Navegació

Pàgina principal
Joan Quintana
Joanillo Blog

Institut Jaume Balmes

DAW-M06-WEC
ASIX-M09-IAW
ASIX-M10-UF2
Extraescolar Robòtica 16-17
Restful API (temperatures)

Màquines recreatives

Màquines Recreatives
RetroPlaneta

CNC

CNC
Projectes CNC

Informàtica musical

Informàtica musical
Ubuntu Studio
Projectes musicals
50 Ways to Play...

joanillo.org Planet

RetroPlaneta Arcade
Espai Balena Coworking
LangTrainer
Arthropoda
GPS Routes
Dòlmens Cat

Eines

Què hi enllaça
Seguiment d'enllaços
Pàgines especials
Versió per a impressora
Enllaç permanent

Node TTN amb TTGO

Contingut [amaga]

Referències

- https://primalcortex.wordpress.com/2017/11/24/the-esp32-oled-lora-ttgo-lora32-board-and-connecting-it-to-ttn/
- https://randomnerdtutorials.com/esp32-lora-rfm95-transceiver-arduino-ide/
- https://tutorial.cytron.io/2017/09/15/lesson-1-build-simple-arduino-lora-node-10-minutes/

Introducció

Ara que ja tinc feta la gateway, ara he de fer un node per tal de connectar-se a aquesta antena (o bé a qualsevol altra!).

tinc els següents codis que funcionen:

- TTGO_TTN_Node_v1.ino
- ESP-ttn-apb-node_sensor_v1.ino i ESP-ttn-abp-node_sensor_v2.ino

El segon codi l'he tret d'aquí:

- https://github.com/McOrts/LoRa_gateway/blob/master/ESP-ttn-abp-node_sensor/ESP-ttn-abp-node_sensor.ino

i he eliminat tota la part del sensor, i la part del OLED no funciona. Però allò important és que la connexió del meu device amb la meva gateway ja funciona. El meu dispositiu l'he registrat a *thethingsnetwork*, i ja puc veure com es van enviant les dades (es pot veure tant a la part del device, com a la part del gateway). La informació que s'envia està en el payload, en codi hexadecimal, i per veure que efectivament s'està enviant el missatge necessito desxifrar els caràcters ASCII. En el tràfic de la Gateway la informació està xifrada. En canvi, en les dades del device (aplicació), aquí sí que es poden veure les dades en el *payload*.

En la v2 ja funciona la pantalla OLED, i he simulat un sensor ficant números aleatoris.

NOTA. Tot i que els dos exemples funcionen, el segon exemple (*ESP-ttn-apb-node_sensor_v1*) consumeix més corrent que el primer, de manera que la bateria autònoma que tinc de 5600mAh (i 1.5A) no aguanta el node. En canvi, amb el primer exemple (*TTGO_TTN_Node_v1.ino*) la bateria sí que aguanta el node. Per tant, s'ha d'estudiar quina és la causa d'aquesta disparitat de consum entre un exemple i un altre (i no és atribuible a la pantalla OLED). De fet, el codi *ESP-ttn-apb-node_sensor* és bastant més llarg que el primer (*TTGO_TTN_Node_v1*) tot i que al cap i a la fi fan el mateix, que és enviar dades del node al gateway.

Visualitzar les dades

Consola TheThingsNetwork. Payload format

L'objectiu és que en les dades de l'Aplicació, juntament amb els bytes que rep el gateway, es puguin veure les dades que s'envien en un format reconeixible. En el *payload formats* es tracta de substituir la funció **Decoder** de forma convenient.

Anem a veure el primer exemple pràctic. Estic generant números aleatoris entre 0 i 300, simulant un sensor. En el codi Arduino tinc:

```
      randNumber = random(300);
      sprintf(randNumberStr,"%d",randNumber);
      Serial.println(F(randNumberStr));
...
      //aquí enviem les dades a la Gateway
      LMIC_setTxData2(1, (uint8_t*) randNumberStr, sizeof((uint8_t*)randNumberStr)-1, 0);
```

Per exemple, per al valor *240* s'envien els bytes hexadecimals *32 34 30* (el valor 32hex es correspon al símbola ASCII *2*, que és el valor 50 decimal. A mi m'interessa el símbol ASCII).

