

Registrazione del gateway sulla rete TTN

The Things Network consente di registrare un gateway LoRa in pochissimo tempo con pochi e semplici passaggi. Consente inoltre di creare applicazioni per la decodifica dei payload (carichi utili) e l'integrazione con servizi di terze parti per elaborare ulteriormente i dati.

Creazione di un account

Per prima cosa bisogna iscriversi recandosi all'indirizzo: <https://www.thethingsnetwork.org> e fare clic sul pulsante *Sign Up* nell'angolo in alto a destra della pagina (Figura 6.1).

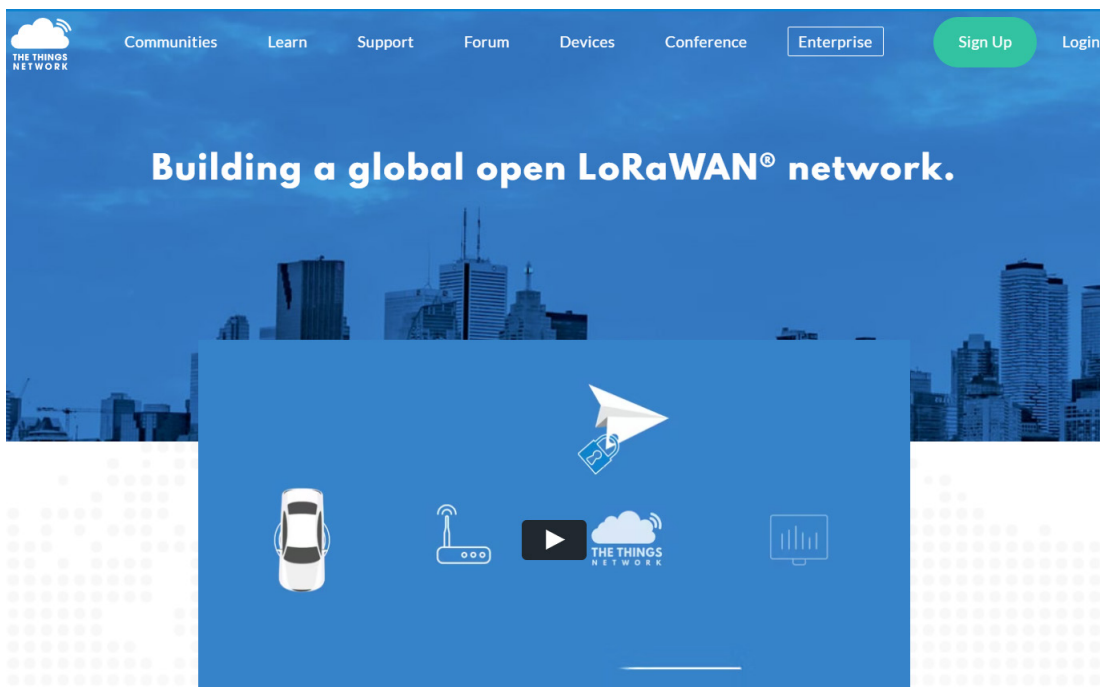
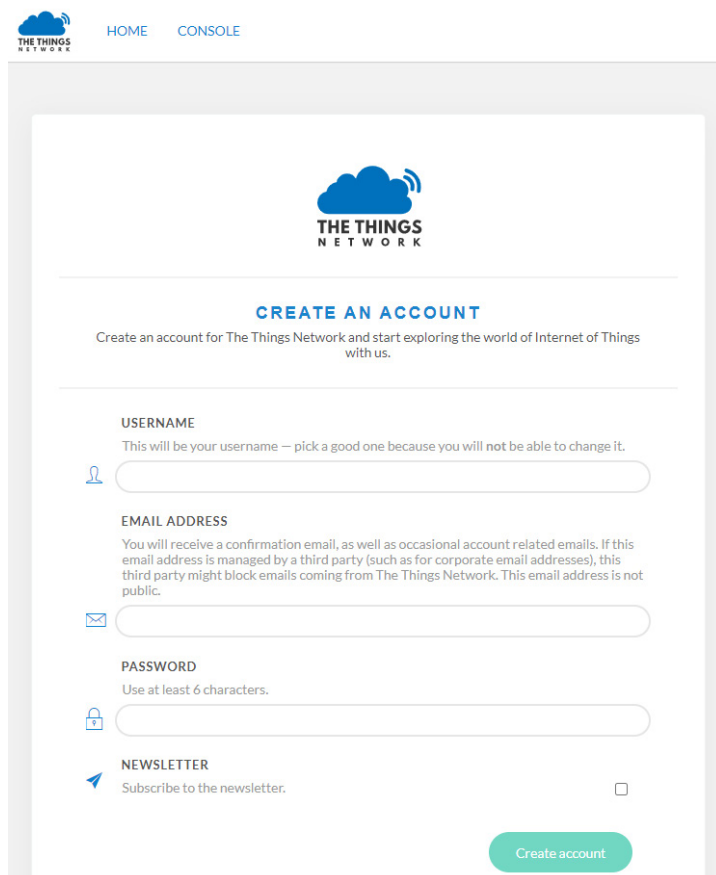


Figura 6.1 Registrarsi con TTN.

Nella pagina *Create an account*, digitare un nome utente, un indirizzo email valido e una password con almeno sei caratteri. Quindi fare clic sul pulsante “Create account” (Figura 6.2).



THE THINGS NETWORK

HOME CONSOLE

THE THINGS NETWORK

CREATE AN ACCOUNT

Create an account for The Things Network and start exploring the world of Internet of Things with us.

USERNAME
This will be your username — pick a good one because you will not be able to change it.

EMAIL ADDRESS
You will receive a confirmation email, as well as occasional account related emails. If this email address is managed by a third party (such as for corporate email addresses), this third party might block emails coming from The Things Network. This email address is not public.

PASSWORD
Use at least 6 characters.

NEWSLETTER
Subscribe to the newsletter. ☐

Create account

Figura 6.2 Creazione di un nuovo account.

Dopo qualche minuto, nella posta in arrivo, arriverà un email di convalida dell’account The Things Network. Quindi fare clic sul pulsante *Activate account*.

Dopo aver attivato l’account, nella pagina di benvenuto, fare clic sul pulsante *Login* nella parte superiore della pagina e inserire le proprie credenziali per effettuare l’accesso (Figura 6.3).

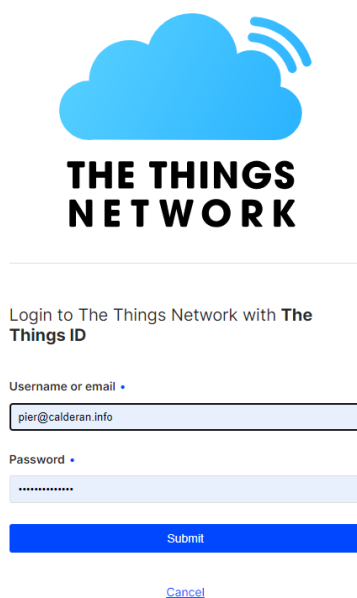


Figura 6.3 Pagina di login per l'accesso a The Things Network

Una volta effettuato l'accesso, si potranno modificare i dati del proprio profilo, uscire da TTN o accedere alla propria *Console* (Figura 6.4).

