device,
                                                           variable)

        headers = {"X-Auth-Token": token, "Content-Type": "application/json"}

        attempts = 0
        status_code = 400

        while status_code >= 400 and attempts < 5:
            print("[INFO] Retrieving data, attempt number: {}".format(attempts))
            req = requests.get(url=url, headers=headers)
            status_code = req.status_code
            attempts += 1
            time.sleep(1)

        print("[INFO] Results:")
        print(req.text)
    except Exception as e:
        print("[ERROR] Error posting, details: {}".format(e))
```

"lv" get the last value

```
pi@rpi4-A00:~/iot/lec06 $ python3 ubidots_http_get.py
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
62.57861623935298
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
62.57861623935298
```

Sending & Retrieving

- Run these two programs at the same time.

pi@rpi4-A00: ~/iot/lec06

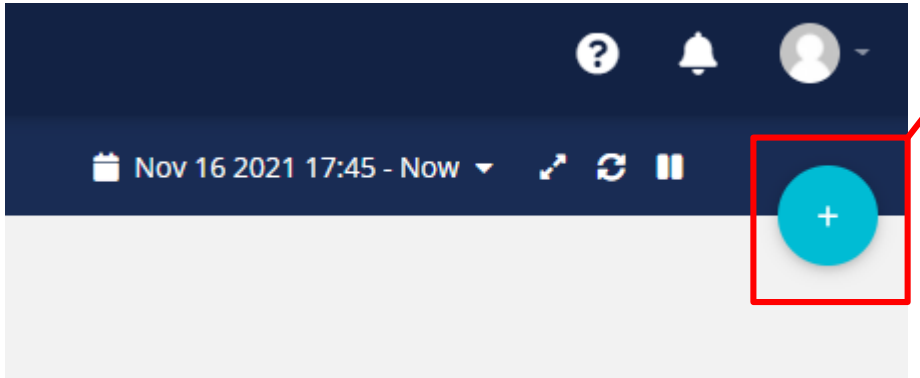
```
pi@rpi4-A00:~/iot/lec06 $ python3 ubidots_http_send.py
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
[INFO] Sending data, attempt number: 0
[INFO] Results:
{"temperature": [{"status_code": 201}]}
```

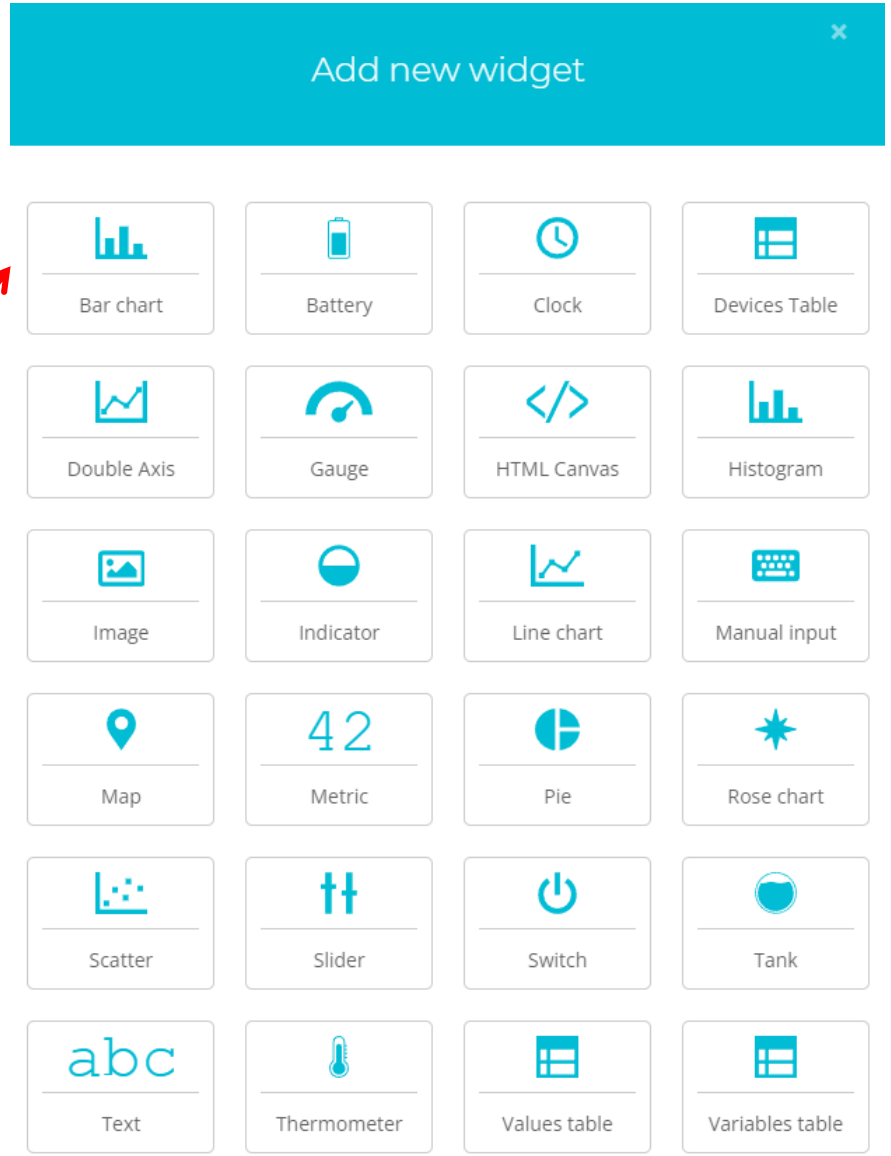
pi@rpi4-A00: ~/iot/lec06

```
pi@rpi4-A00:~/iot/lec06 $ python3 ubidots_http_get.py
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
60.29928194982523
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
91.82705206262754
