

# Lab 4

Rafik Zitouni

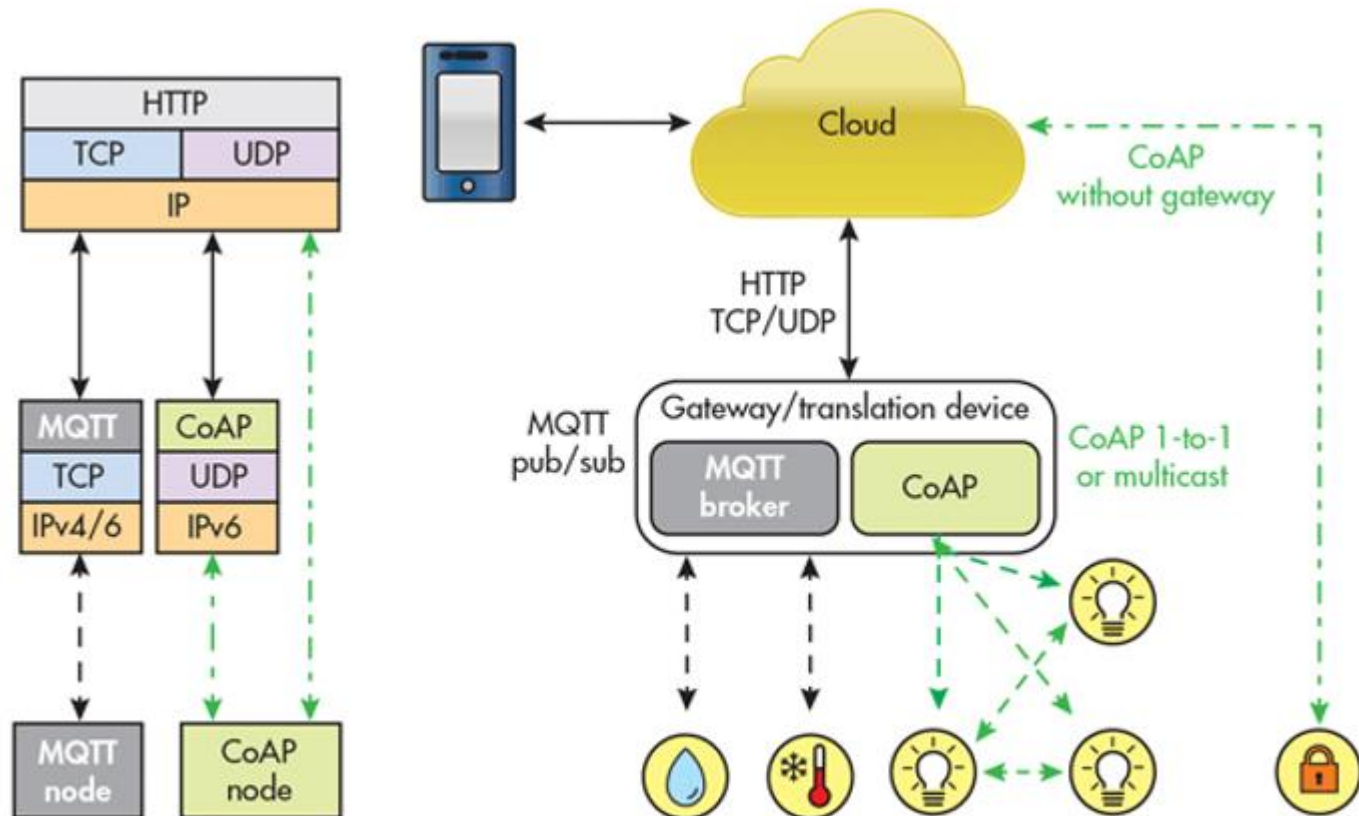
ECE Paris

[rafik.zitouni@ece.fr](mailto:rafik.zitouni@ece.fr)