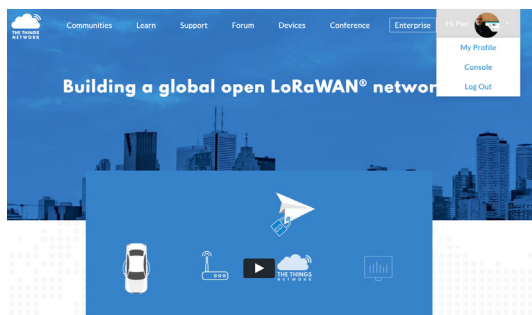


Figura 6.4 Pagina per l'accesso alla Console personale.

La console consente di registrare gateway e dispositivi, creare applicazioni, creare integrazioni e gestire collaboratori e varie impostazioni. Se si memorizzano i dati di accesso nel browser, la pagina della Console, dopo aver selezionato Europe 1 come cloud, sarà accessibile direttamente da questo link:

<https://eu1.cloud.thethings.network/console/>.

La pagina della console si presenta come illustrato in Figura 6.5, in cui sono visibili due grosse icone con le scritte **Go to applications** e **Go to gateways**.

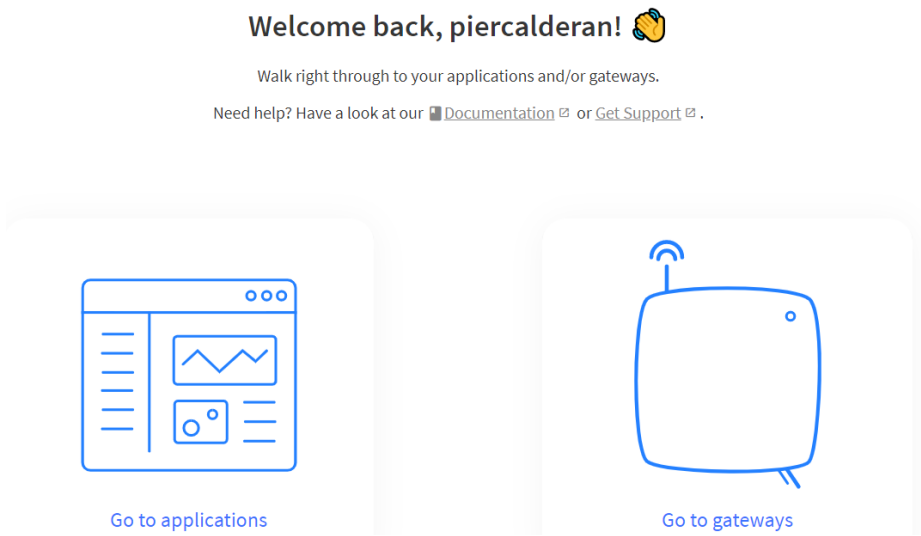


Figura 6.5 Accesso alla console di The Things Network.

A questo punto, se è la prima volta che vi si accede, è necessario registrare almeno un gateway.

Se sono stati registrati già altri gateway in precedenza verranno elencati, come nel caso illustrato in Figura 6.6, in cui si può vedere un gateway già registrato come “iC880 packet forwarder” fatto con iC880A (multicanale).

Attenzione!

Nel nuovo stack V3 di TTN i gateway a singolo canale non sono più supportati.

Per registrare un nuovo gateway o per aggiungerne uno a quelli già esistenti, basta fare clic su *Add gateway*, come indicato nella figura.

Gateways (1)					Search by ID	+ Add gateway
ID	Name	Gateway EUI	Frequency plan	Status		
ic880a-ttn-v3-pier	ic880a-ttn-v3-pier	BB555A0000000000	EU_863_870_TTN	Connected		

Figura 6.6 Registrazione di un nuovo gateway.

Nella pagina di registrazione *Add Gateway*, è necessario compilare il modulo con le seguenti informazioni (Figura 6.7):

- *Gateway EUI*: questo è un identificatore unico per il gateway. Utilizzare il gateway ID del file `global_conf.json` che si trova nel Semtech UDP Packet Forwarder come spiegato nel capitolo 7 dedicato al gateway iC880A e modificarlo. Per esempio, l'ID di default **AA555A0000000000**, deve diventare **BB555A0000000000** come illustrato nella Figura 6.8. Ovviamente questo ID deve essere unico quindi è possibile scegliere qualsiasi numero.
- Una volta modificato, bisogna riportare lo stesso ID anche nel file `local_conf.json` nella stessa cartella.
Il Semtech UDP Packet Forwarder (con file `global_conf.json`) è disponibile a questo link:
https://github.com/Lora-net/packet_forwarder

Add gateway

General settings

Owner *

Gateway ID ⓘ *

Gateway EUI ⓘ

Gateway name ⓘ

Figura 6.7 La pagina di registrazione di un nuovo gateway.

- Una volta incollato il gateway ID nella casella Gateway EUI, questo diventerà un identificatore unico EUI, a cui TTN assegnerà una chiave unica di accesso.

```

"gateway_conf": {
  "gateway_ID": "BB555A0000000000",
  "server_address": "eu1.cloud.thethings.network",
  "serv_port_up": 1700,
  "serv_port_down": 1700,
  "servers": [
    {
      "gateway_ID": "BB555A0000000000",
      "server_address": "eu1.cloud.thethings.network",
      "serv_port_up": 1700,
      "serv_port_down": 1700,
      "serv_enabled": true
    }
  ]
}

```

Figura 6.8 La sezione gateway ID del file `global_conf.json` del Semtech UDP Packet Forwarder con l'ID modificato.

- *Gateway ID*: scegliere un nome creativo con lettere minuscole e trattini, ad esempio: `ic880a-ttn-v3-pier`. Si fa notare che, una volta assegnato l'ID, non lo si può più riutilizzare neanche dopo aver eliminato il gateway.
- *Description*: in questa casella è possibile inserire una descrizione per il gateway. Per esempio, noi abbiamo usato il nome `"ic880a-ttn-pier"`, ma potrebbe essere un qualsiasi altro nome a piacere.
- *Frequency Plan*: (scendere nella pagina) qui è importante scegliere un piano di frequenza dall'elenco a discesa. Nel nostro caso deve essere *Europe 863 - 870 MHz (SF9 for RX2 -recommended)*.
- *Gateway server address*: è di default `eu1.cloud.thethings.network`.
- *Gateway name*: un nome a piacere, ad esempio `ic880a-ttn-pier`.

Tutto il resto può rimanere così com'è.

- Una volta avviato il gateway con il Semtech Packet Forwa-

der con Raspberry e l'ic880a (come spiegato nel capitolo 7) si vedrà qualcosa di simile alla Figura 6.9 con l'indicazione *Last seen* e il tempo trascorso dall'ultimo accesso del gateway.

