

# THE THINGS NETWORK.CAT

TALLER BÀSIC - FEM UN TERMÒSTAT AMB UNA LOPY4

**Xarxa comunitària per  
l'Internet de les Coses**

**@ttncat**  
thethingsnetwork.cat

**@thethingsntwrk**  
thethingsnetwork.org

## Introducció

CONSOLA

# CONSOLA TTN



Des de la consola de TTN podem **gestionar les passarel·les i aplicacions**. També podem monitoritzar el tràfic que passa o arriba a qualsevol de les dues i veure els continguts si tenim les claus de desxifratge.

Els nodes (*devices*) estan associats a les aplicacions. Podem donar-ne d'alta, configurar el mode en que es connectaran a la xarxa (OTAA o ABP), veure els missatges que envien o enviar nosaltres un missatge a qualsevol dispositiu.

També des de l'aplicació podem fer integracions amb diferents serveis *online* com ara IFTTT.

The screenshot shows the TTN Console Community Edition interface. At the top, there's a navigation bar with 'CONSOLE COMMUNITY EDITION', 'Applications', 'Gateways', and a user profile 'xoseperez'. Below this, the breadcrumb 'Applications > ttncat-taller' is visible. A horizontal menu contains 'Overview', 'Devices', 'Payload Formats', 'Integrations', 'Data', and 'Settings'. The main content area is titled 'APPLICATION OVERVIEW' and includes a 'documentation' link. It lists the following details for the application 'ttncat-taller':

- Application ID:** ttncat-taller
- Description:** Aplicatiu de test per els tallers de TTN.cat
- Created:** 2 days ago
- Handler:** ttn-handler-eu (current handler)

Below this, there's a section for 'APPLICATION EUIs' with a 'manage euis' link. A hexagonal EUI value '70 B3 D5 7E D0 00 9F 0C' is displayed with navigation icons. At the bottom, the 'DEVICES' section has 'register device' and 'manage devices' links.

# COMPTE PER EL TALLER

**URL:** console.thethingsnetwork.org

**User:** ttncat-tallers

**Clau:** ttncat-tallers

A screenshot of the login page for The Things Network console. The page has a white background with a blue logo at the top center. Below the logo, there are navigation links for 'HOME' and 'CONSOLE'. The main content area contains a login form with two input fields: 'EMAIL OR USERNAME' and 'PASSWORD'. A green 'Log in' button is positioned to the right of the password field. At the bottom of the form, there are links for 'Forgot your password?' and 'Create an account'. The footer of the page contains the text 'You are the network. Let's build this thing together. - The Things Network' and links for 'Terms and Conditions' and 'Privacy Policy'.

# CONSOLA TTN



Cada dispositiu té unes claus per accedir a la xarxa (**network session key**) i connectar-se amb un aplicatiu (**application session key**).

Aquestes claus es poden gravar al firmware (**ABP, Activation by Personalisation**) o negociar-se en el moment de fer el *join* (**OTAA, Over the Air Activation**).

Amb OTAA el dispositiu caldrà estar donat d'alta a l'aplicatiu (**Device EUI**).

Applications > pyladies-20181023 > Devices > ttn-lopy4-01

Overview Data Settings

### DEVICE OVERVIEW

Application ID `pyladies-20181023`

Device ID `ttn-lopy4-01`

Activation Method `OTAA`

Device EUI `<> 70 B3 D5 49 9C 47 23 E8`

Application EUI `<> 70 B3 D5 7E D0 01 3A 9B`

App Key `<> [REDACTED]`

Status `never seen`

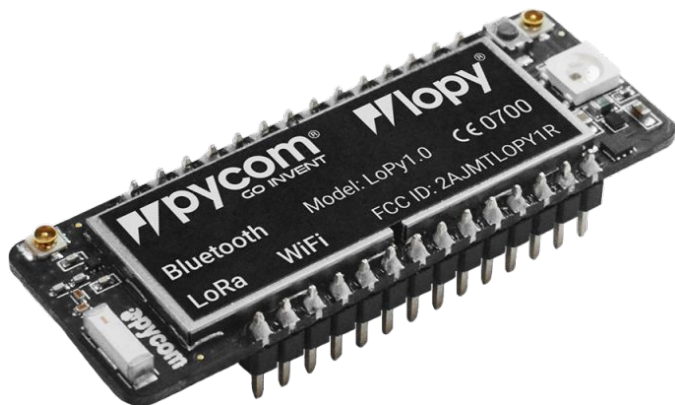
Frames up 0 [reset frame counters](#)

Frames down 0

HARDWARE



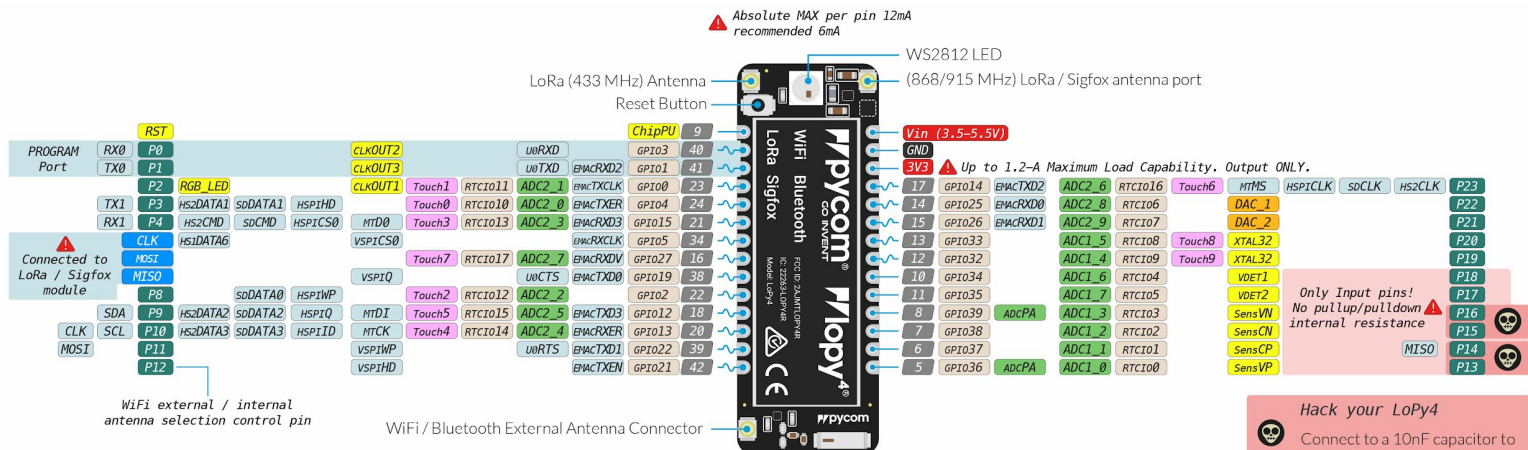
# HARDWARE (LOPY / LOPY4)



- Espressif ESP32. Xtensa dual-core 32-bit LX6 microprocessor, up to 240 MHz
- 512Kb RAM, 4MB external flash
- 802.11 b/g/n 16mbps
- Bluetooth LE and classic
- 868 LoRa +14dBm, LoRaWan stack class A&C
- 2xUART, 2xSPI, I2C, I2S, micro SD card
- 8x 12bits ADCs
- 4x 16bits Timers
- 24x GPIOs
- Ethernet MAC
- SSL/TLS support
- WPA Enterprise security
- FCC & CE certs



# LOPY4



⚠ Absolute MAX per pin 12mA recommended 6mA

WS2812 LED

(868/915 MHz) LoRa / Sigfox antenna port

Vin (3.5-5.5V)

GND

3V3

⚠ Up to 1.2-A Maximum Load Capability. Output ONLY.