# REST, CoAP and MQTT

Antonio Liñán, Zolertia. 2016 - CC-NC-SA 4.0



<http://electronicdesign.com/iot/mqtt-and-coap-underlying-protocols-iot>

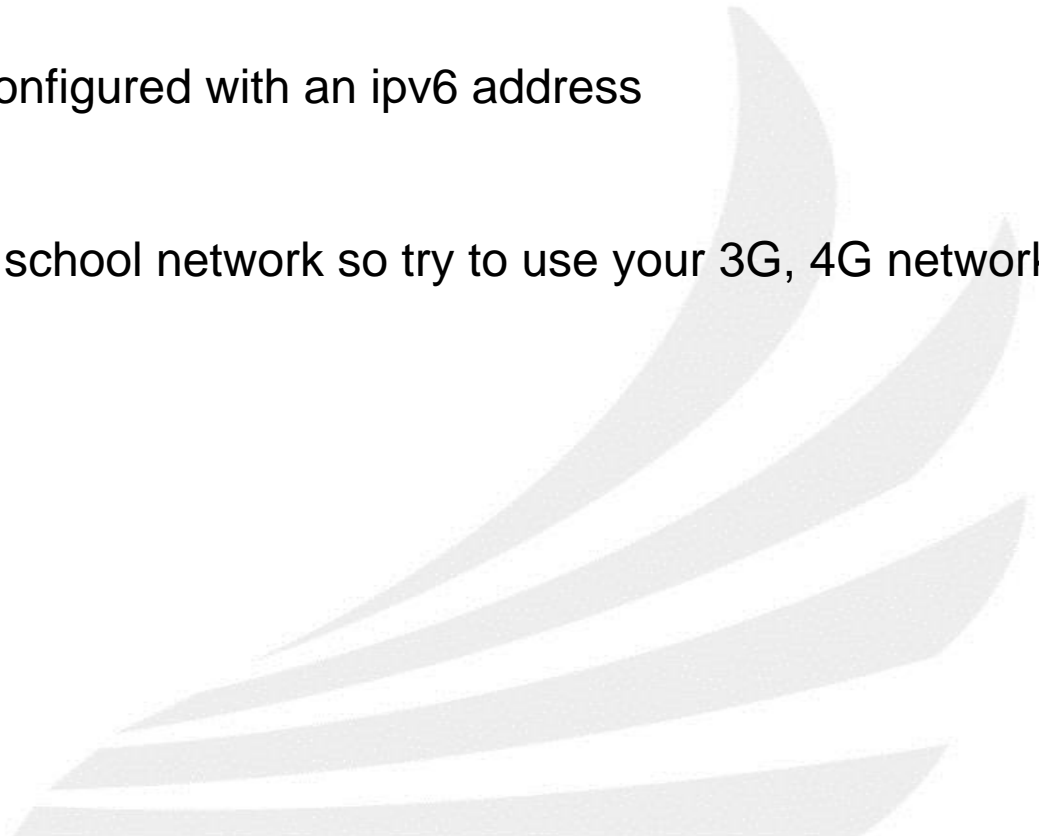
# Connecting to a cloud platform

## UBIDOTS Platform

- You should subscribe to your **things.ubidots.com** platform
- Version Education <https://ubidots.com/education/>
- You can follow these two tutorials :  
<https://www.youtube.com/watch?v=ynhJ4YhhQEM>  
<https://www.youtube.com/watch?v=p3DVe7TLvdl>
- The objective is to demonstrate the basic functionality of Contiki's **Ubidots** library
  - How to use the library to POST to a **variable**
  - How to use the library to POST to a **collection**
  - How to receive (parts of) **the HTTP reply**