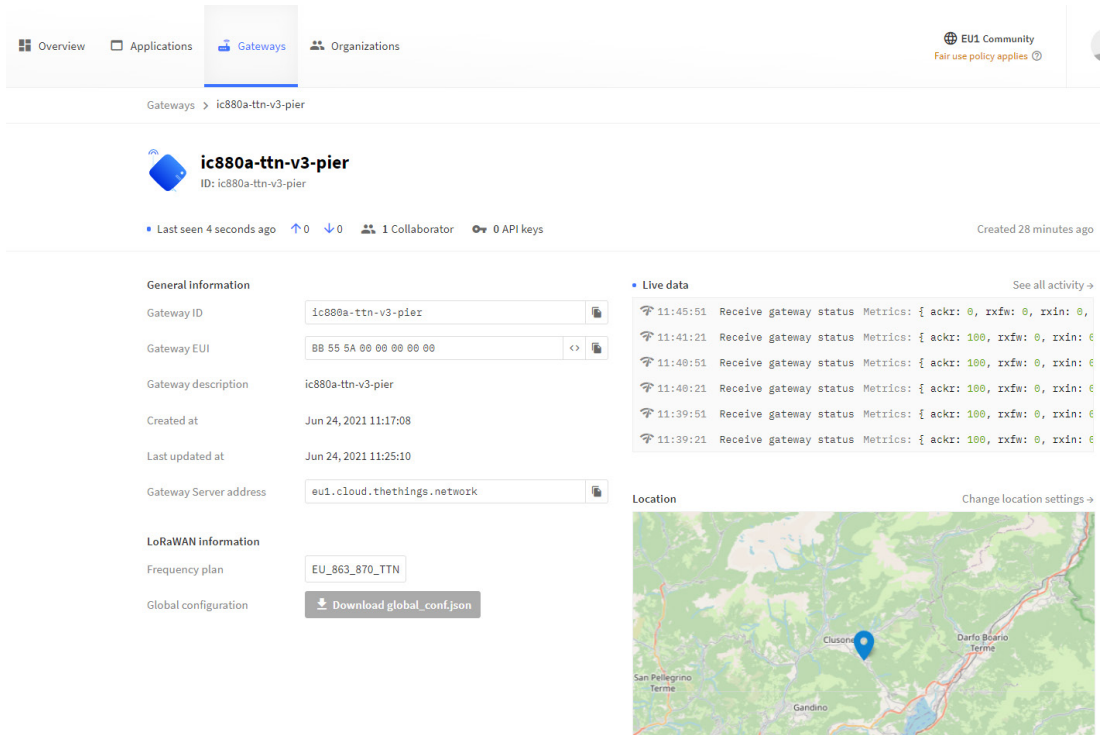


Figura 6.9 Il nostro gateway attivo.

Pagina Gateway Overview

- Nella sezione *Gateway Overview* è possibile visualizzare lo stato del gateway che dovrebbe riportare "Last seen..." (Visto l'ultima volta...) quando il Packet Forwarder è in esecuzione su Raspberry Pi.

In caso contrario, eseguire il Packet Forwarder e attendere che lo stato si aggiorni. A volte è necessario attendere qualche minuto prima che l'ID del gateway si propaghi.

Il *Frequency Plan* e il *Router* sono quelli scelti nella pagina di registrazione.

Alla voce Last seen, si dovrebbe vedere l'ultimo accesso del gateway. Se è previsto dal Packet Forwarder (come nel nostro caso) si vedrà l'aggiornamento ogni 30 secondi.

Alla voce *Live data* si possono leggere i pacchetti ricevuti. Facendo clic su *Live data* si apre una pagina con tutti i pacchetti ricevuti e facendo clic su un pacchetto se ne può visualizzare il contenuto decodificato in formato JSON(Figura 6.10).

Gateways > ic880a-ttn-v3-pier > Live data			
Time	Type	Data preview	Event details
11:58:51	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	<pre>1 { 2 "name": "gs.status.receive", 3 "time": "2021-06-24T09:58:51.132353346Z", 4 "identifiers": [5 { 6 "gateway_ids": { 7 "gateway_id": "ic880a-ttn-v3-pier" 8 }, 9 }, 10 { 11 "gateway_ids": { 12 "gateway_id": "ic880a-ttn-v3-pier", 13 "eui": "B8555A0000000000" 14 }, 15 }, 16], 17 "data": { 18 "@type": "type.googleapis.com/ttn/lorawan.v3.GatewayStatus", 19 "time": "2021-06-24T09:58:51Z", 20 "boot_time": "0001-01-01T00:00:00Z", 21 "versions": { 22 "ttn-lb-gateway-server": "3.13.2" 23 }, 24 "ip": [25 "212.216.218.199" 26], 27 "metrics": { 28 "ackr": 100, 29 "rxfw": 0, 30 "rxin": 0, 31 "rxok": 0, 32 "txin": 0, 33 "txok": 0 34 } 35 } 36 }</pre>
11:58:21	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:49:51	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:49:21	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:48:51	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:48:21	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:47:51	Receive gateway status	Metrics: { ackr: 0, rxfw: 0, rxin: 1, rxok: 0, txin: 0, txok: 0 } Ver	
11:47:21	Receive gateway status	Metrics: { ackr: 0, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } Ver	
11:46:51	Receive gateway status	Metrics: { ackr: 0, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } Ver	
11:46:21	Receive gateway status	Metrics: { ackr: 0, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } Ver	
11:45:51	Receive gateway status	Metrics: { ackr: 0, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } Ver	
11:41:21	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:40:51	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	
11:40:21	Receive gateway status	Metrics: { ackr: 100, rxfw: 0, rxin: 0, rxok: 0, txin: 0, txok: 0 } V	

Figura 6.10 Pacchetti Live data e Event details.

Pagina Location

Facendo clic sul *Location*, si accede alla pagina per impostare la location del gateway (Figura 6.11).

Location

Gateway antenna location settings

Privacy

☐ Publish location

The location of this gateway may be publicly displayed

Location source ②

☐ Update from status messages

Update the location of this gateway based on incoming status messages



Figura 6.11 Pagina del traffico di dati.

Pagina General settings

Se si desidera modificare le impostazioni per il gateway, fare clic su *General settings* che aprirà una pagina *Basis settings* (Figura 6.12). Se non si vuole più usare il gateway, da questa pagina lo si può anche eliminare con il tasto *Delete gateway*.

The screenshot shows the 'General settings' page for a gateway. At the top, there is a navigation bar with three tabs: 'Applications', 'Gateways' (which is active and highlighted with a blue underline), and 'Organizations'. Below the navigation bar, there is a breadcrumb trail: 'Gateways > ic880a-ttn-v3-pier > General settings'. The main content area is titled 'Basic settings' and has a 'Collapse' button on the right. Below the title, there is a subtitle: 'General settings, gateway updates and metadata'. The settings are organized into several sections, each with a title and a help icon (a question mark in a circle). The first section is 'Gateway ID', which has a red asterisk indicating it is required. The input field contains the text 'ic880a-ttn-v3-pier'. The second section is 'Gateway EUI', with an input field containing 'BB 55 5A 00 00 00 00 00'. The third section is 'Gateway name', with an input field containing 'ic880a-ttn-v3-pier'. The fourth section is 'Gateway description', with a text area containing 'ic880a-ttn-v3-pier'. Below the text area, there is a note: 'Optional gateway description; can also be used to save notes about the gateway'. The fifth section is 'Gateway Server address', with an input field containing 'eu1.cloud.thethings.network'. Below the input field, there is a note: 'The address of the Gateway Server to connect to'. The sixth section is 'Require authenticated connection', which has a checkbox labeled 'Enabled'. Below the checkbox, there is a note: 'Controls whether this gateway may only connect if it uses an authenticated Basic Station or MQTT connection'.

Applications Gateways Organizations

Gateways > ic880a-ttn-v3-pier > General settings

Basic settings

Collapse

General settings, gateway updates and metadata

Gateway ID *

ic880a-ttn-v3-pier

Gateway EUI

BB 55 5A 00 00 00 00 00

Gateway name

ic880a-ttn-v3-pier

Gateway description

ic880a-ttn-v3-pier

Optional gateway description; can also be used to save notes about the gateway

Gateway Server address

eu1.cloud.thethings.network

The address of the Gateway Server to connect to

Require authenticated connection

☐ Enabled

Controls whether this gateway may only connect if it uses an authenticated Basic Station or MQTT connection

Figura 6.12 Pagina General settings.