En la consola he de recuperar el valor numèric de 240.

La funció **Decoder()** original, que s'ha de sobreescriure, és:

```
function Decoder(bytes, port) {
  // Decode an uplink message from a buffer
  // (array) of bytes to an object of fields.
  var decoded = {};

  // if (port === 1) decoded.led = bytes[0];

  return decoded;
}
<pre>
Defineixo la següent funció '''Decoder''':
<pre>
function Decoder(bytes) {
var temp = parseInt(String.fromCharCode(bytes[0],bytes[1],bytes[2]),10);
return {
Temp : (temp)
```

```
};
}
```

testejo la funció (dóna error perquè no li passem cap valor, però està bé). A partir dels bytes *32 34 30* recupero el String *240*, i converteixo a integer.

Ara, en el data application, al costat del Payload ja tinc una informació digerible. El més interessant és que la funció Decoder() pot retornar el format JSON.

En funció de com el meu sensor em doni les dades, hauré de programar la funció **Decoder()**. Per exemple, un exemple trobat en el fòrum:

```
The data im sending to ttn is this 000A34 this is Temperature: 26.12
Can anyone help me decode it in ttn backend ?

use the following Payload Function to decode it in console:

function Decoder(bytes) {
var temp = (((bytes[1] << 8) | bytes[2])/100 ).toFixed(2);
return {
Temp : (temp),
};
}
```

Integration i Storage

Ara que ja tinc les dades a TTN, en un format comprensible, he de pensar què fer amb aquestes dades. Com es guarden, com es visualitzen. Hi ha moltes possibilitats: diferents possibilitats d'Integration, APIs,... A continuació es fan diferents proves amb diferents mètodes d'integració.

- https://www.thethingsnetwork.org/docs/applications/storage/

Mètode 1. The Things Network Application SDK for Python

- https://github.com/TheThingsNetwork/python-app-sdk

API Reference:

- https://github.com/TheThingsNetwork/python-app-sdk/blob/master/DOCUMENTATION.md

```
$ sudo pip install 'ttn<3'
```

primer exemple: script **ttn_example.py**:

```
import time
import ttn

app_id = "prova_gateway_ttgo"
access_key = "ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0"

def uplink_callback(msg, client):
  print("Received uplink from ", msg.dev_id)
  print(msg)

handler = ttn.HandlerClient(app_id, access_key)

# using mqtt client
mqtt_client = handler.data()
mqtt_client.set_uplink_callback(uplink_callback)
mqtt_client.connect()
time.sleep(60)
mqtt_client.close()

# using application manager client
app_client =  handler.application()
my_app = app_client.get()
print(my_app)
my_devices = app_client.devices()
print(my_devices)
```

```
$ python ttn_example.py
('Received uplink from ', u'provasensor1')
MSG(dev_id=u'provasensor1', counter=114, app_id=u'prova_gateway_ttgo', payload_fields=MSG(Temp=153), payload_raw=u'MTUz', hardware
...
```

Tanmateix, el app_client.get() i app_client.devices() em dóna un error tipus *Permission denied*.

Mètode 2. Client de MQTT

TTN és un broker de MQTT, que vol dir que ens podem subscriure a un Node per recuperar les dades mitjançant una subscripció MQTT. En la meva màquina Ubunt instal·lo un client de **mosquitto**.

```
$ sudo apt install mosquitto-clients
```

- https://www.thethingsnetwork.org/docs/applications/mqtt/quick-start.html

Receive Messages (up)

```
$ mosquitto_sub -h <Region>.thethings.network -t '+/devices/+/up' -u '<AppID>' -P '<AppKey>' -v
```

A <Region> ficaré la última part del meu handler: **eu**

En el meu cas:

```
les dades del meu node:
app_id = "prova_gateway_ttgo"
access_key = "ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0"

$ mosquitto_sub -h eu.thethings.network -t '+/devices/+/up' -u 'prova_gateway_ttgo' -P 'ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTI
```

```
prova_gateway_ttgo/devices/provasensor1/up {"app_id":"prova_gateway_ttgo","dev_id":"provasensor1","hardware_serial":"00824559895E61
prova_gateway_ttgo/devices/provasensor1/up {"app_id":"prova_gateway_ttgo","dev_id":"provasensor1","hardware_serial":"00824559895E61
```