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
65.85730364930622
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
6.664240612596406
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
23.886499728943356
[INFO] Retrieving data, attempt number: 0
[INFO] Results:
1.0679152957109084
```

Dashboard

- You can also use some widgets to visualize the data on the dashboard page.



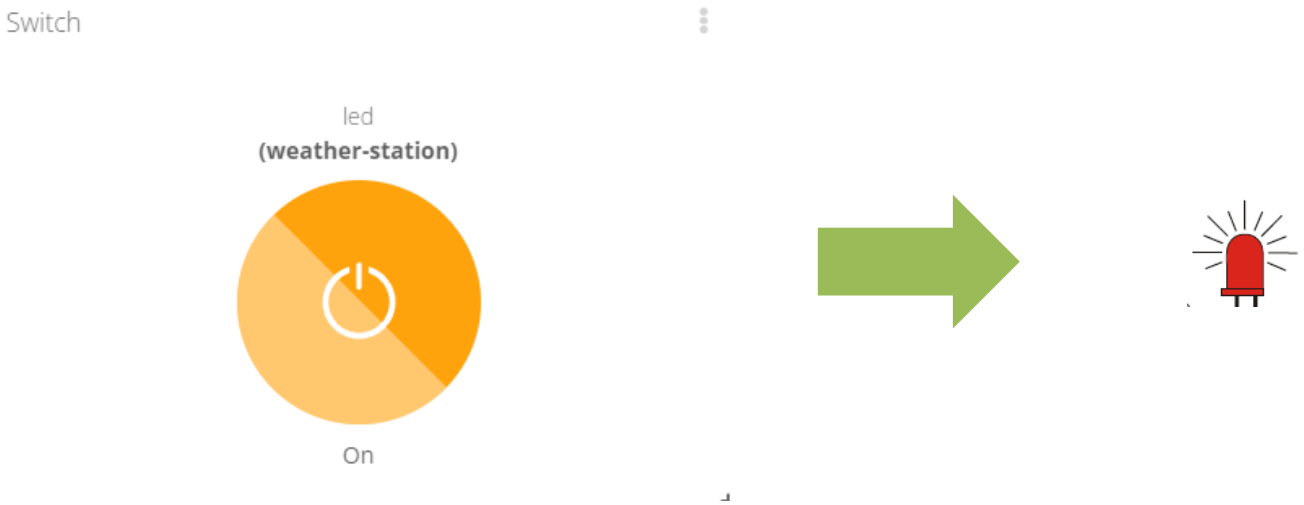


Outline

- Introduction
  - IoT Architecture
  - IoT Networks
  - IoT Platform
- Network-controlled LED
  - TCP socket server
  - HTTP server
- IoT Platform
  - Ubidots
- Lab
  - Remote switch

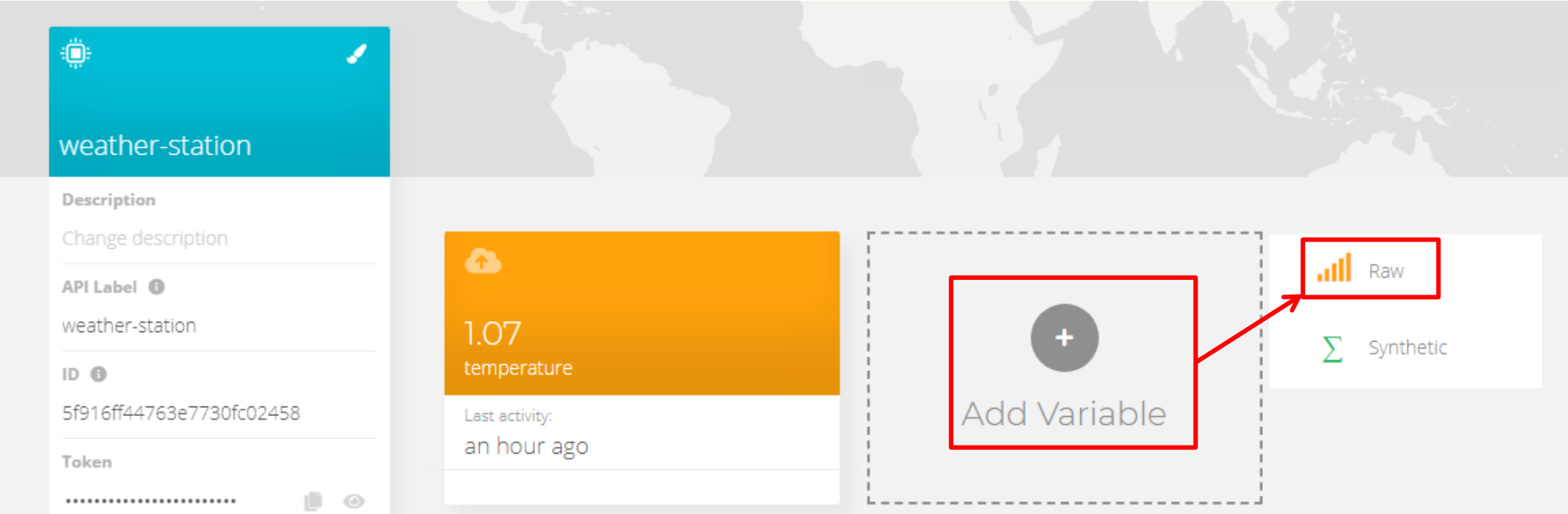
Lab

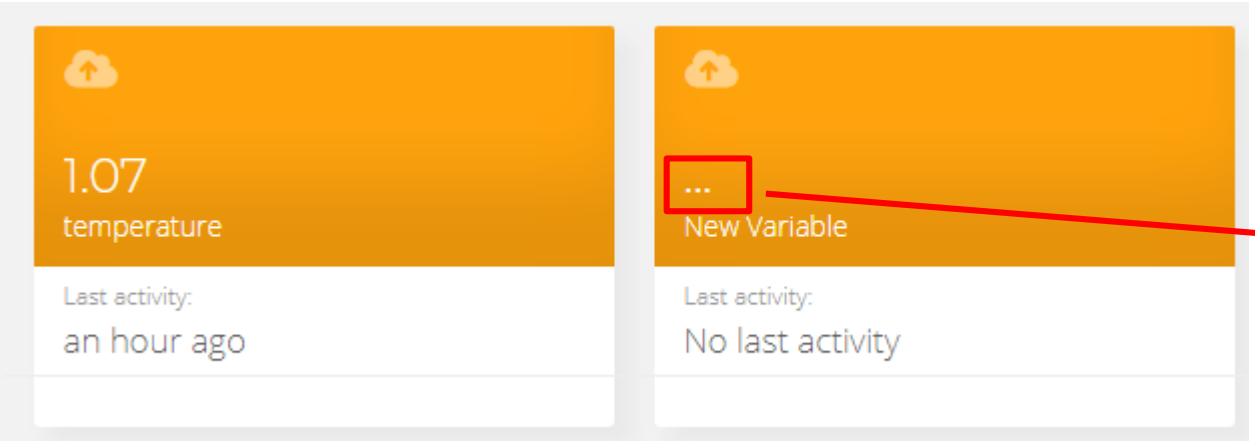
- Remote switch
  - Create a remote switch to control an LED.