⚠ Connected to LoRa / Sigfox module

WiFi external / internal antenna selection control pin

WiFi / Bluetooth External Antenna Connector

Only Input pins!  
No pullup/pulldown internal resistance

**Hack your LoPy4**  
Connect to a 10nF capacitor to enable Touch Pin function

- Power
- GND
- Serial Pin
- Analog Pin
- Control
- Physical Pin
- Port Pin
- Touch Pin
- DAC Pin
- ~ PWM Pin

### Low Level Bootloader

P2 + GND

### Boot modes and safe boot

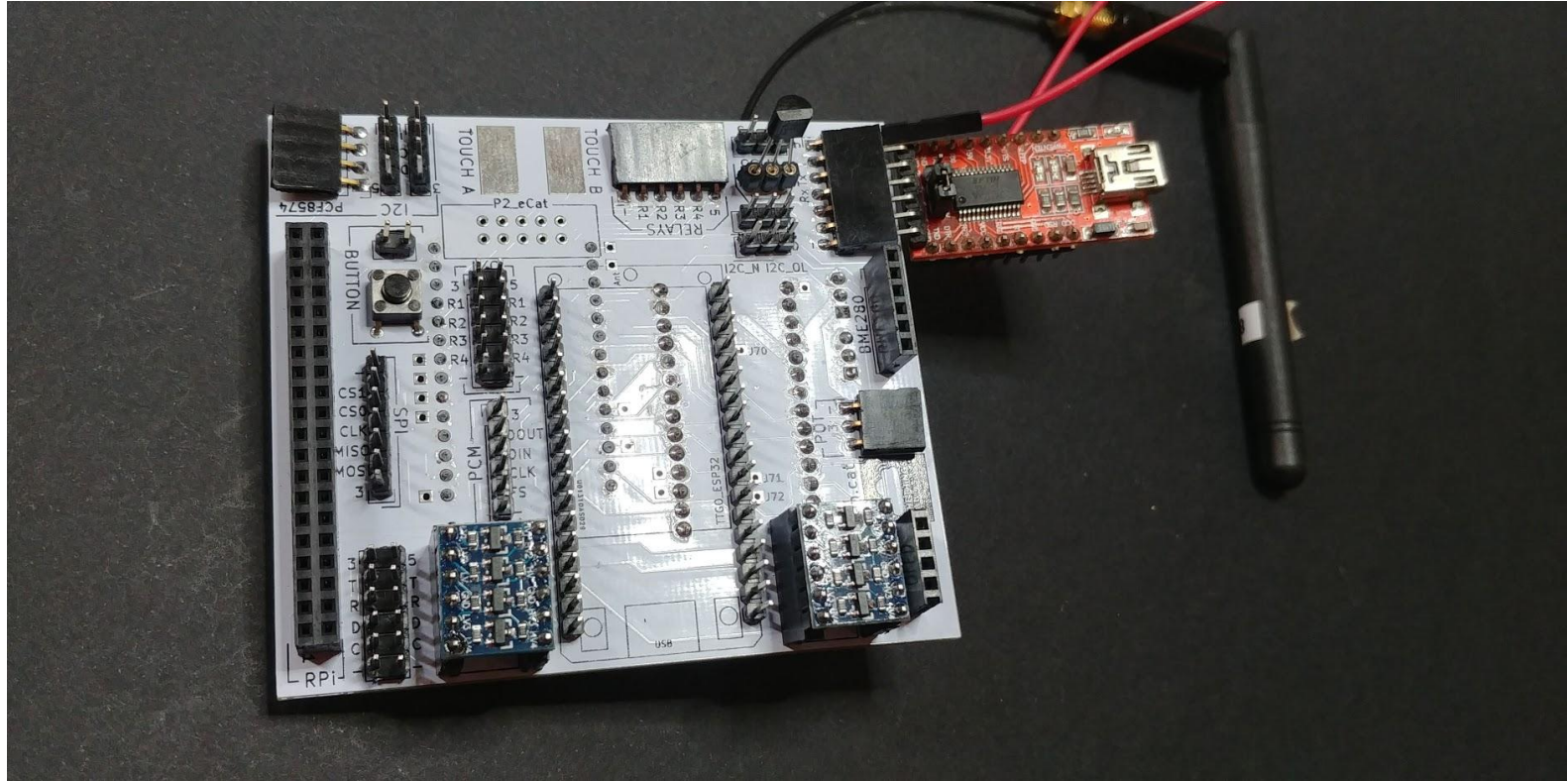
P12 + 3V3

- 1-3 sec Safe boot, latest firmware is selected
- 4-6 sec Safe boot, previous user update selected
- 7-9 sec Safe boot, the factory firmware is selected