Creazione di un'applicazione

Fino a questo momento il gateway TTN riceve pacchetti di dati grezzi dal gateway fisico (iC880a, per intenderci). Questi pacchetti di dati includono payload e altre informazioni relative al trasmettitore.

In pratica, non siamo ancora in grado di decodificare quello che abbiamo effettivamente mandato come messaggio, ovvero quel **"Hello, world!"** dell'esempio ttn-abp che abbiamo caricato nel nostro end device (si veda il capitolo 4).

Per poter decodificare il messaggio "Hello World!" o qualsiasi altro messaggio che vorremo inviare al gateway, dobbiamo necessariamente creare un'applicazione che decodifichi i payload per estrarre i dati dai pacchetti. Per fare questo, seguire la procedura seguente.

Pagina Applications

Facendo clic su *Applications* della console, possiamo vedere tutte le applicazioni attive al momento o aggiungerne una nuova. Nella Figura 6.13 è presente già un'applicazione. Facendo clic su *Add application* si aprirà una pagina *Add Application*, simile a quella illustrata in Figura 6.14.

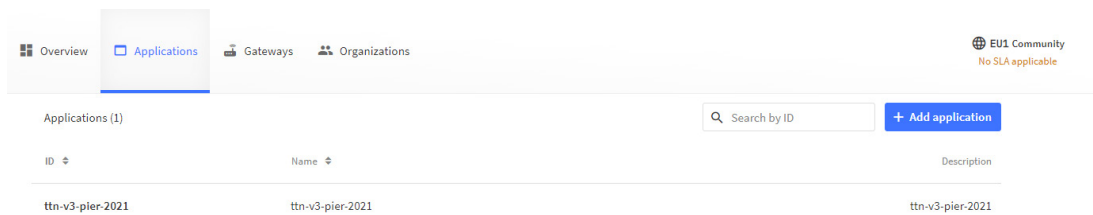


Figura 6.13 Pagina Applications.

Add application

Owner*

piercalderan

Application ID*

ttn-v3-pier-2021

Application name

ttn-v3-pier-2021

Description

ttn-v3-pier-2021

Optional application description; can also be used to save notes about the application

Create application

Figura 6.14 Pagina Add Application.

Nella pagina *Add Application* compilare tutti i campi del modulo come segue:

- *Application ID*: inserire un identificativo univoco della applicazione sulla rete. Nel nostro caso abbiamo scelto: **ttn-v3-pier-2021**. Notare che tutti i caratteri devono essere minuscoli e senza spazi.
- *Application name*: un nome qualsiasi.
- *Description*: questa è una descrizione a piacere della nuova applicazione.
- Fare clic sul pulsante "Create application" per creare l'applicazione. Si verrà indirizzati alla pagina di anteprima dell'applicazione che riassume tutti i dati dell'applicazione (Figura 6.15).

Applications > ttn-v3-pier-2021

ttn-v3-pier-2021
ID: ttn-v3-pier-2021

• Last seen info unavailable • 2 End devices • 1 Collaborator • 2 API keys Created 28 days ago

General information

Application ID: ttn-v3-pier-2021

Created at: May 27, 2021 08:38:41

Last updated at: May 27, 2021 15:23:41

Live data See all activity →

Waiting for events from ttn-v3-pier-2021...

Figura 6.15 Pagina Application Overview.

- Da questa pagina si può accedere alla sezione *End devices*, per accedere alla pagina omonima (Figura 6.16). Per il momento concentriamoci sulla sezione Devices perché avremo modo di tornare a vedere la sezione *Access Key*.
- Nella sezione *End devices* si può notare che non c'è ancora nessun dispositivo registrato. Quindi, per registrarne uno, fare clic sul collegamento *Add end device* che aprirà la pagina *Register end device*, ovvero quella che serve a registrare un dispositivo, nella fattispecie il nostro end device.

Overview Applications Gateways Organizations

Applications > ttn-v3-pier-2021 > End devices

End devices (0)

Search by ID Import end devices Add end device

ID	Name	DevEUI	JoinEUI	Last seen
No items found				

Figura 6.16 Sezione End devices.

Pagina Register end device

Nella pagina *Register end device* (Figura 6.17), si può scegliere una marca di dispositivo commerciale dal menu a tendina *Brand...*

Register end device

[From The LoRaWAN Device Repository](#) [Manually](#)

1. Select the end device

Brand ? *

▼

Cannot find your exact end device? [Get help here](#) and [try manual device registration](#).

2. Enter registration data

Please choose an end device first to proceed with entering registration data

[Register end device](#)

Figura 6.17 Pagina Register Device.

Ovviamente per il nostro dispositivo auto costruito, scegliere l'opzione *Manually* per compilare manualmente i campi come segue:

- *Activation mode*: selezionare ABP.
- *LoRaWAN version*: selezionare 1.0.2.

A questo punto basta fare clic sul pulsante *Start* per continuare la registrazione del dispositivo. Si verrà indirizzati alla pagina *Basic settings*. Compilare i campi come segue (Figura 6.18).

- *End device ID*: un nome qualsiasi, per esempio, device-001.
- *DevEUI*: un ID qualsiasi, per esempio AAAAAAAAAA.
- *End device name*: un nome qualsiasi, per esempio, devi-

ce-001.

- *End device description*: un nome qualsiasi, per esempio, device-001.

Al termine premere sul tasto "Network layer settings".

Register end device

From The LoRaWAN Device Repository [Manually](#)

1 **Basic settings**
End device ID's, Name and Description

2 **Network layer settings**
Frequency plan, regional parameters, end device class and session keys.

End device ID ⓘ *

DevEUI ⓘ

AA AA AA AA AA AA AA AA

End device name ⓘ

End device description ⓘ

device-001

Optional end device description; can also be used to save notes about the end device

Figura 6.18 Pagina *Basic settings*.

Network layer settings

Nella pagina *Network layer settings* (Figura 6.19) compilare i campi in questo modo:

- *Frequency plan*: Europe 863-870 MHz (SF9 for RX2 - recommended).
- *Regional Parameters version*: PHY 1.0.2 REV A.
- *Device address*: fare clic sulle frecce per generare un indirizzo casuale.
- *NwkSKey*: fare clic sulle frecce per generare una chiave casuale.

Al termine premere sul tasto "Application layer settings".

✓ Basic settings

End device ID's, Name and Description

2 Network layer settings

3 Application layer settings

Application session key to encrypt/decrypt LoRaWAN payload.

Frequency plan ⓘ *

Europe 863-870 MHz (SF9 for RX2 - recommended) ▼

LoRaWAN version ⓘ *

MAC V1.0.2 ▼

Regional Parameters version ⓘ *

PHY V1.0.2 REV A ▼

LoRaWAN class capabilities ⓘ

☐ Supports class B

☐ Supports class C

Device address ⓘ *

26 0B 39 48 ↻

NwkSKey *

E7 F3 7A 84 0E AC 80 8E 1B DF FE AF AE FE A0 7A ↻

Advanced settings ▼

< Basic settings

Application layer settings >

Figura 6.19 Impostazioni Network layer settings.

Application layer settings

Nella pagina *Application layer settings* (Figura 6.20) fare clic sulle frecce della casella *AppSKey* per generare una chiave casuale. Premere sul tasto "Add end device" per aggiungere il dispositivo.

✓ Basic settings
End device ID's, Name and Description

✓ Network layer settings
Frequency plan, regional parameters, end device class and session keys.