  - Use Switch widget on Ubidots.
  - Add a physical button to upload the status onto Ubidots.
  - Synchronize the button and Switch widget for controlling LED.

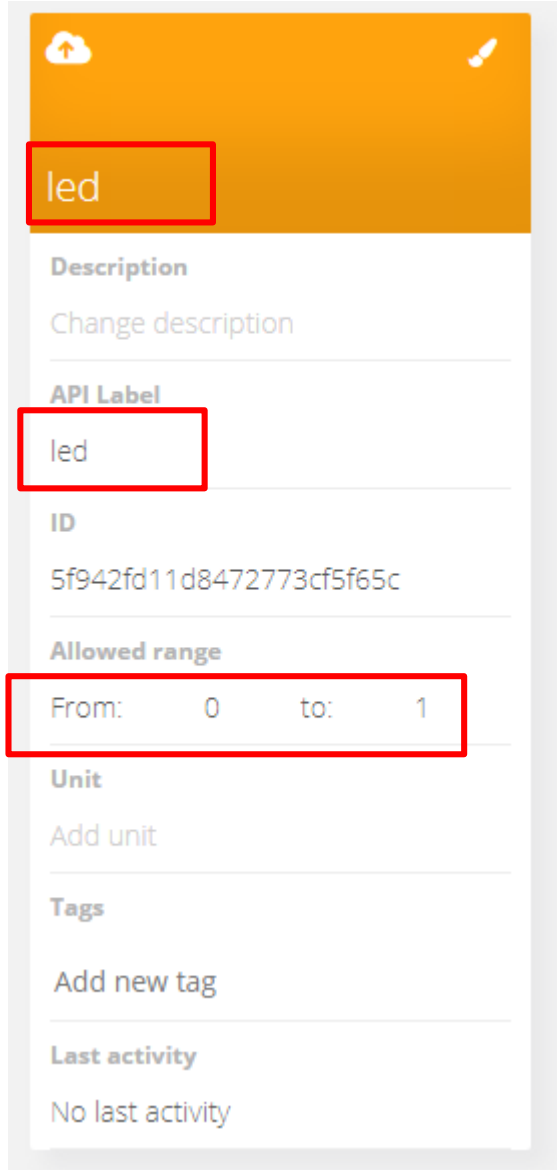


New Variable

- Add a new variable in the device “weather-station”.