### Internal Functions

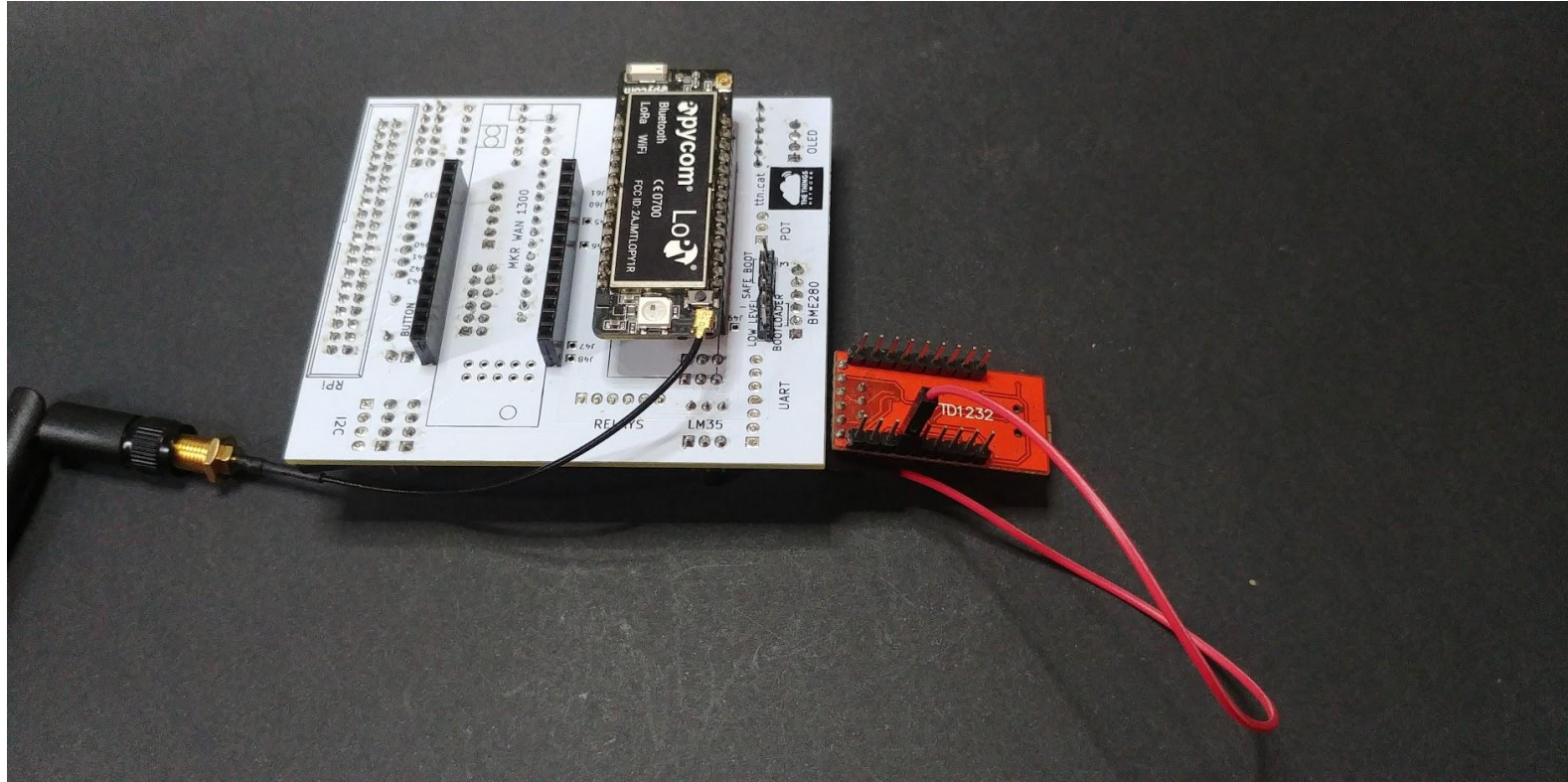
36	GPIO23	VSPID	HSTROBE	LoRa / Sigfox Interrupt
27	GPIO18	EMCCO180	U0TXD	LoRa / Sigfox Select