Mètode 3. Client de MQTT amb Python

- https://sakshambhatla.wordpress.com/2014/08/11/simple-mqtt-broker-and-client-in-python/

```
sudo apt-get install python-pip
```

Next, install the MQTT broker Mosquitto (or Paho now)

```
pip install paho-mqtt
```

script **mqtt_client.py**:

```
# -*- coding: utf-8 -*-
#aquesta instrucció funciona:
#$ mosquitto_sub -h eu.thethings.network -t '+/devices/+/up' -u 'prova_gateway_ttgo' -P 'ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmT

# Simple Python client to show node activity from ttn MQTT brooker with credentials
# Author: R.Schimmel
# www.schimmel-bisolutions.nl
# first install paho.mqtt.client: pip install paho-mqtt
#
import paho.mqtt.client as mqtt

#Call back functions

# gives connection message
def on_connect(mqttc, mosq, obj,rc):
    print("Connected with result code:"+str(rc))
    # subscribe for all devices of user
    mqttc.subscribe('+/devices/+/up')

# gives message from device
def on_message(client,userdata,msg):
        print(str(msg.payload));
        #print"Topic",msg.topic + "\nMessage:" + str(msg.payload)

def on_log(client,userdata,level,buf):
        print("message:" + str(buf))
        print("userdata:" + str(userdata))

mqttc= mqtt.Client()
mqttc.on_connect=on_connect
mqttc.on_message=on_message

#mqttc.username_pw_set("App-eu registered op ttn dashboard","key registered op ttn dashboard")
mqttc.username_pw_set("prova_gateway_ttgo","ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0")

mqttc.connect("eu.thethings.network",1883,60)

# and listen to server
run = True
while run:
        mqttc.loop()
```

script **mqtt_client2.py** (ara ja puc extreure la informació de les dades):

```
#!/<path-to>/python
# Get date from MQTT server

import paho.mqtt.client as mqtt
import json
import base64

APPEUI = "70B3D57ED00120A7"
APPID  = "prova_gateway_ttgo"
PSW    = 'ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0'

#Call back functions

# gives connection message
def on_connect(mqttc, mosq, obj,rc):
    print("Connected with result code:"+str(rc))
    # subscribe for all devices of user
    #mqttc.subscribe('+/devices/#')
    mqttc.subscribe('+/devices/+/up')

# gives message from device
def on_message(mqttc,obj,msg):
    x = json.loads(msg.payload)
    device = x["dev_id"]
    payload_raw = x["payload_raw"]
    payload_plain = base64.b64decode(payload_raw)
    datetime = x["metadata"]["time"]
    #rssi = x["metadata"]["gateways"]["rssi"]  # <== this raises an error (why?)
    rssi = -1
    print(device + ": " + payload_raw + " ==> " + payload_plain + ", RSSI ["+ str(rssi) + "] @" + datetime )

def on_publish(mosq, obj, mid):
    print("mid: " + str(mid))

def on_subscribe(mosq, obj, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

def on_log(mqttc,obj,level,buf):
    print("message:" + str(buf))
    print("userdata:" + str(obj))

mqttc= mqtt.Client()
# Assign event callbacks
mqttc.on_connect=on_connect
```

```
mqttc.on_message=on_message

mqttc.username_pw_set(APPID, PSW)
mqttc.connect("eu.thethings.network",1883,60)

# and listen to server
run = True
while run:
    mqttc.loop()
```

```
$ python mqtt_client2.py
Connected with result code:0
provasensor1: MgAA ==> 2, RSSI [-1] @2018-09-06T17:45:50.583737426Z
provasensor1: MTY5 ==> 169, RSSI [-1] @2018-09-06T17:46:52.66040219Z
```

(les dades que es volien obtindre són *2* i *169*).