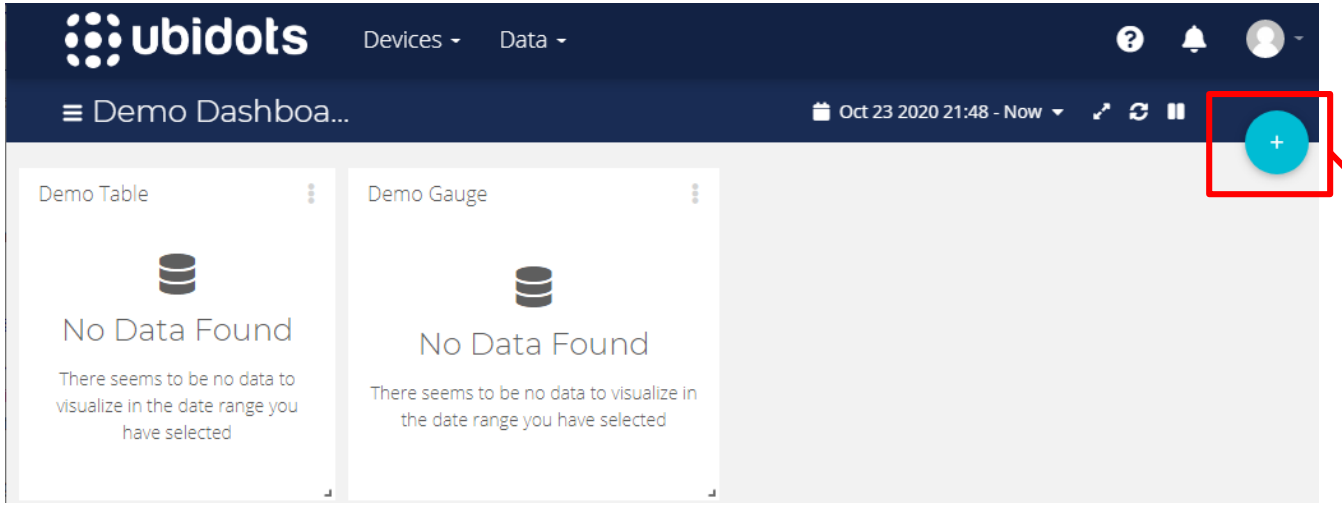


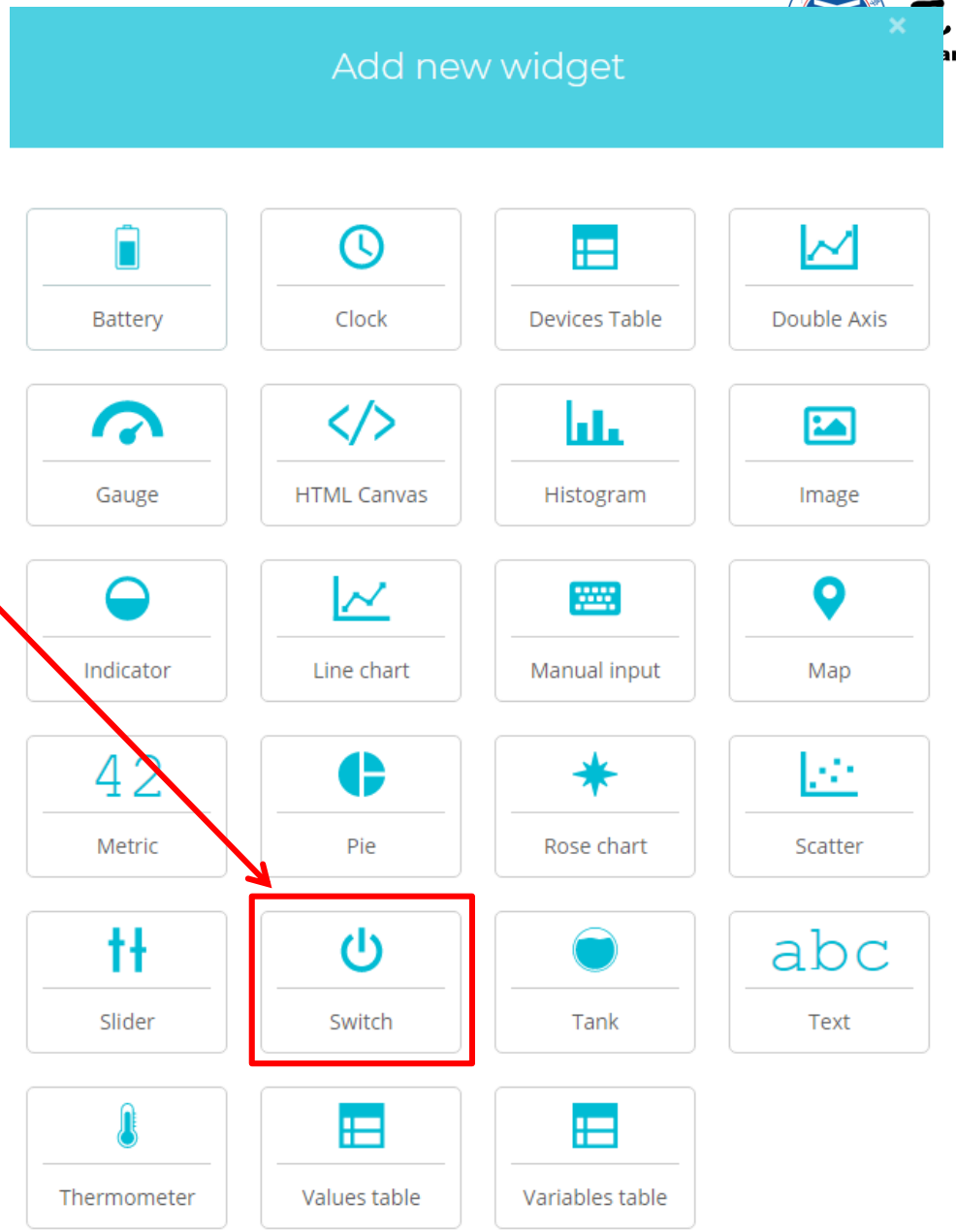

1.07
temperature
Last activity:
an hour ago
...
New Variable
Last activity:
No last activity


led
Description
Change description
API Label
led
ID
5f942fd11d8472773cf5f65c
Allowed range
From: 0 to: 1
Unit
Add unit
Tags
Add new tag
Last activity
No last activity

Widget (1/3)

- @dashboard page





BACK

Data

+ Add Variables

Appearance