# CARRIER BOARD

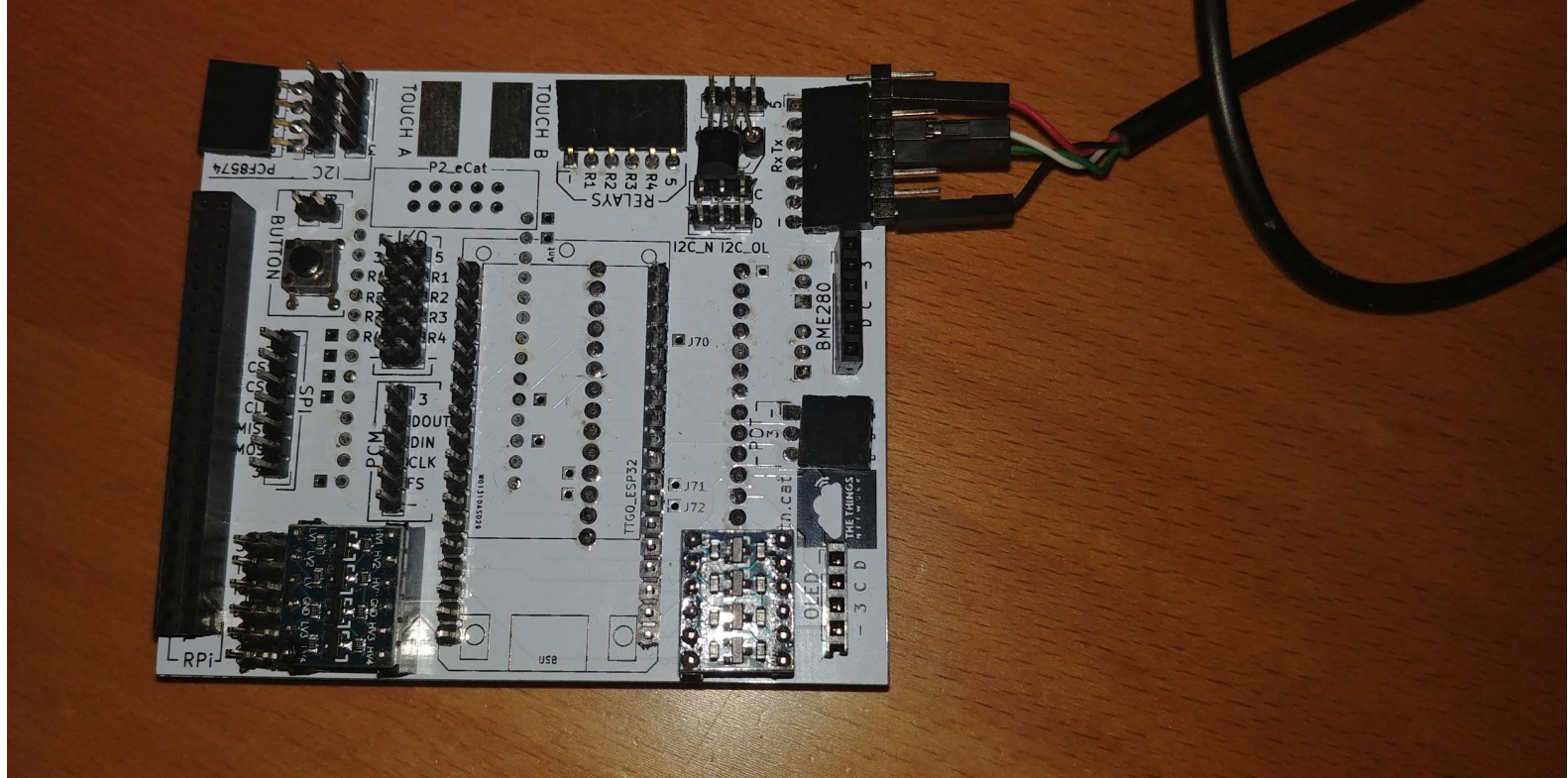




# CARRIER BOARD

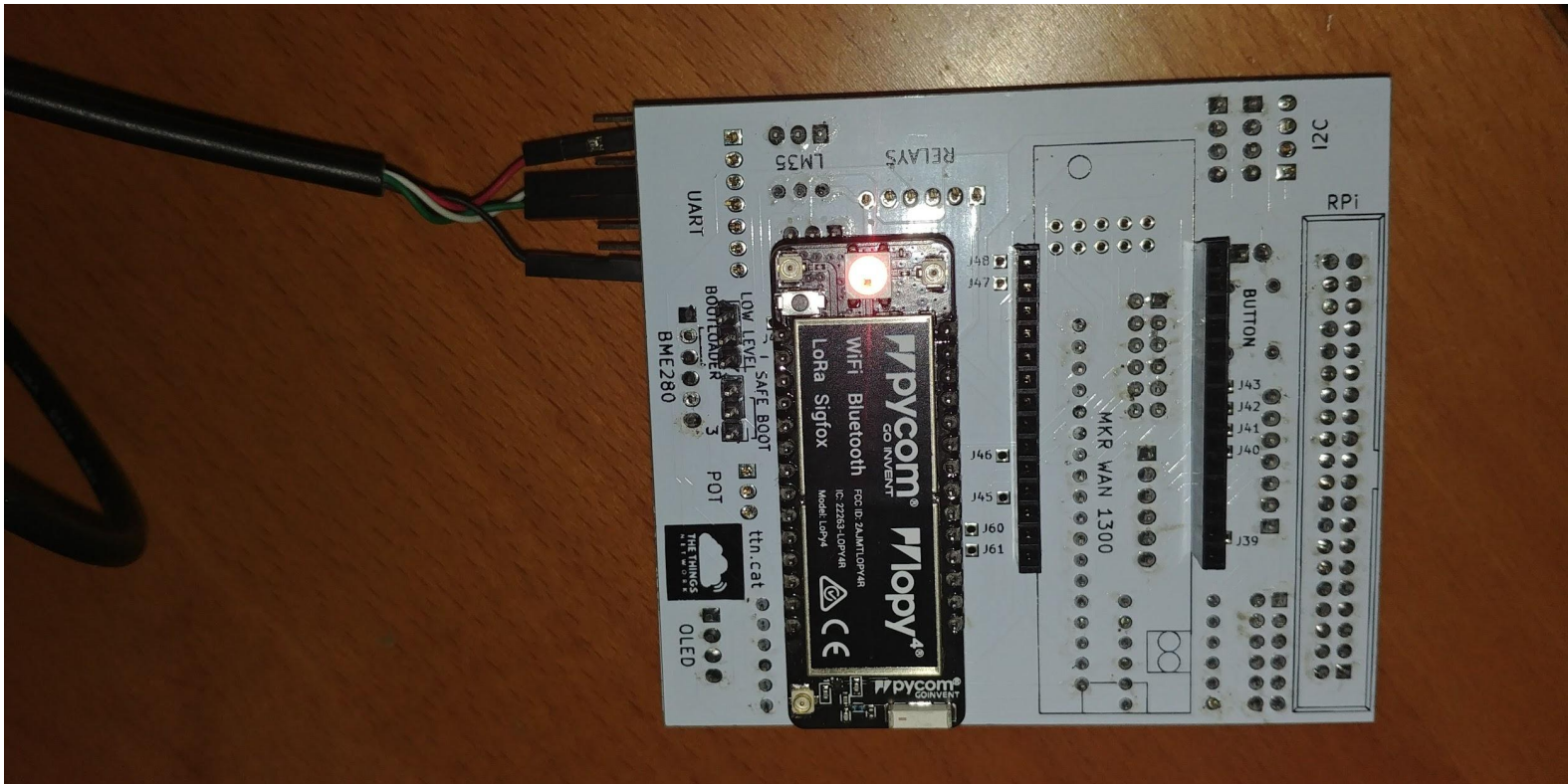


# CARRIER BOARD





# CARRIER BOARD



SOFTWARE



# MICROPYTHON BY PYCOM



<https://micropython.org/>

<https://github.com/micropython/micropython>

<https://docs.pycom.io/>

<https://github.com/pycom/pycom-micropython-sigfox>

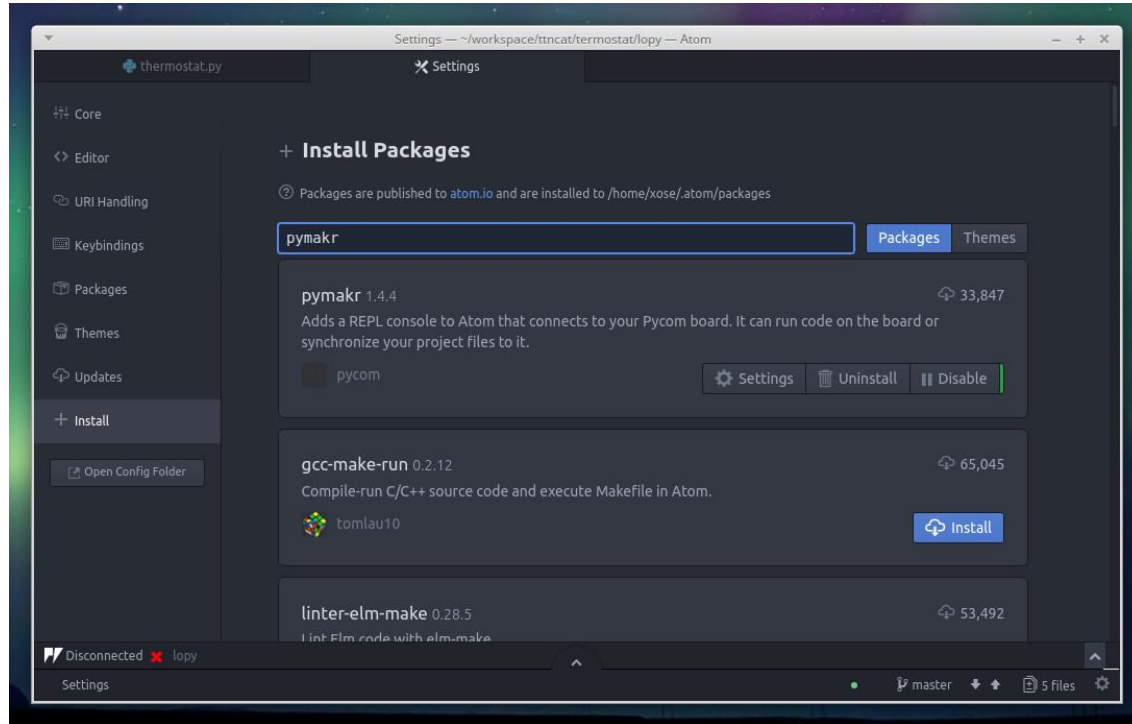
“MicroPython is intended for constrained environments, in particular, microcontrollers, which have orders of magnitude less performance and memory than “desktop” systems on which CPython3 runs.”

```
leds.py
1 import pycom
2 import time
3
4 pycom.heartbeat(False)
5
6 while True:
7     pycom.rgbled(0xFF0000) # Red
8     time.sleep(1)
9     pycom.rgbled(0x00FF00) # Green
10    time.sleep(1)
11    pycom.rgbled(0x0000FF) # Blue
12    time.sleep(1)
13
```

```
Connected to thermostat
Connecting on /dev/ttyUSB0...

>>> import machine
>>> pin = machine.Pin("G10", mode=machine.Pin.OUT)
>>> pin.value(True)
>>> pin()
1
>>> pin(False)
>>> pin()
0
>>>
```