Omplir una base de dades mysql

A partir de les dades rebudes amb el client de MQTT es pot omplir una base de dades mysql.

```
create database sensor_rand;
create table sensor_rand (id integer primary key,valor integer);

$ sudo apt-get install python-mysqldb
```

script **mqtt_mysql.py**:

```
#!/<path-to>/python
# Get date from MQTT server (TTN, TheTingsNetwork), i insereix la informacio a una base de dades local


import paho.mqtt.client as mqtt
import json
import base64
import MySQLdb

APPEUI = "70B3D57ED00120A7"
APPID  = "prova_gateway_ttgo"
PSW    = 'ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0'

#Call back functions

# gives connection message
def on_connect(mqttc, mosq, obj,rc):
    print("Connected with result code: "+str(rc))
    # subscribe for all devices of user
    #mqttc.subscribe('+/devices/#')
    mqttc.subscribe('+/devices/+/up')

# gives message from device
def on_message(mqttc,obj,msg):
    global db, cur, comptador

    x = json.loads(msg.payload)
    device = x["dev_id"]
    payload_raw = x["payload_raw"]
    payload_plain = base64.b64decode(payload_raw)
    datetime = x["metadata"]["time"]
    #rssi = x["metadata"]["gateways"]["rssi"]  # <== this raises an error (why?)
    rssi = -1
    print(device + ": " + payload_raw + " ==> " + payload_plain + ", RSSI ["+ str(rssi) + "] @" + datetime )
    comptador = comptador + 1
    cur.execute("INSERT INTO sensor_rand(id,valor) VALUES (%s, %s)", (comptador,payload_plain))
    db.commit()
    print("valor " + payload_plain + " inserit a la bd")

def on_publish(mosq, obj, mid):
    print("mid: " + str(mid))

def on_subscribe(mosq, obj, mid, granted_qos):
    print("Subscribed: " + str(mid) + " " + str(granted_qos))

def on_log(mqttc,obj,level,buf):
    print("message:" + str(buf))
    print("userdata:" + str(obj))

db = MySQLdb.connect(host="localhost",    # your host, usually localhost
                     user="root",         # your username
                     passwd="She4aiVa",  # your password
                     db="sensor_rand")        # name of the data base

# you must create a Cursor object. It will let
#  you execute all the queries you need
cur = db.cursor()

cur.execute("SELECT max(id) FROM sensor_rand")
result = cur.fetchone()
#print result[0]
if (result[0]):
    comptador = result[0]
else:
    comptador = 0

mqttc= mqtt.Client()
# Assign event callbacks
mqttc.on_connect=on_connect
mqttc.on_message=on_message

mqttc.username_pw_set(APPID, PSW)
mqttc.connect("eu.thethings.network",1883,60)

# and listen to server
run = True
while run:
    mqttc.loop()
```

```
cur.close()
db.close()
```

Visualitzar les dades del mysql amb la llibreria HighCharts de Javascript

Estem agafant les dades del *sensor_rand* del núvol (TTN), i les estem important a una base de dades loca. podem fer una pàgina web per visualitzar aquestes dades. Un dels exemples més senzills és el script