# Connecting to a cloud platform

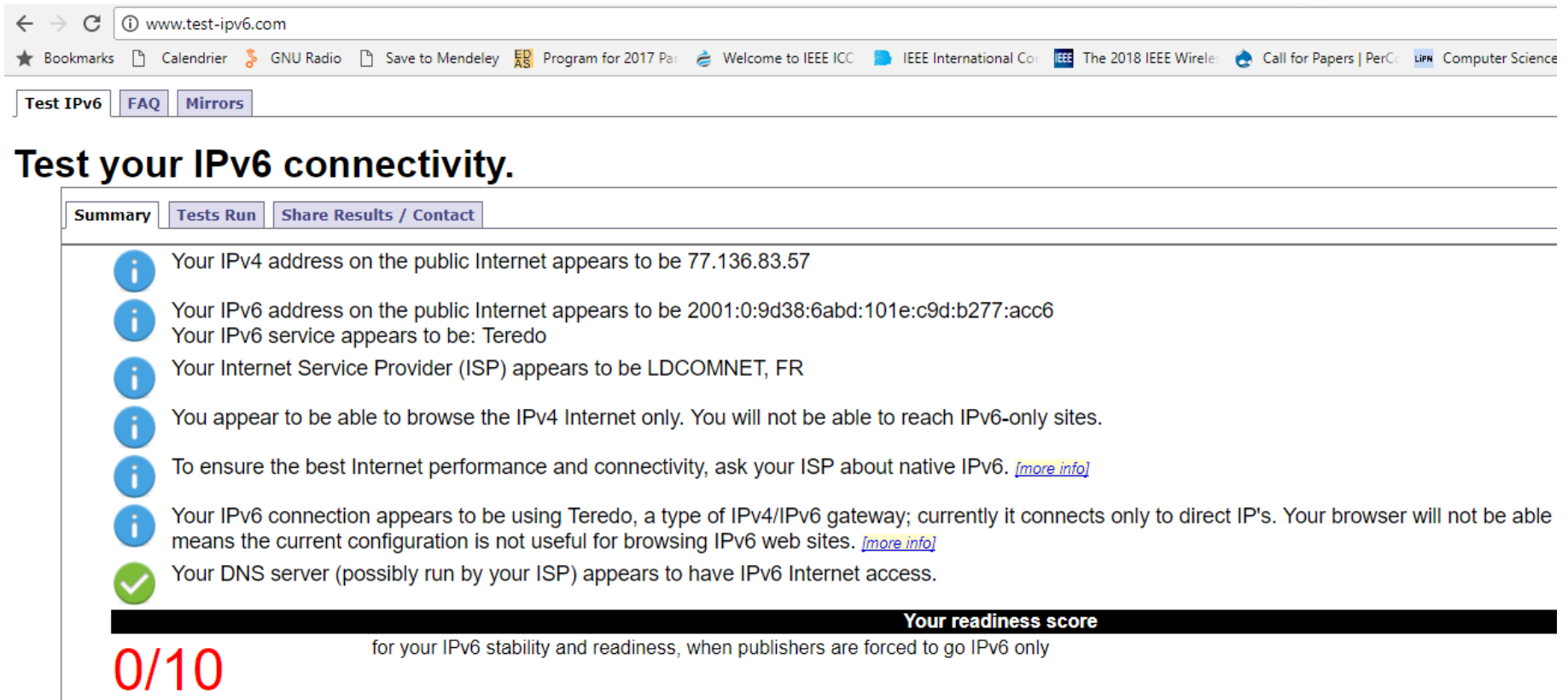
## Example of an application with Ubidots

- Use **miredo** package to allow full ipv6 connectivity to your host  
**\$ sudo apt-get install miredo**
  - You can test if your interface is configured with an ipv6 address  
<http://test-ipv6.com/>
  - If the IPv6 cannot be used in the school network so try to use your 3G, 4G networks
  - Check if you can do ipv6 ping  
**\$ ping6 ipv6.google.com**
- 

# Connecting to a cloud platform

## Example of an application with Ubidots

We can see the obtained local ipv6 address



The screenshot shows a web browser at the URL [www.test-ipv6.com](http://www.test-ipv6.com). The page has navigation links: Test IPv6, FAQ, and Mirrors. The main heading is "Test your IPv6 connectivity." Below this are tabs for Summary, Tests Run, and Share Results / Contact. The Summary tab is active, displaying a list of test results:

- Your IPv4 address on the public Internet appears to be 77.136.83.57
- Your IPv6 address on the public Internet appears to be 2001:0:9d38:6abd:101e:c9d:b277:acc6
- Your IPv6 service appears to be: Teredo
- Your Internet Service Provider (ISP) appears to be LDCOMNET, FR
- You appear to be able to browse the IPv4 Internet only. You will not be able to reach IPv6-only sites.
- To ensure the best Internet performance and connectivity, ask your ISP about native IPv6. [\[more info\]](#)
- Your IPv6 connection appears to be using Teredo, a type of IPv4/IPv6 gateway; currently it connects only to direct IP's. Your browser will not be able means the current configuration is not useful for browsing IPv6 web sites. [\[more info\]](#)
- Your DNS server (possibly run by your ISP) appears to have IPv6 Internet access.

At the bottom, a black bar displays "Your readiness score" followed by a large red "0/10" and the text "for your IPv6 stability and readiness, when publishers are forced to go IPv6 only".

# Connecting to a cloud platform

## Example of an application with Ubidots

➤ Ubidots uses TCP sockets to connect to the host **things.ubidots.com**. You can get the IPv6 address of **ubidots** platform from <https://centralops.net/co/>



The screenshot shows the 'Domain Dossier' interface on CentralOps.net. On the left is a sidebar with 'Utilities' including Domain Dossier, Domain Check, Email Dossier, Browser Mirror, Ping, Traceroute, NsLookup, AutoWhois, TcpQuery, and AnalyzePath. The main area is titled 'Domain Dossier' with the subtitle 'Investigate domains and IP addresses'. A search bar contains 'things.ubidots.com'. Below it are checkboxes for 'domain whois record', 'DNS records', 'network whois record', 'service scan', and 'traceroute'. A 'go' button is present. Below the search bar, it shows 'user: anonymous [91.199.6.245]' and 'balance: 49 units' with links for 'log in' and 'account info'. The 'Address lookup' section shows the canonical name 'things.ubidots.com.', aliases, and addresses '2607:f0d0:2101:39::2' and '50.23.124.68'. The 'Domain Whois record' section is partially visible at the bottom.

**Utilities**

- Domain Dossier
- Domain Check
- Email Dossier
- Browser Mirror
- Ping
- Traceroute
- NsLookup
- AutoWhois
- TcpQuery
- AnalyzePath

**Domain Dossier** Investigate domains and IP addresses

domain or IP address

☒ domain whois record ☒ DNS records ☐ traceroute

☒ network whois record ☐ service scan

user: anonymous [91.199.6.245]  
balance: 49 units  
[log in](#) | [account info](#)

**Address lookup**

canonical name **things.ubidots.com.**

aliases

addresses **2607:f0d0:2101:39::2**  
**50.23.124.68**

**Domain Whois record**

IPv6 of ubidots server is: **2607:f0d0:2101:39::2**

# Connecting to a cloud platform

## Example of an application with Ubidots

- This application has been developed by **G. Oikonomou** (<https://github.com/g-oikonomou/contiki/tree/ubidots-demo>)
- The source code of the **/contiki/examples/ipv6/ubidots** is **ubidots-demo.c**
- Before compilation check if you have the folder in **/contiki/app/ubidots**  
**sudo make ubidots-demo.upload TARGET=zoul**