# ATOM - INSTAL·LAR PYMAKR



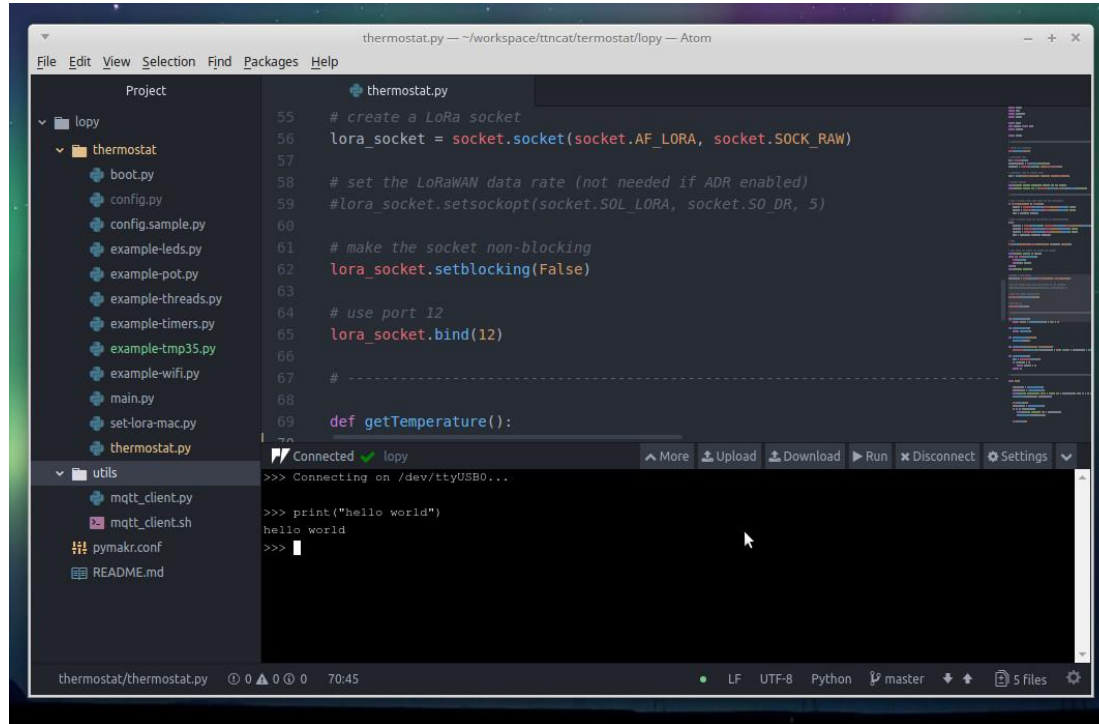
# ATOM - PYMAKR



```
Project — ~/workspace/tncat/thermostat/lopy — Atom
thermostat.py
55 # create a LoRa socket
56 lora_socket = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
57
58 # set the LoRaWAN data rate (not needed if ADR enabled)
59 #lora_socket.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
60
61 # make the socket non-blocking
62 lora_socket.setblocking(False)
63
64 # use port 12
65 lora_socket.bind(12)
66
67 # -----
68
69 def getTemperature():
70     return 1100.0 * temperature_pin() / 4096 / 10
71
72 def getRelayStatus():
73     return relay_pin()
74
75 def setRelayStatus(status):
76     relay_pin(status)
77
78 def sendMessage(temperature, relay_status):
79     lora_socket.send(bytes([int(temperature + 100), int(100 * (temperature -
80
```

thermostat/thermostat.py 0 0 0 0 1:1 LF UTF-8 Python master 5 files

# ATOM - PYMAKR REPL



The screenshot shows the Atom IDE interface. The main editor displays a Python script named `thermostat.py` with the following code:

```
55 # create a LoRa socket
56 lora_socket = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
57
58 # set the LoRaWAN data rate (not needed if ADR enabled)
59 #lora_socket.setsockopt(socket.SOL_LORA, socket.SO_DR, 5)
60
61 # make the socket non-blocking
62 lora_socket.setblocking(False)
63
64 # use port 12
65 lora_socket.bind(12)
66
67 # -----
68
69 def getTemperature():
70
```

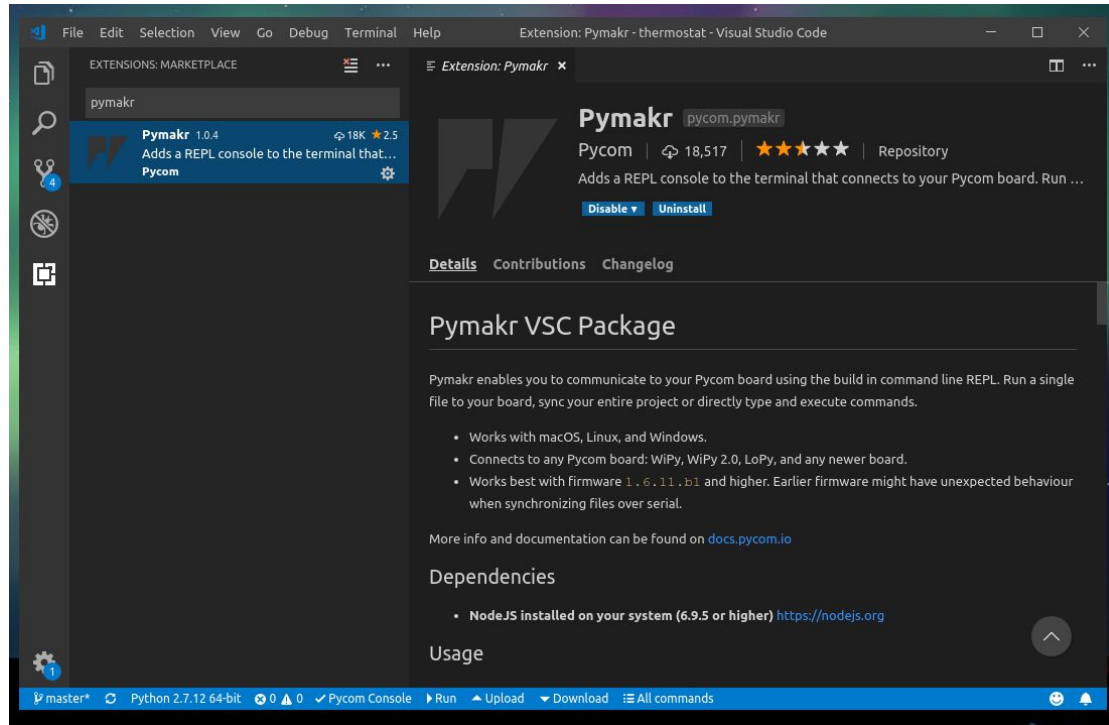
The left sidebar shows a project structure with folders `lopy` and `utils`. The `lopy` folder contains several Python files, including `thermostat.py`. The `utils` folder contains `mqtt_client.py`, `mqtt_client.sh`, `pymakr.conf`, and `README.md`.