sensor_rand_data_mysql.php:

```
<?php
//codi original i adaptat de:
//https://geekytheory.com/php-mysql-highchart-mostrar-varias-graficas-dinamicamente

 function conectarBD(){
            $server = "localhost";
            $usuario = "root";
            $pass = "She4aiVa";
            $BD = "sensor_rand";
            //variable que guarda la conexión de la base de datos
            $conexion = mysqli_connect($server, $usuario, $pass, $BD);
            //Comprobamos si la conexión ha tenido exito
            if(!$conexion){
               echo 'Ha sucedido un error inexperado en la conexion de la base de datos<br>';
            }
            //devolvemos el objeto de conexión para usarlo en las consultas
            return $conexion;
    }
    /*Desconectar la conexion a la base de datos*/
    function desconectarBD($conexion){
            //Cierra la conexión y guarda el estado de la operación en una variable
            $close = mysqli_close($conexion);
            //Comprobamos si se ha cerrado la conexión correctamente
            if(!$close){
               echo 'Ha sucedido un error inexperado en la desconexion de la base de datos<br>';
            }
            //devuelve el estado del cierre de conexión
            return $close;
    }

    //Devuelve un array multidimensional con el resultado de la consulta
    function getArraySQL($sql){
        //Creamos la conexión
        $conexion = conectarBD();
        //generamos la consulta
        if(!$result = mysqli_query($conexion, $sql)) die();

        $rawdata = array();
        //guardamos en un array multidimensional todos los datos de la consulta
        $i=0;
        while($row = mysqli_fetch_array($result))
        {
            //guardamos en rawdata todos los vectores/filas que nos devuelve la consulta
            $rawdata[$i] = $row;
            $i++;
        }
        //Cerramos la base de datos
        desconectarBD($conexion);
        //devolvemos rawdata
        return $rawdata;
    }

    //Sentencia SQL
//$sql = "SELECT energy,water,temperature,time from tabla;";
$sql = "SELECT id,valor from sensor_rand;";
//Array Multidimensional
$rawdata = getArraySQL($sql);

//Adaptar el tiempo
for($i=0;$i<count($rawdata);$i++){
    //$time = $rawdata[$i]["time"];
    //$date = new DateTime($time);
    //$rawdata[$i]["time"]=$date->getTimestamp()*1000;
    $id = $rawdata[$i]["id"];
    //$date = new DateTime($time);
    $rawdata[$i]["id"]=$id;
}

?>

<HTML>
<BODY>

<meta charset="utf-8">

<!-- Latest compiled and minified JavaScript -->
<script src="https://code.jquery.com/jquery.js"></script>
    <!-- Importo el archivo Javascript de Highcharts directamente desde su servidor -->
<script src="http://code.highcharts.com/stock/highstock.js"></script>
<script src="http://code.highcharts.com/modules/exporting.js"></script>

<div id="container">
</div>


<script type='text/javascript'>
$(function () {
    $(document).ready(function() {
        Highcharts.setOptions({
            global: {
                useUTC: false
            }
        });

        var chart;
        $('#container').highcharts({
            chart: {
                type: 'spline',
                animation: Highcharts.svg, // don't animate in old IE
```

```
                marginRight: 10,
                events: {
                    load: function() {

                    }
                }
            },
            title: {
                text: 'Physical Variables'
            },
            xAxis: {
                type: 'linear',
                tickPixelInterval: 5
            },
            yAxis: {
                title: {
                    text: 'Value'
                },
                plotLines: [{
                    value: 0,
                    width: 1,
                    color: '#808080'
                }]
            },
            tooltip: {
                formatter: function() {
                        return '<b>'+ this.series.name +'</b><br/>'+
                        Highcharts.dateFormat('%Y-%m-%d %H:%M:%S', this.x) +'<br/>'+
                        Highcharts.numberFormat(this.y, 2);
                }
            },
            legend: {
                enabled: true
            },
            exporting: {
                enabled: true
            },
            series: [{
                name: 'Valor Temp',
                data: (function() {
                   var data = [];
                    <?php
                        for($i = 0 ;$i<count($rawdata);$i++){
                    ?>
                    data.push([<?php echo $rawdata[$i]["id"];?>,<?php echo $rawdata[$i]["valor"];?>]);
                    <?php } ?>
                return data;
                })()
            }]
        });
    });

});


</script>
</html>
```

Mètode 4. Recuperar les dades amb Node.js

- https://www.thethingsnetwork.org/forum/t/ttn-node-js-api-quick-start-start-guide/12802/8

script **nodejs_ttn.js**:

```
var ttn = require("ttn")

var appID = "prova_gateway_ttgo"
var accessKey = "ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0"

console.log("Program running")

ttn.data(appID, accessKey).then(function (client) {
        client.on("uplink", function (devID, payload) {
            console.log("Received uplink from ", devID)
            console.log(payload)
                        console.log(payload.dev_id)
                        console.log(payload.payload_fields)
                        console.log(payload.payload_fields.Temp)
        })
    })
    .catch(function (error) {
        console.error("Error", error)
        process.exit(1)
    })
```

De fet, la tècnica que utilitza *require("ttn")* és com si fos un client de MQTT. Per tant, no cal buscar un exemple d'un client MQTT de NodeJS, amb aquest tros de codi ja recupero la informació del meu node TTN.

```
$ node nodejs_ttn.js
Program running
Received uplink from  provasensor1
{ app_id: 'prova_gateway_ttgo',
  dev_id: 'provasensor1',
  hardware_serial: '0082455989E61BC0',
  port: 1,
  counter: 551,
  payload_raw: <Buffer 31 37 35>,
  payload_fields: { Temp: 175 },
  metadata:
   { time: '2018-09-06T23:15:54.464993292Z',
     frequency: 868.5,
     modulation: 'LORA',
     data_rate: 'SF7BW125',
     airtime: 51456000,
     coding_rate: '4/5',
     gateways: [ [Object] ],
     latitude: 41.4016,
```

```
      longitude: 2.1618063,
      location_source: 'registry' } }
provasensor1
{ Temp: 175 }
175
...
```

Ara des de NodeJS també podria ficar les dades en un mysql local.