3 Application layer settings
Application session key to encrypt/decrypt LoRaWAN payload.

Skip payload encryption and decryption

☐ Enabled
Skip decryption of uplink payloads and encryption of downlink payloads

AppSKey ⓘ *

0F 83 20 C0 1F 20 2E 3C 64 CC 8B CB 3F D4 06 56

< Network layer settings

Add end device

Figura 6.20 Impostazioni Application layer settings.

Come vedremo fra poco, i dati delle chiavi e l'indirizzo generati casualmente andranno copiati e incollati nell'esempio **ttn-abp**.

Configurazione dell'esempio ttn-abp-ino

Dalla pagina Overviews dell'end device appena creato è possibile ricavare le informazioni che ci servono per il nostro esempio.

Questa operazione è parecchio facilitata perché basterà fare clic sull'icona "occhio" per vedere in chiaro le chiavi e fare clic sull'icona "<>" per cambiare lo stile di visualizzazione da HEX a C (Figura 6.21).

A questo punto basta copiare con l'icona copia il *Device Address* e incollarlo nel codice del file di esempio **ttn-abp** in corrispondenza della variabile:

```
static const u4_t DEVADDR = 0x260B3948;
```

Attenzione a mettere il prefisso 0x davanti al Device Address.

Copiare, sempre con l'icona copia, i campi *NwkSKey* (*Network Session Key*) e *AppSKey* (*App Session Key*) in corrispondenza delle variabili:


```
static const PROGMEM u1_t NWKSKKEY[16] = { 0xE7, 0xF3, 0x7A, 0x84, 0x0E,
0xAC, 0x80, 0x8E, 0x1B, 0xDF, 0xFE, 0xAF, 0xAE, 0xFE, 0xA0, 0x7A };

e...

static const u1_t PROGMEM APPSKKEY[16] = { 0x0F, 0x83, 0x20, 0xC0, 0x1F,
0x20, 0x2E, 0x3C, 0x64, 0xCC, 0x8B, 0xCB, 0x3F, 0xD4, 0x06, 0x56 };
```

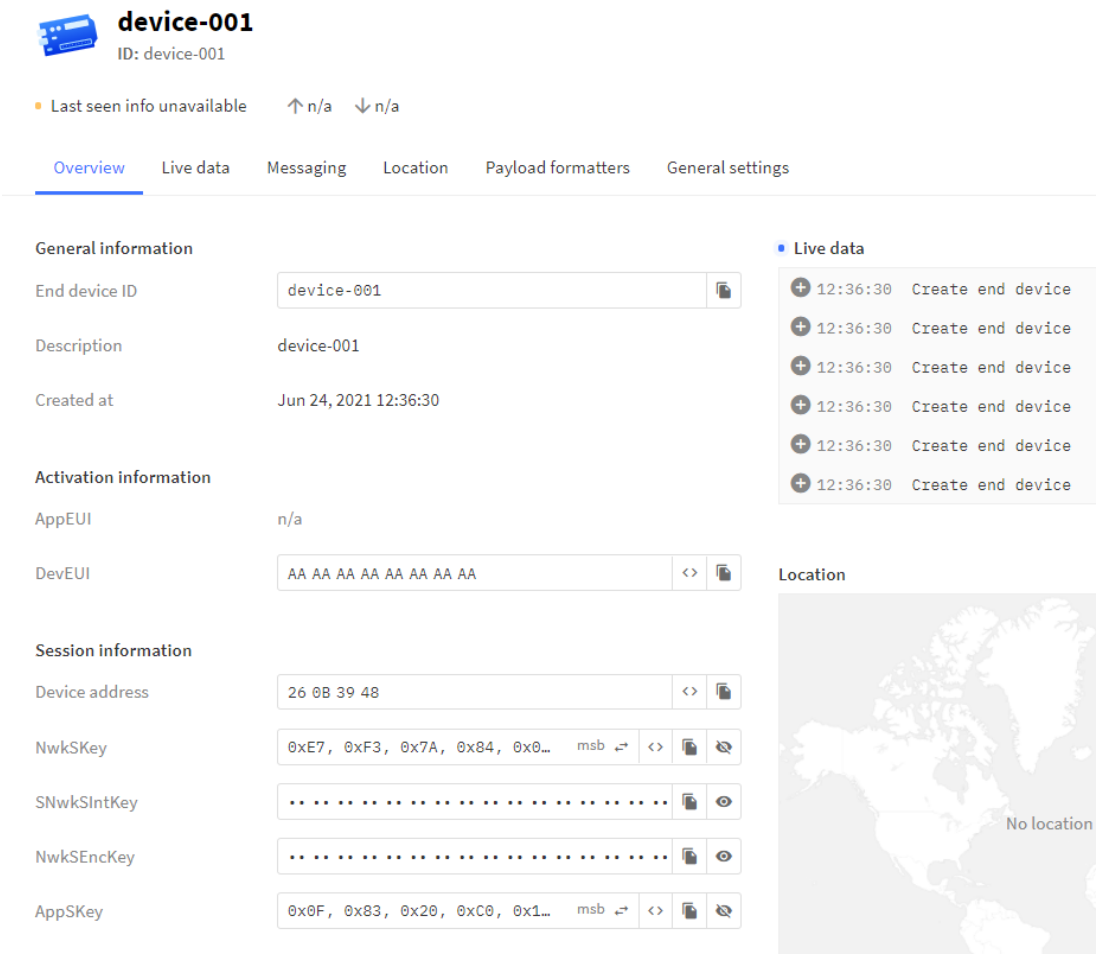


Figura 6.21 Pagina Overview dell’end device.

Ora il file di esempio ttn-abp è configurato correttamente con l’indirizzo del dispositivo e le due chiavi di accesso. Per poter connettere il nostro end device all’applicazione, basta caricarlo nuovamente dall’IDE di Arduino e andare a vedere quello che succede su TTN.

Quello che dovremo vedere arrivare è il messaggio “Hello, world!” così come è stato previsto nell’esempio.

Live data

Per visualizzare il messaggio in arrivo, basta fare clic sull’area Live data o dal menu laterale. Osservando la Figura 6.22, si noterà che sono stati ricevuti cinque payload, uno ogni 30 secondi circa, tutti uguali e contenenti il messaggio:

48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

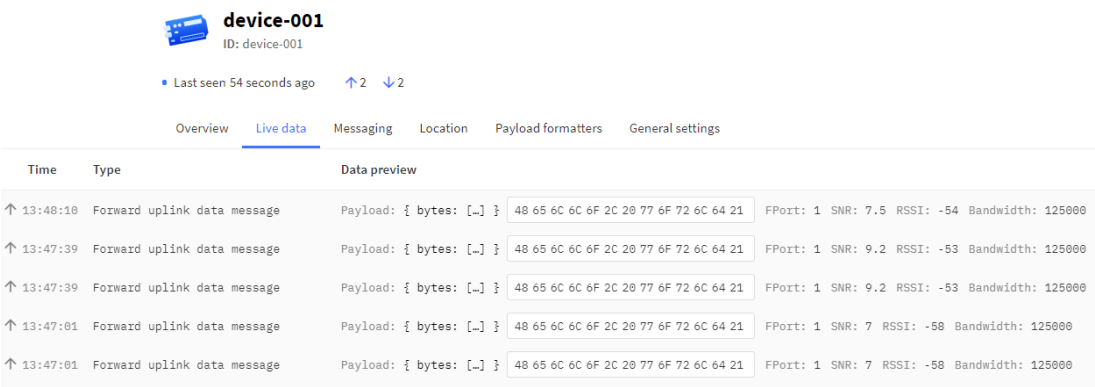


Figura 6.22 Visualizzazione dei payload.