Name: Switch

Custom style: Add style

Widget Creation

Select Variables

Search

weather-station — 2 Variables

led

temperature

Switch

BACK

Data

led (weather-station)

Minimum Value: 0

Maximum Value: 1

Off message: Off

On message: On

+ Add Variables

Appearance

Name: Switch

Custom style: Add style

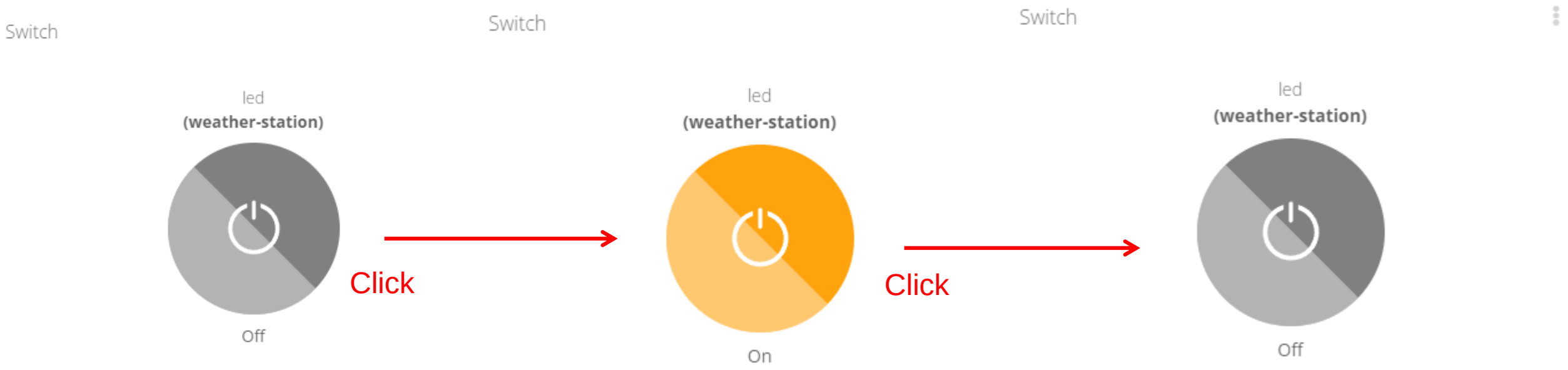
Switch

led
(weather-station)

Off

Widget (3/3)

- Add some values into the variable.