Mètode 5. Data Storage v2.0.1(TBD)

Integration. Data Storage (v2.0.1)

Stores data and makes it available through an API (a REST API). Your data is stored for seven days.

- https://www.youtube.com/watch?v=kVf8GmCbOuE&index=3
- https://prova_gateway_ttgo.data.thethingsnetwork.org/swagger.yaml

```
$ curl -X GET --header 'Accept: application/json' --header 'Authorization: key ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLI
```

L'exemple amb CURL funciona, des de la consola.

- http://blog.dreamfactory.com/get-this-7-simple-rest-client-examples-for-retrieving-api-data/

For our first example we will look at two simple NodeJS scripts. Below is an example of a native NodeJS HTTP GET request. In your favorite text editor create a new file called 'rest.js' and enter the following code:

Ara es tracta de tenir diferents codis de clients REST API des dels quals puc accedir a aquesta informació. Començo amb **Node.js**:

script **rest.js**:

```
var http = require("http");

var options = {
        "method": "GET",
        "hostname": "https://prova_gateway_ttgo.data.thethingsnetwork.org",
        "port": null,
        "path": "/api/v2/query",
        "headers": {
                "Accept": "application/json",
                "Authorization-key": "key ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0"
        }
};


var req = http.request(options, function(res) {
        var chunks = [];

        res.on("data", function(chunk) {
                chunks.push(chunk);
        });

        res.on("end", function() {
                var body = Buffer.concat(chunks);
                console.log(body.toString());
        });


});
req.end();
console.log ("adeu");
```

Però no acaba de funcionar (TBD)

```
$ node rest.js
```

Mètode 6. HTTP Integration (v2.6.0) (TBD)

HTTP Integration (v2.6.0) The Things Industries B.V

The first type is the "HTTP integration". Using this one, all received messages for your application will be forwarded to the url provided in the integration configuration. A complete documentation is available at https://www.thethingsnetwork.org/docs/applications/http/

- https://www.youtube.com/watch?v=Uebcq7xmI1M&index=2&list=PLM8eOeiKY7JVwrBYRHxsf9p0VM_dVapXl

Alternative to requestb.in https://beeceptor.com/

- https://beeceptor.com/console/testejant

The following endpoint is all set up. Use it in your code as base URL and send a request. You can inspect these requests here and build rules to mock responses.
https://testejant.free.beeceptor.com

For example, run the following command in shell/terminal to get started.

```
curl -v -X GET 'https://testejant.free.beeceptor.com/my/api/path' -H 'some-header: some-value'
```

Mètode 7. AWS IoT Integration (TBD)

Video:

- https://www.youtube.com/watch?v=XuMv258g1kY

Quick Start:

- https://www.thethingsnetwork.org/docs/applications/aws/quick-start.html

Primer de tot ens hem de registrar a AWS (Amazon Web Services)

- AWS account name: joanillo
- login: joanqc@gmail.com / Jqc********

Vaig seguint el tutorial i el video:

```
Stack name: ttn-integration

Stack ID:arn:aws:cloudformation:eu-west-1:585202107532:stack/ttn-integration/8dd38050-b1dc-11e8-bd90-50a68645b236

Status:CREATE_IN_PROGRESS
```

Services > IoT Core AWS IoT is a managed cloud platform that lets connected devices - cars, light bulbs, sensor grids, and more - easily and securely interact with cloud applications and other devices.

Manage > Things > Register a Thing > Create a Simple Thing

```
app_id: prova_gateway_ttgo
dev_eui: 0082455989E61BC0
dev_ id: provasensor1

app_eui: 70B3D57ED00120A7
app_key:  ttn-account-v2.V3Sus2pUKvMq8qEV2ZE-gtEVmTDRX8wxBeLLKtCwSM0
```

Create a Thing without certificate,...

i ja tinc creada la meva thing

(TBD)

---

creat per Joan Quintana Compte setembre 2018