Visto così il messaggio non ci dice nulla, ma se convertiamo i numeri esadecimali in caratteri di testo ASCII, usando un convertitore online (es. <https://codebeautify.org/hex-string-converter>) vedremmo proprio il messaggio “Hello, world!”. Per fare la copia del payload, basta fare clic sul payload stesso e utilizzare CTRL+C (Figura 6.23).

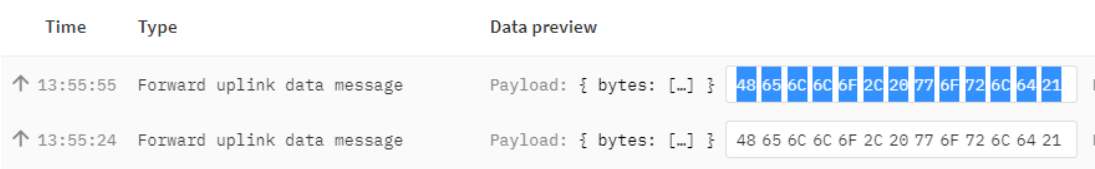
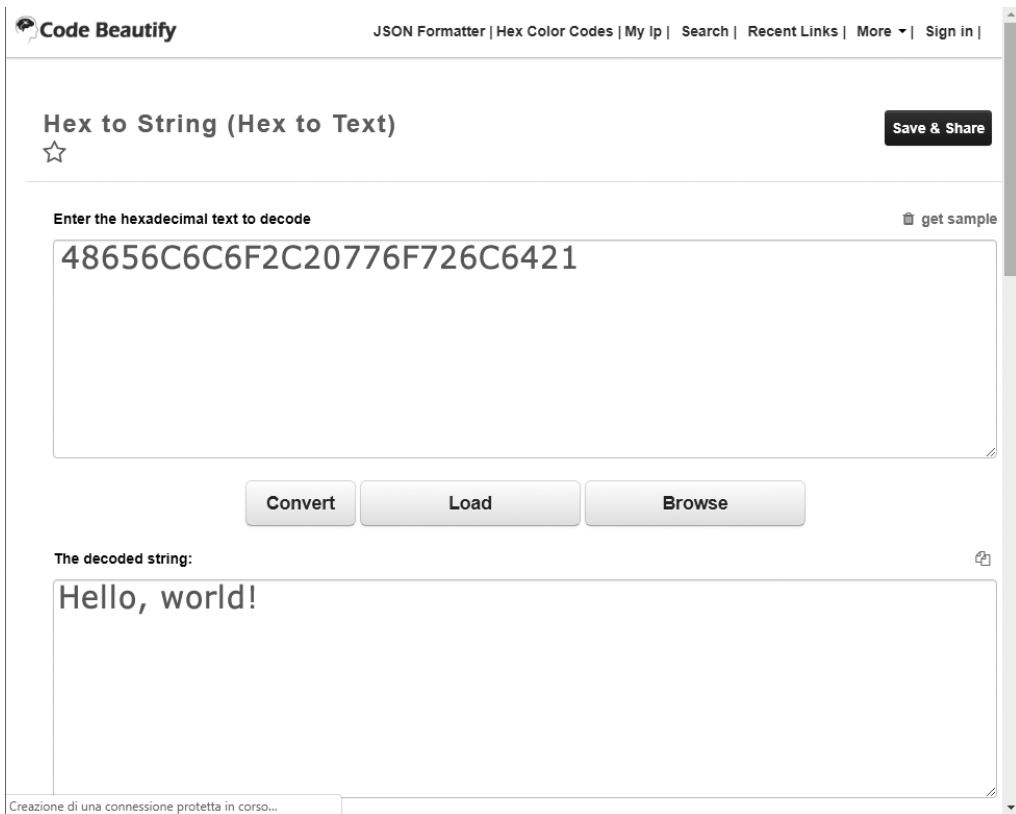


Figura 6.23 Copia del payload.

Una volta copiato il payload (48656C6C6F2C20776F726C6421), basterà incollarlo sul convertitore online Hex to String e si vedrà il

testo del payload convertito (Figura 6.24).



The screenshot shows the 'Code Beautify' website's 'Hex to String (Hex to Text)' tool. The header includes navigation links like 'JSON Formatter', 'Hex Color Codes', 'My Ip', 'Search', 'Recent Links', 'More', and 'Sign in'. The main heading is 'Hex to String (Hex to Text)' with a star icon and a 'Save & Share' button. Below the heading, there's a prompt 'Enter the hexadecimal text to decode' and a 'get sample' link. A large text input field contains the hexadecimal string '48656C6C6F2C20776F726C6421'. Below this field are three buttons: 'Convert', 'Load', and 'Browse'. Underneath the buttons, it says 'The decoded string:' followed by a text area containing 'Hello, world!'. At the bottom left, a small status bar indicates 'Creazione di una connessione protetta in corso...'. The right side of the page has a vertical scrollbar.

Figura 6.24 Il payload esadecimale convertito in stringa.

Payload Formatters

Se da una parte la soluzione del convertitore online può andare bene in questa fase di test, non potrà andare bene quando vorremo visualizzare direttamente i messaggi dei payload nella pagina Data.

Per facilitare questa operazione di conversione, TTN mette a disposizione una pagina di *Payload formatters*, raggiungibile facendo clic su *Payload Formatters* dalla pagina *Application Overview* (Figura 6.25). Qui sono disponibili i payload formatter per Uplink e Downlink.

Siccome ci serve decodificare ogni uplink, sceglieremo il payload formatter Uplink.

Default uplink payload formatter

1 You can use the "Payload formatter" tab of individual end devices to test uplink payload formatters and to define individual payload formatters

Setup

Formatter type *

Javascript

Formatter parameter *

```
1 function decodeUplink(input) {
2   return {
3     data: {
4       bytes: input.bytes
5     },
6     warnings: [],
7     errors: []
8   };
9 }
```

Save changes

Figura 6.25 Payload formatter Uplink.

Nella pagina *Default uplink payload formatter*, scriveremo una funzione JavaScript per la conversione automatica del payload da esadecimale a testo, sostituendo la funzione JavaScript esistente con questa (Figura 6.26):

```
function Decoder(bytes, port) {
  return {
    message: String.fromCharCode.apply(null,bytes)
  }; }

function decodeUplink(input) {
  var data = input.bytes;
  var valid = true;
  if (typeof Decoder === "function") {
    data = Decoder(data, input.fPort);
  }
  if (valid) {
    return {
```

```

    data: data
  };
} else {
  return {
    data: {},
    errors: ["Invalid data received"]
  };
}
}
}

```

Questa funzione decodificherà il payload in una stringa leggibile. Fare clic sul pulsante “*Save changes*” per salvare le modifiche.

```

1 function Decoder(bytes, port) {
2   return {
3     message: String.fromCharCode.apply(null,bytes)
4   }; }
5
6 function decodeUplink(input) {
7   var data = input.bytes;
8   var valid = true;
9
10  if (typeof Decoder === "function") {
11    data = Decoder(data, input.fPort);
12  }
13  if (valid) {
14    return {
15      data: data

```

Figura 6.26 La funzione di conversione del payload.

Tornando alla pagina Live Data dell’end device si vedranno i payload convertiti in testo nella come message: “Hello, world!” (Figura 6.27).



device-001
ID: device-001

Last seen 12 seconds ago ↑ 75 ↓ 31 Created 2 hours ago

Overview **Live data** Messaging Location Payload formatters General settings

Time	Type	Data preview
↑ 14:53:55	Forward uplink data message	Payload: { message: "Hello, world!" } 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21 FPort: 1 SNR: 7.5 RSSI: -55 Bandwidth: 125000

☐ Verbose stream

Figura 6.27 Il payload esadecimale convertito in testo.