At the bottom, a terminal window is connected to a device named `lopy`. The terminal shows the following output:

```
>>> Connecting on /dev/ttyUSB0...
>>> print("hello world")
hello world
>>> █
```

The status bar at the bottom of the IDE shows the current file is `thermostat/thermostat.py`, the encoding is `UTF-8`, the language is `Python`, and the workspace is `master`.

# VSCODE - INSTAL·LAR PYMAKR



The screenshot shows the Visual Studio Code interface with the Pymakr extension page open. The left sidebar shows the 'EXTENSIONS: MARKETPLACE' view with a search for 'pymakr'. The main panel displays the extension details for 'Pymakr' by 'pycom.pymakr'. The extension is version 1.0.4, has 18K downloads, and a 2.5 star rating. The description states: 'Adds a REPL console to the terminal that connects to your Pycom board. Run ...'. There are 'Disable' and 'Uninstall' buttons. Below the description, there are tabs for 'Details', 'Contributions', and 'Changelog'. The 'Details' tab is active, showing the 'Pymakr VSC Package' section. The text describes that Pymakr enables communication with a Pycom board using a built-in command line REPL. It lists several features: works with macOS, Linux, and Windows; connects to any Pycom board (WiPy, WiPy 2.0, LoPy, and any newer board); and works best with firmware 1.6.11.b1 and higher. A link to 'docs.pycom.io' is provided for more information. The 'Dependencies' section lists 'NodeJS installed on your system (6.9.5 or higher)' with a link to 'https://nodejs.org'. The 'Usage' section is partially visible. The bottom status bar shows 'master', 'Python 2.7.12 64-bit', '0', 'Pycom Console', 'Run', 'Upload', 'Download', and 'All commands'.

File Edit Selection View Go Debug Terminal Help Extension: Pymakr - thermostat - Visual Studio Code

EXTENSIONS: MARKETPLACE

pymakr

**Pymakr** 1.0.4 18K 2.5  
Adds a REPL console to the terminal that...  
Pycom

**Pymakr** pycom.pymakr  
Pycom | 18,517 | 2.5 | Repository  
Adds a REPL console to the terminal that connects to your Pycom board. Run ...  
Disable Uninstall

Details Contributions Changelog

### Pymakr VSC Package

Pymakr enables you to communicate to your Pycom board using the build in command line REPL. Run a single file to your board, sync your entire project or directly type and execute commands.

- Works with macOS, Linux, and Windows.
- Connects to any Pycom board: WiPy, WiPy 2.0, LoPy, and any newer board.
- Works best with firmware 1.6.11.b1 and higher. Earlier firmware might have unexpected behaviour when synchronizing files over serial.

More info and documentation can be found on [docs.pycom.io](https://docs.pycom.io)