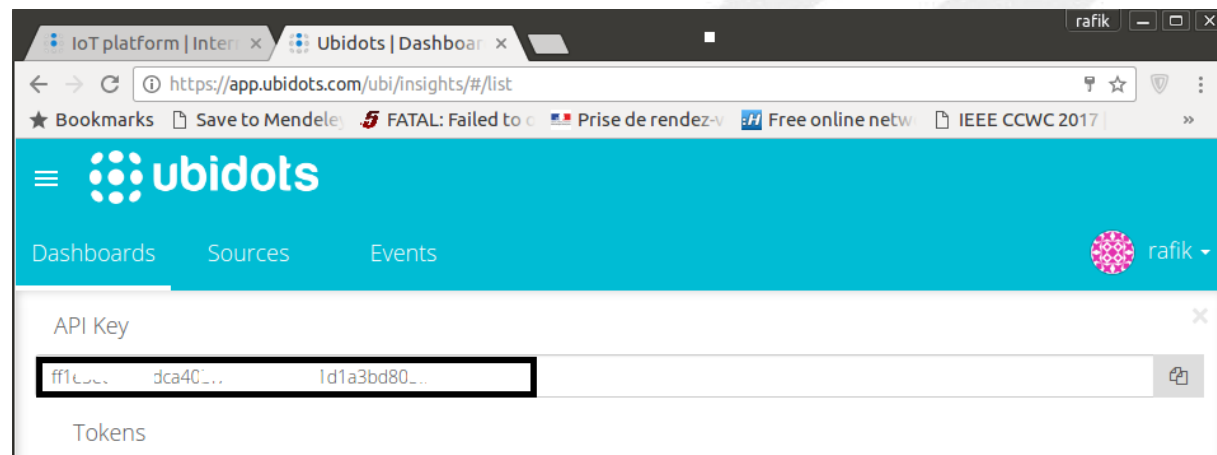
Check if you have the following instruction in your **project-conf.h**

```
#define UBIDOTS_CONF_REMOTE_HOST "2607:f0d0:2101:39::2"
```

IPv6 of ubidots server

The Ubidots demo posts every 30 seconds the zoul mote's **uptime** and **sequence number**, so we need to create these two variables at Ubidots.

Login or subscribe for an **ubidots** account. The most important information to keep confidential is the API Key.



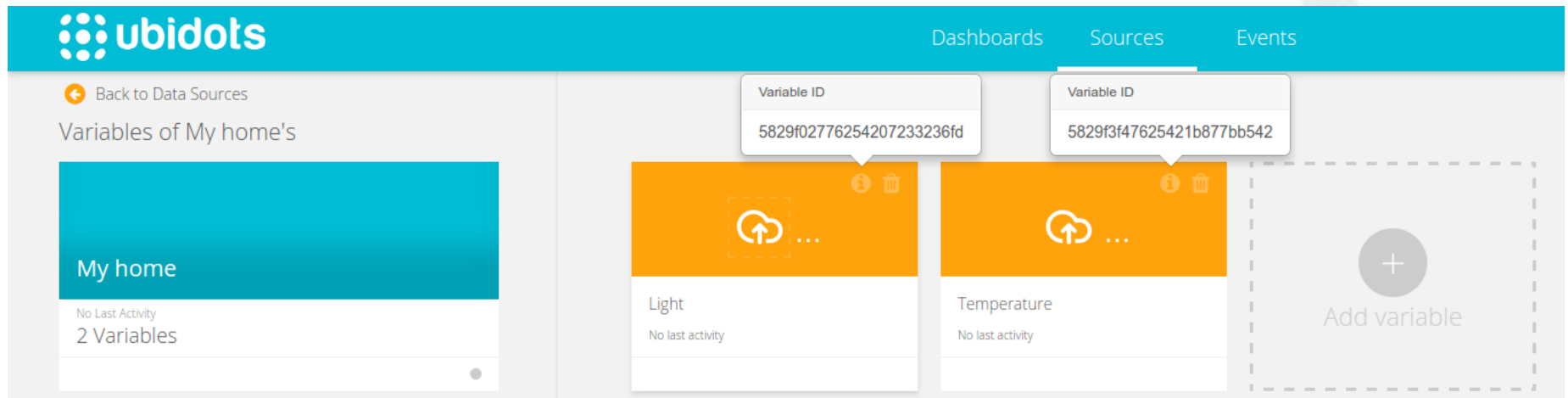


# Connecting to a cloud platform

## Example of an application with Ubidots

- Copy/Past the **API key** of your Ubidots API. It defines the value of the parameter