Visualizzare i dati dei sensori

Giunti a questo punto pensiamo che sia tutto pronto per poter gestire uno o più end device dotati di sensori in grado di trasmettere i loro dati a un gateway e da qui alla rete TTN e di visualizzare nella pagina di *Live data*.

Lo schema di funzionamento di tutto questo è schematizzato in Figura 6.28.

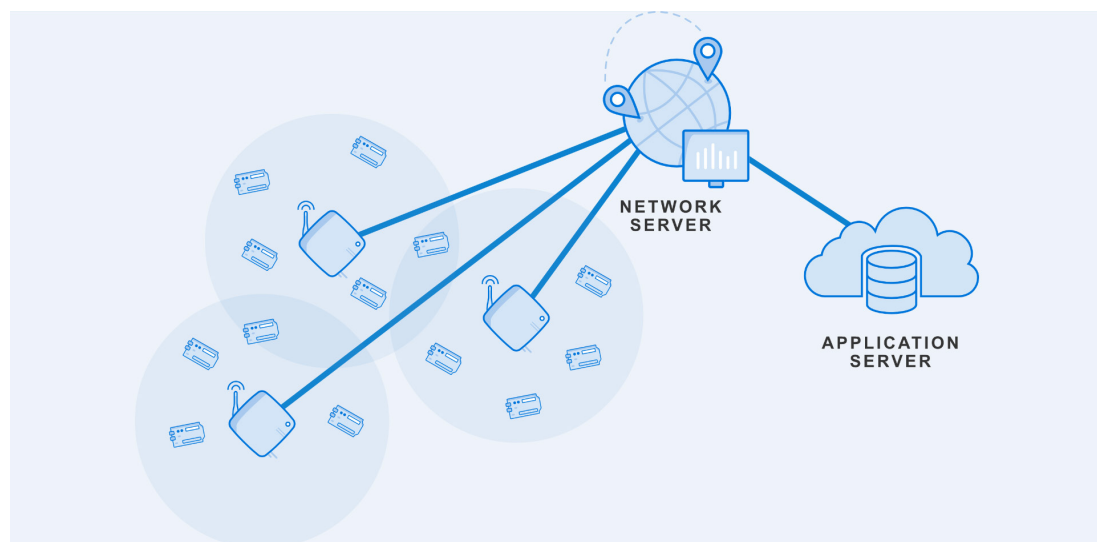


Figura 6.28 *Il network di TTN.*

Si possono usare decine, centinaia, migliaia di end device tutti collegati a uno o più gateway. I gateway sono connessi al server di rete, in questo caso TTN, collegato a sua volta a un server per le applicazioni.

Questa organizzazione di rete permette di creare un numero indefinito di gateway e un numero indefinito di applicazioni, tutte contenenti un numero indefinito di device. Infine, ogni device può essere collegato a un numero elevato di sensori.

Per gli scopi di questo libro, ci limiteremo a due gateway, due applicazioni e cinque end device.

Visualizzazione di temperatura e umidità

Abbiamo pensato di modificare il file di esempio `ttn-abp` in qualcosa di utile, trasformando il payload "Hello, world!" i dati prove-

nienti da un sensore di temperatura e umidità.

A questo scopo ci siamo avvalsi dell'aiuto di un sensore molto diffuso ed economico e, soprattutto, compatibile con Arduino e affini, il **DHT11**.

Il sensore DHT11

Il DHT11 è un sensore di temperatura e umidità con uscita dei dati in formato digitale (Figura 6.29).

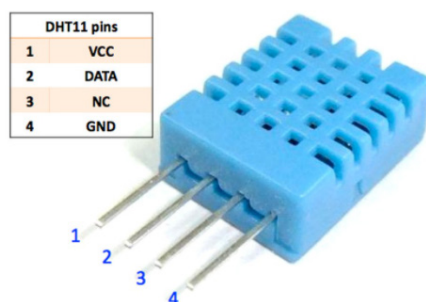


Figura 6.29 Il sensore di temperatura e umidità DHT11.

Il sensore utilizza una tecnica digitale esclusiva *OneWire* che permette di trasmettere i dati di temperatura e umidità su un unico terminale di uscita. I suoi elementi sensibili sono connessi con un processore 8 bit single-chip.

Ogni sensore di questo modello è compensato in temperatura e calibrato in una apposita camera di calibrazione che determina in modo preciso il valore di calibrazione il cui coefficiente viene salvato all'interno della memoria OTP.

Il package con quattro pin in linea ne rendono facile la connessione, ma solo tre sono collegati: VCC, DATA e GND.

■ Caratteristiche

Gamma completa di temperatura compensata

Umidità relativa e temperatura

Segnale digitale calibrato

Eccezionale stabilità a lungo termine

Componenti aggiuntivi non necessari

Distanza di trasmissione lunga

Basso assorbimento
Contenitore con 4 pin

Modello	DHT11
Alimentazione	3-5.5V DC
Segnale di uscita	digitale del segnale tramite single-bus
Elemento sensibile	Resistenza in Polimero
Campo di misura umidità	20-90% di umidità relativa, temperatura di 0-50 gradi Celsius
Precisione umidità	umidità + -4% RH (Max + -5% di umidità relativa)
Precisione temperatura	+ -2.0Celsius
Risoluzione o sensibilità	umidità 1% di umidità relativa, temperatura 0.1 Celsius
Umidità isteresi	+ -1% RH
Stabilità a lungo termine	+ -0.5% UR/anno
Tempo di rilevazione	medio: 2s
Dimensioni	12 x 15,5 x 5,5 millimetri

Collegamento del DHT11

La Figura 6.30 mostra il facile collegamento al nostro end device. Si possono saldare tre fili al DHT11 e direttamente alle piazzole del circuito stampato, oppure infilati (sconsigliato) nei pin femmina della striscia.

I tre pin da collegare sono:

DHT11	End device
VCC	pin 7
DATA	pin 4
GND	pin 8

■ Attenzione!

Per funzionare il DHT11 ha bisogno di un resistore da 4.7-10 K Ω fra il terminale DATA e VCC. Si può saldare direttamente al DHT11.

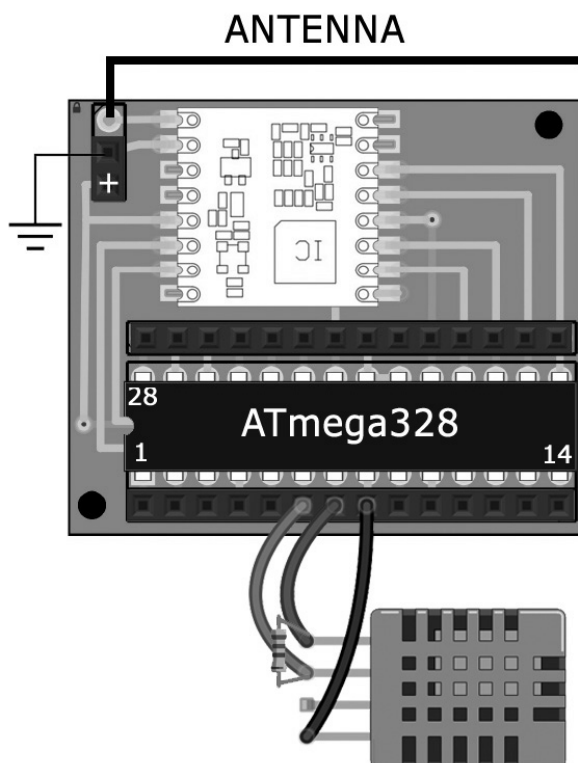


Figura 6.30 Il collegamento del sensore DHT11 all'end device.