remote_switch.py

$ wget https://raw.githubusercontent.com/yachentw/yzucseiot/main/lec06/remote_switch.py

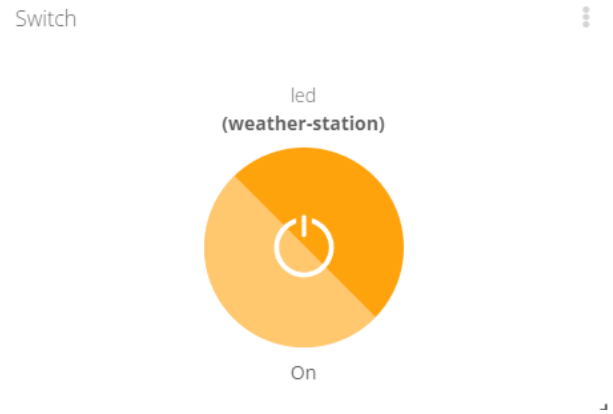
```python
import requests
import time
import RPi.GPIO as GPIO

'''
global variables
'''
LED_PIN = 12
GPIO.setmode(GPIO.BOARD)
GPIO.setup(LED_PIN, GPIO.OUT)

ENDPOINT = "things.ubidots.com"
DEVICE_LABEL = "weather-station"
VARIABLE_LABEL = "led"
TOKEN = ". . ."
DELAY = 0.2  # Delay in seconds
URL = "http://{}/api/v1.6/devices/{}/{}/lv".format(ENDPOINT, DEVICE_LABEL, VARIABLE_
LABEL)
HEADERS = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}

def led(cmd):
    if cmd == 1:
        # print("led on.")
        GPIO.output(LED_PIN, GPIO.HIGH)
    elif cmd == 0:
        # print("led off.")
        GPIO.output(LED_PIN, GPIO.LOW)
```

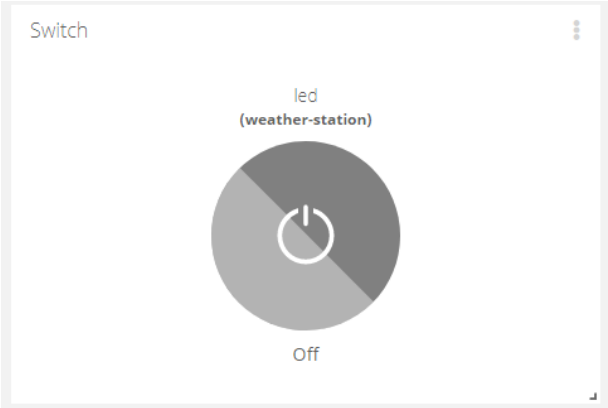
Replace with your token.

```python
def get_var():
    try:
        attempts = 0
        status_code = 400
        while status_code >= 400 and attempts < 5:
            req = requests.get(url=URL, headers=HEADERS)
            status_code = req.status_code
            attempts += 1
            time.sleep(1)
        # print(req.text)
        led(int(float(req.text)))
    except Exception as e:
        print("[ERROR] Error posting, details: {}".format(e))

if __name__ == "__main__":
    try:
        while True:
            get_var()
            time.sleep(DELAY)
    except KeyboardInterrupt:
        print("Exception: KeyboardInterrupt")
    finally:
        GPIO.cleanup()
```

Remote Switch

- Control the LED by the added widget





Your Task

- Wire the button circuit.
- Combine button and ubidots_http_send.py for updating the switch status.
- Run two programs on RPi where one is for getting switch status and the other is updating status.