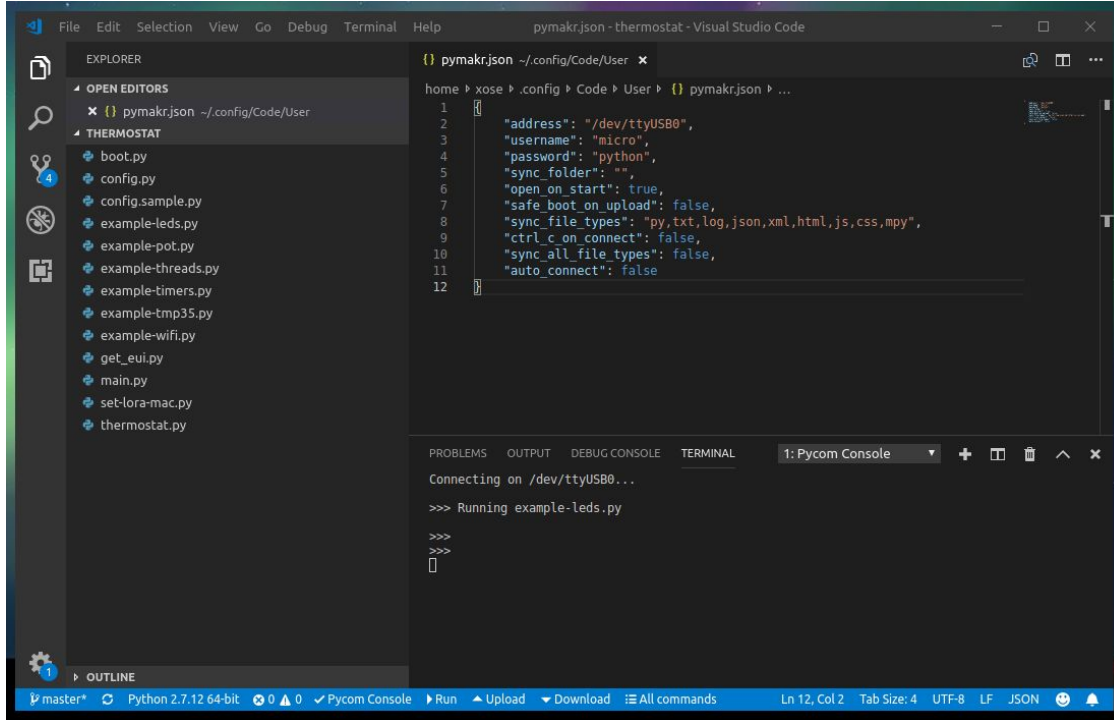
### Dependencies

- NodeJS installed on your system (6.9.5 or higher) <https://nodejs.org>

### Usage

master Python 2.7.12 64-bit 0 Pycom Console Run Upload Download All commands

# VSCODE - PYMAKR CONFIG



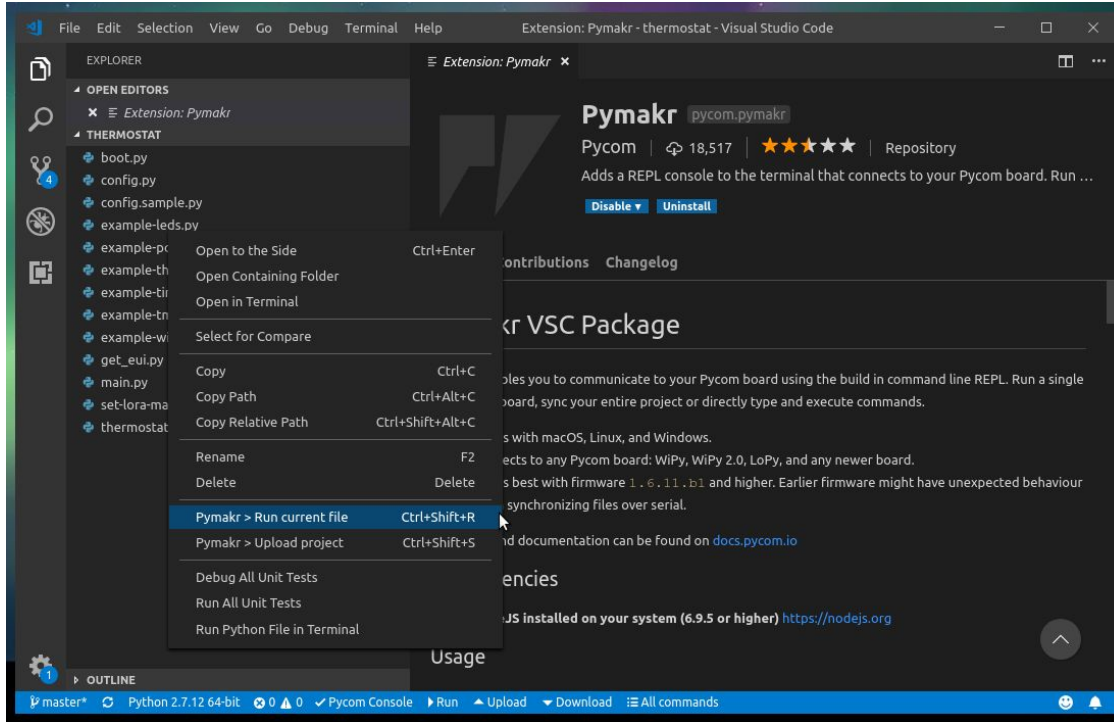
The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Shows a project structure for 'THERMOSTAT' with files like `boot.py`, `config.py`, `config.sample.py`, `example-leds.py`, `example-pot.py`, `example-threads.py`, `example-timers.py`, `example-tmp35.py`, `example-wifi.py`, `get_eui.py`, `main.py`, `set-lora-mac.py`, and `thermostat.py`.
- EDITOR:** Displays the `pymakr.json` configuration file with the following content:

```
1 {
2   "address": "/dev/ttyUSB0",
3   "username": "micro",
4   "password": "python",
5   "sync_folder": "",
6   "open_on_start": true,
7   "safe_boot_on_upload": false,
8   "sync_file_types": "py,txt,log,json,xml,html,js,css,mpy",
9   "ctrl_c_on_connect": false,
10  "sync_all_file_types": false,
11  "auto_connect": false
12 }
```
- TERMINAL:** Shows the output of the 'Pycom Console' with the following text:

```
Connecting on /dev/ttyUSB0...
>>> Running example-leds.py
>>>
>>>
[]
```
- STATUS BAR:** Shows the current environment as 'Python 2.7.12 64-bit' and the active tool as 'Pycom Console'. It also displays 'Ln 12, Col 2', 'Tab Size: 4', and 'UTF-8 LF JSON'.

# VSCODE - PYMAKR UPLOAD



The screenshot shows the Visual Studio Code interface with the Pymakr extension installed. The Explorer sidebar on the left shows a project named 'THERMOSTAT' with several files. A context menu is open over the 'thermostat' file, listing actions like 'Open to the Side', 'Copy', 'Run current file', and 'Upload project'. The main editor area displays the Pymakr extension page, which includes the extension name 'Pymakr', the publisher 'pycom.pymakr', a star rating, and a description: 'Adds a REPL console to the terminal that connects to your Pycom board. Run ...'. The status bar at the bottom shows 'Python 2.7.12 64-bit', 'Pycom Console', and buttons for 'Run', 'Upload', and 'Download'.


# EXEMPLES

<https://github.com/ttnecat/termostat/tree/master/lopy>





ttnecat / **termostat** Unwatch 3 Unstar 4 Fork 0



[Code](#) [Issues 0](#) [Pull requests 0](#) [Insights](#)

Branch: **master** [termostat / lopy /](#) Create new file Upload files Find file History

 **xoseperez** Small changes Latest commit deaf06d 2 hours ago

..

 <a href="#">thermostat</a>	Small changes	2 hours ago
 <a href="#">utils</a>	More code for the LoPys	16 hours ago
 <a href="#">README.md</a>	LoPy resources	6 months ago
 <a href="#">pymakr.conf</a>	Move pymakr config file	16 hours ago

 **README.md** 

## Termostat amb PyCom LoPy

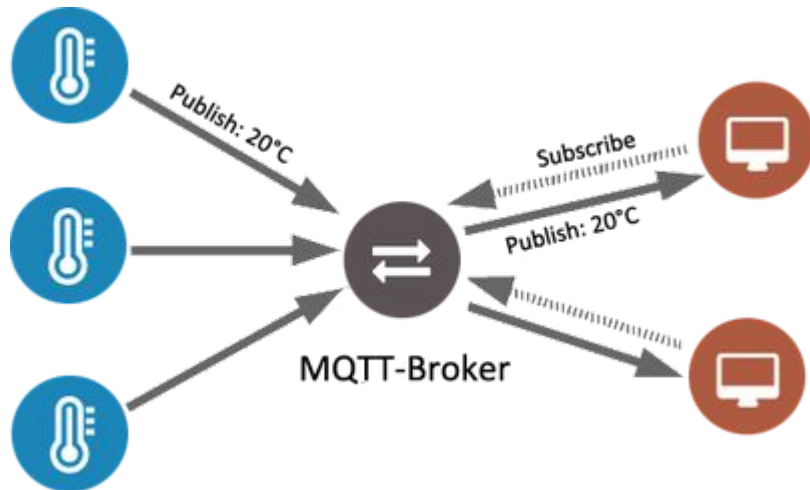
Exemple de comunicació bidireccional sobre TTN amb un PyCom LoPy.