**UBIDOTS\_CONF\_AUTH\_TOKEN** in **project-conf.h**



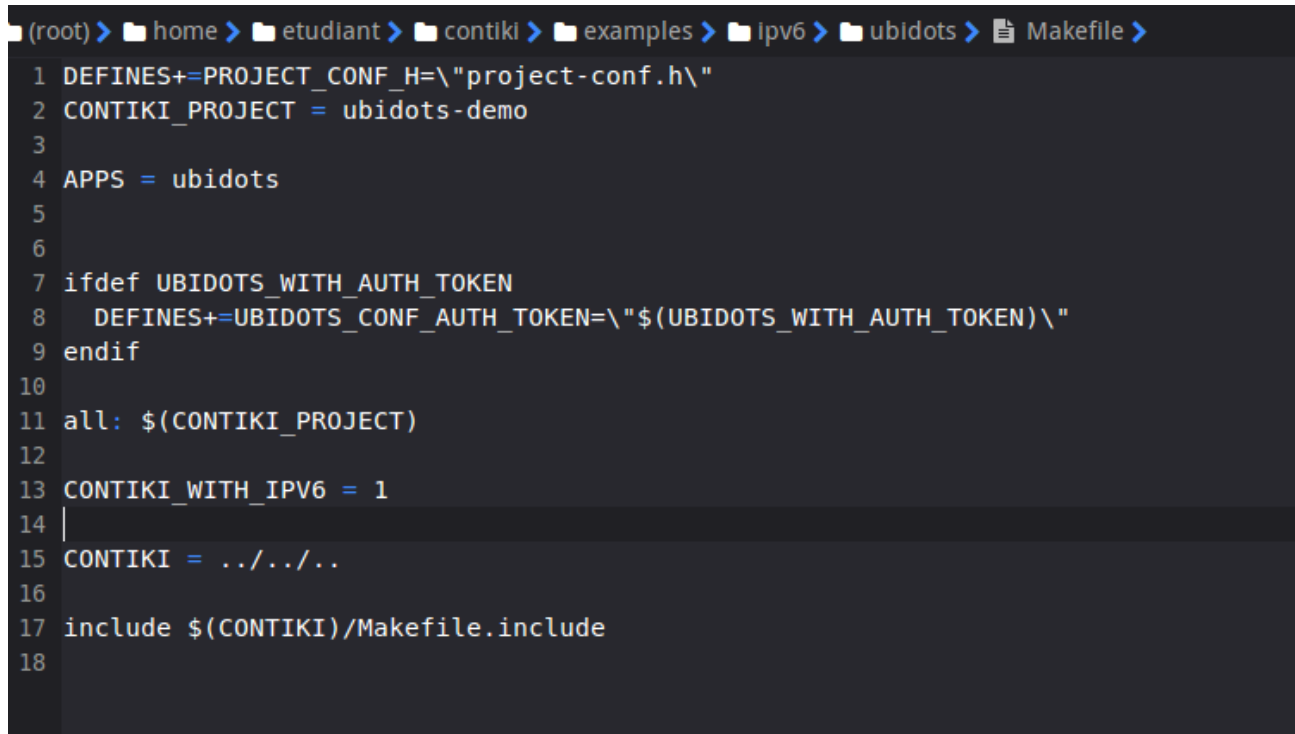
- Copy the **Variables IDs** which correspond to  
**#define UBIDOTS\_DEMO\_CONF\_UPTIME** “ ”  
**#define UBIDOTS\_DEMO\_CONF\_SEQUENCE** “ ”

Of course, you add these variables in your **project-conf.h**

# Connecting to a cloud platform

## Example of an application with Ubidots

- Check if your **Makefile** looks like the following screenshot:



```
(root) > home > etudiant > contiki > examples > ipv6 > ubidots > Makefile >
1  DEFINES+=PROJECT_CONF_H=\"project-conf.h\"
2  CONTIKI_PROJECT = ubidots-demo
3
4  APPS = ubidots
5
6
7  ifdef UBIDOTS_WITH_AUTH_TOKEN
8      DEFINES+=UBIDOTS_CONF_AUTH_TOKEN=\"$(UBIDOTS_WITH_AUTH_TOKEN)\"
9  endif
10
11 all: $(CONTIKI_PROJECT)
12
13 CONTIKI_WITH_IPV6 = 1
14 |
15 CONTIKI = ../../..
16
17 include $(CONTIKI)/Makefile.include
18
```

- Compile the project

# Connecting to a cloud platform

## Example of an application with Ubidots

- Compile and program a Border Router device as shown in the previous Labs.
- Verify the Border Router is online by making a ping6 request to it.
- Browse the Border Router's web service and also ping6 the **ubidots-demo** node

```
Ubidots client: HTTP Reply 200
HTTP Status: 200
Ubidots client: New header: <Server: nginx>
Ubidots client: New header: <Date: Fri, 13 Mar 2015 09:35:08 GMT>
Ubidots client: New header: <Content-Type: application/json>
Ubidots client: New header: <Transfer-Encoding: chunked>
Ubidots client: New header: <Connection: keep-alive>
Ubidots client: New header: <Vary: Accept-Encoding>
Ubidots client: Client wants header 'Vary'
H: 'Vary: Accept-Encoding'
Ubidots client: New header: <Vary: Accept>
Ubidots client: Client wants header 'Vary'
H: 'Vary: Accept'
Ubidots client: New header: <Allow: GET, POST, HEAD, OPTIONS>
Ubidots client: Chunk, len 22: <[{"status_code": 201}]> (counter = 22)
Ubidots client: Chunk, len 0: <(End of Reply)> (Payload Length 22 bytes)
P: '[{"status_code": 201}]'
```

# Connecting to a cloud platform

## Example of an application with Ubidots

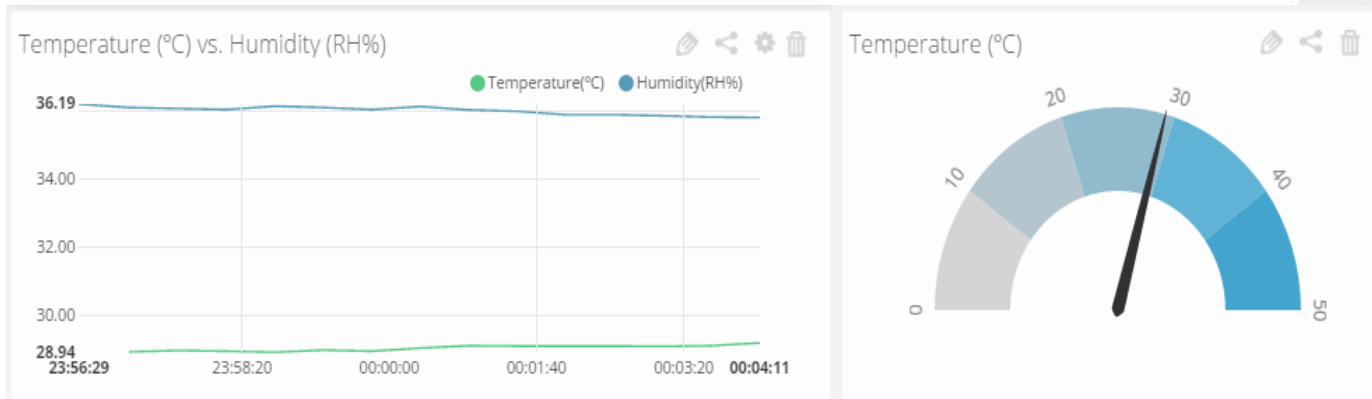
- After fixing the missing in your source code, you should obtain the following information after pushing the **reset button**.

```
rafik@rafik-Precision-7510:~/enseignement/OCRES/contiki/examples/ipv6/ubidots$ sudo make login PORT=/dev/ttyUSB1
Using saved target 'zoul'
./../../tools/sky/serialdump-linux -b115200 /dev/ttyUSB1
Connecting to /dev/ttyUSB1 (115200) [OK]
Contiki-2.6-4018-gcdf76cb
Zolertia RE-Mote revision B platform
CC2538: ID: 0xb964, rev.: PG2.0, Flash: 512 KiB, SRAM: 32 KiB, AES/SHA: 1, ECC/RSA: 1
System clock: 16000000 Hz
I/O clock: 16000000 Hz
Reset cause: External reset
Time configured with address 00:12:4b:00:06:0d:b3:ef
Net: sicslowpan
MAC: CSMA
RDC: ContikiMAC
Ubidots client: STATE_ERROR_NO_NET
Server IPv6 addresses: [fd00:0000:0000:0000:0212:4b00:060d:b3ef]
[fe80:0000:0000:0000:0212:4b00:060d:b3ef]
Ubidots client: STATE_ERROR_NO_NET
Ubidots client: STATE_STARTING
Ubidots client: Checking 2607:f0d0:2101:39::2
Ubidots client: 'Host: [2607:f0d0:2101:39::2]' (remaining 42)
Ubidots client: STATE_TCP_CONNECT (1)
Ubidots client: Connect 2607:f0d0:2101:39::2 port 80
```