license GPL-3.0 Follow @ttnecat 734

### Hardware



# MQTT



```
import sys
import json
from pygments import highlight, lexers, formatters
import paho.mqtt.client as mqtt

# configuration
app_id = "test"
access_key = "ttn-account-v2.ALT...."

# callback functions
def on_connect(client, userdata, flags, rc):
    print("Subscribing...")
    # subscribe for all devices of user
    client.subscribe('+/devices/+/up')

def on_subscribe(client, userdata, mid, granted_qos):
    print("Subscribed")

def on_message(client, userdata, msg):
    formatted_json = json.dumps(json.loads(msg.payload), indent=4)
    colorful_json = highlight(unicode(formatted_json, 'UTF-8'), lexers.JsonLexer(), formatters.TerminalFormatter())

    print(colorful_json)

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.on_subscribe = on_subscribe
client.username_pw_set(app_id, access_key)

print("Connecting...")
client.connect("eu.thethings.network", 1883, 60)

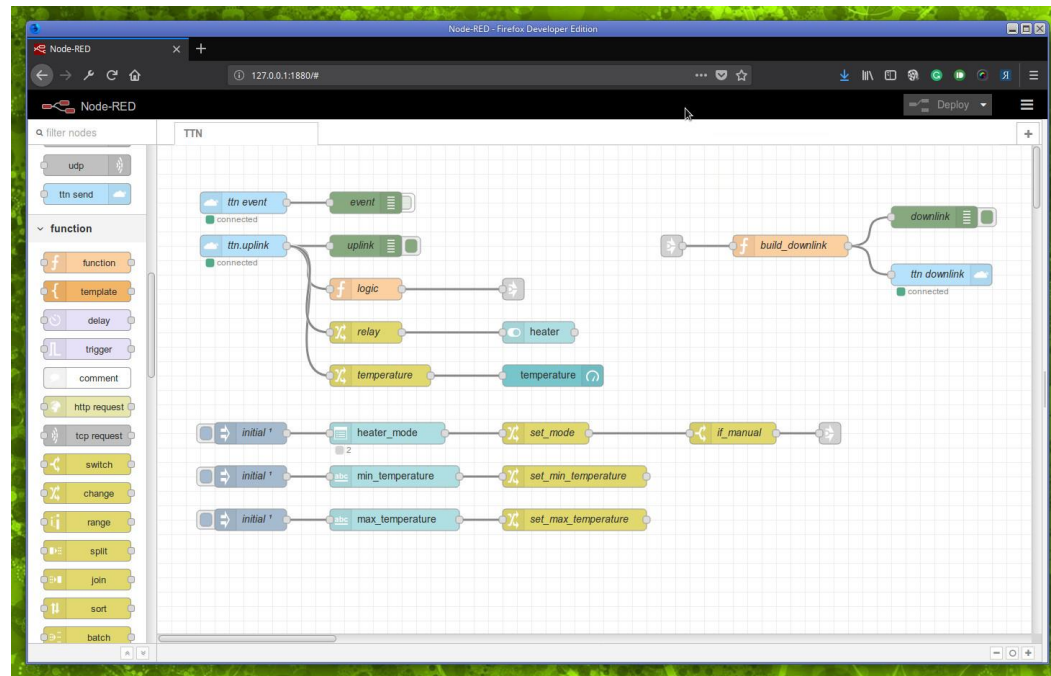
# and listen to server
client.loop_forever()
```

# APLICATIU (NODE-RED)

L'aplicatiu de destí pot ser el que volguem. Ens podem connectar a TTN des de qualsevol aplicatiu que “parli” MQTT.

Node-RED és un entorn de treball on podem definir la lògica de l'aplicació a partir de nodes que reben, manipulen i re-envien missatges.

La gent de TTN ha creat un *plugin* específic (*node-red-contrib-ttn*) que ens permet rebre notificacions i rebre missatges (*uplink*) o enviar-ne (*downlink*).

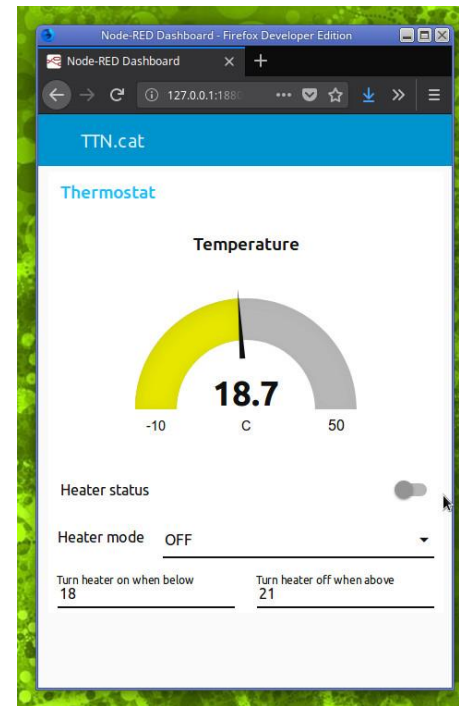


# APLICATIU (NODE-RED)

A més Node-RED disposa d'un senzill entorn gràfic on visualitzar la informació i interactuar amb ella (node-red-dashboard).

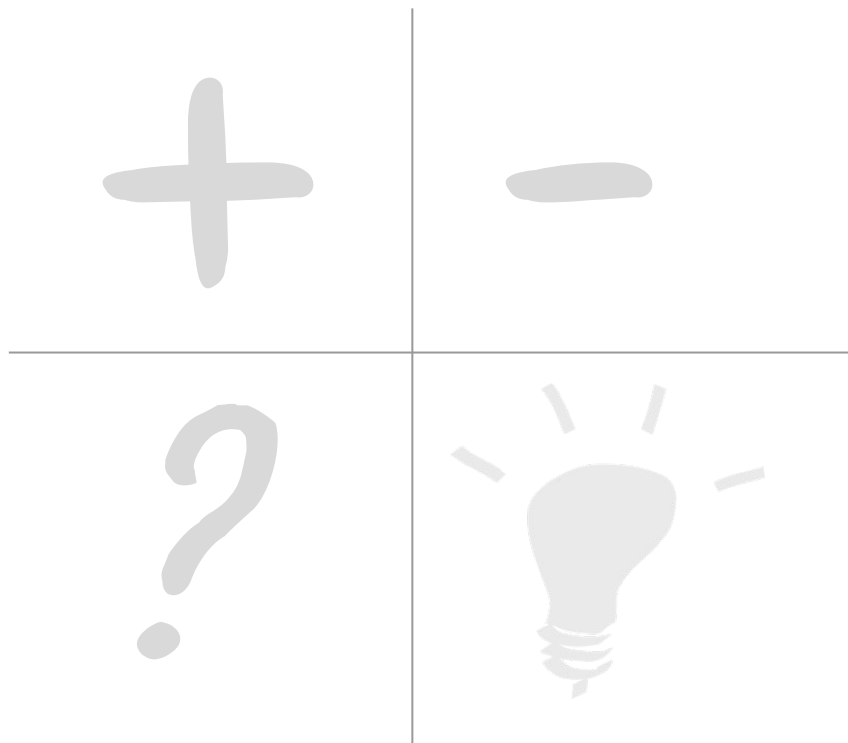
Aquest entorn ens permet fer, per exemple, el control del nostre termòstat i definir si volem que el calefactor estigui obert, tancat o que depengui de la temperatura.

Un aspecte important a tenir en compte és que Node-RED és una aplicació web, de manera que el podem tenir executant-se en un servidor web (que suporti node.js) i accedir-hi des de qualsevol lloc: des de l'ordinador de casa, el mòbil en el tren cap a la feina o des de Fuerteventura.

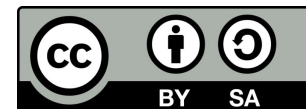


QUÈ T'HA SEMBLAT

# GRÀCIES



thethingsnetwork.cat - ttn.cat  
@ttnocat a twitter



Gràfics de la Wikipedia i de The Things Network