# Connecting to a cloud platform

## Example of an application with Ubidots

- If you are able to exchange real-world data with your network, you should obtain these curves.



- If you have problems to obtain such results see the following WIKI, Section **Hands on: connecting to a real world IoT platform (HTTP-based)**  
<https://github.com/marcozennaro/IPv6-WSN-book/blob/master/5.asc#hands-on-connecting-to-a-real-world-iot-platform-http-based>

# Connecting to a cloud platform

- It is possible to use python middleware to push/pull data to/from cloud platform:

```
1 import time
2 import requests
3 import math
4 import random
5 import json
6
7 TOKEN = "A1E-qp6nRZ0dBrLxhRXTb58PZPeHumsBoZ" # Put your TOKEN here
8 DEVICE_LABEL = "machine" # Put your device label here
9 VARIABLE_LABEL_1 = "temperature" # Put your first variable label here
10 VARIABLE_LABEL_2 = "humidity" # Put your second variable label here
11 VARIABLE_LABEL_3 = "position" # Put your second variable label here
12
13 def build_payload(variable_1, variable_2, variable_3):
14     # Creates two random values for sending data
15     value_1 = random.randint(-10, 50)
16     value_2 = random.randint(0, 85)
17
18     # Creates a random gps coordinates
19     lat = random.randrange(34, 36, 1) + \
20         random.randrange(1, 1000, 1) / 1000.0
21     lng = random.randrange(-83, -87, -1) + \
22         random.randrange(1, 1000, 1) / 1000.0
23     payload = {variable_1: value_1,
24               variable_2: value_2,
25               variable_3: {"value": 1, "context": {"lat": lat, "lng": lng}}}
26
27     return payload
28
29 def post_request(payload):
30     # Creates the headers for the HTTP requests
31     url = "http://things.ubidots.com"
32     url = "{}api/v1.6/devices/{}".format(url, DEVICE_LABEL)
33     headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
34
```

```
def post_request(payload):
    # Creates the headers for the HTTP requests
    url = "http://things.ubidots.com"
    url = "{}api/v1.6/devices/{}".format(url, DEVICE_LABEL)
    headers = {"X-Auth-Token": TOKEN, "Content-Type": "application/json"}
    # Makes the HTTP requests
    status = 400
    attempts = 0
    while status >= 400 and attempts <= 5:
        req = requests.post(url=url, headers=headers, data=json.dumps(payload))
        status = req.status_code
        attempts += 1
        time.sleep(1)
    # Processes results
    if status >= 400:
        print("[ERROR] Could not send data after 5 attempts, please check \
your token credentials and internet connection")
        return False
    print("[INFO] request made properly, your device is updated")
    return True

def main():
    payload = build_payload(
        VARIABLE_LABEL_1, VARIABLE_LABEL_2, VARIABLE_LABEL_3)
    print("[INFO] Attempting to send data")
    post_request(payload)
    print("[INFO] finished")
    |
if __name__ == '__main__':
    while (True):
        main()
        time.sleep(1)
```

The End...

Thank you so much