Libreria DHT11

Per far funzionare il DHT11 è necessario installare una delle tante librerie per Arduino che si trovano online. La libreria che abbiamo installato è quella che si trova al seguente link:

<http://arduino.cc/playground/Main/DHTLib>

Una volta installata la libreria pensiamo sia utile eseguire un test con Arduino UNO, collegando alimentazione il pin di uscita del DHT11 al pin digitale 4 (Figura 6.31) e il resistore fra VCC e DATA. In questo modo è più facile vedere se tutto funziona tramite il monitor seriale. Lo sketch di prova si chiama **dht11_test** e si trova negli esempi della libreria dal menu *File > Esempi > DHTLib*.

■ Attenzione!

Per usare il DHT11 collegato al pin digitale 4, è necessario cambiare la riga di codice nello sketch di prova in questo modo: **#define DHT11_PIN 4**.

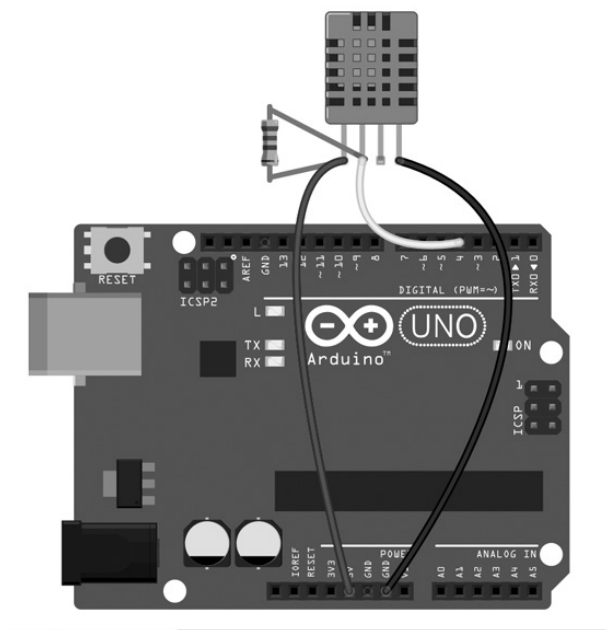


Figura 6.31 Il collegamento del DHT11 ad Arduino UNO.

Una volta caricato lo sketch, se tutto funziona, dovremmo vedere sul monitor seriale i dati di temperatura e umidità, come illustrato in Figura 6.32.

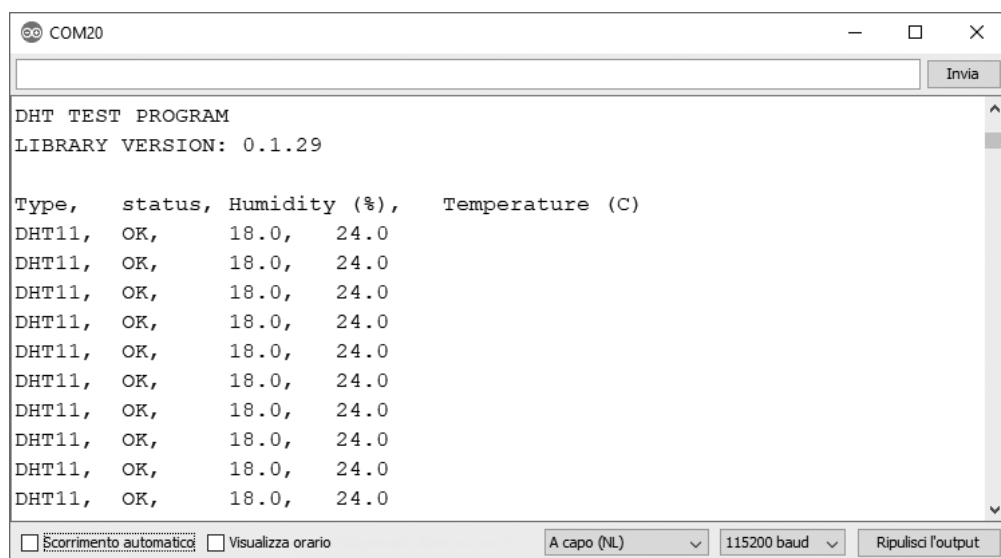


Figura 6.32 Il monitor seriale con i dati di temperatura e umidità del DHT11.

Collegamento del DHT11 a TTN

Giunti a questo punto possiamo modificare il nostro file di esempio **ttn-abp** adattando il codice di rilevamento della temperatura e umidità del DHT11 per poterli inviare al nostro gateway e da qui alla nostra applicazione su TTN.

Abbiamo chiamato lo sketch modificato **ttn-abp-dht11** e lo abbiamo reso disponibile nelle risorse del libro presso il sito dell'autore: www.pierduino.com

Ad ogni modo le uniche righe di codice che andremo ad aggiungere per il rilevamento dal DHT11 sono le seguenti.

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 4
int humi = DHT.humidity;
int temp = DHT.temperature;
```

Ora vediamo dove inserirle e come fare per inviarle correttamente al gateway.

Lo sketch **ttn-abp-dht11**

La prima cosa da aggiungere allo sketch subito dopo le altre librerie è la libreria DHT con la direttiva *#include*:

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <dht.h>
```

Quindi si crea un'istanza DHT:

```
dht DHT;
```

Poi si definisce il pin di ingresso dei dati dal DHT11:

```
#define DHT11_PIN 4
```

Poi si devono aggiungere le seguenti righe per inizializzare la variabile **mydata** come array di 8 byte e le variabili **temp** e **hum** come **uint8_t**:

```
uint8_t mydata[] = "00000000";
```

```
uint8_t temp = "0";  
uint8_t humi = "0";
```

Come abbiamo avuto modo di vedere tutto il lavoro di invio dei dati viene svolto all'interno della funzione *do_send*, per cui il codice per il rilevamento va inserito così (testo in grassetto):

```
void do_send(osjob_t* j){  
int chk = DHT.read11(DHT11_PIN);  
humi = DHT.humidity;  
temp = DHT.temperature;  
  
if (LMIC.opmode & OP_TXRXPEND) {  
    Serial.println("OP_TXRXPEND, not sending");  
} else {  
    mydata[0]=0x54;  
    mydata[1]=temp;  
    mydata[2]=0x48;  
    mydata[3]=humi;  
    LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0); }  
}
```

Dopo che **temp** e **humi** hanno memorizzato rispettivamente il valore di temperatura e umidità, è possibile mettere i loro valori all'interno dell'array mydata:

```
mydata[0]=0x54;  
mydata[1]=temp;  
mydata[2]=0x48;  
mydata[3]=humi;
```

Per comodità, abbiamo aggiunto la lettera "T" e la lettera "H" davanti ai valori di temperatura e umidità ovvero:

0x54 = decimale 84 = carattere ASCII "T"

0x48 = decimale 72 = carattere ASCII "H"

Pertanto, il payload dovrebbe avere questo contenuto:

54 (xx) 48 (xx) ovvero T xx H xx

... dove **xx** sono i valori di temp e humi.

Essendo l'array di 8 byte avremo il payload completo:

TxxHxx0000

Supponendo che la temperatura sia 23 e l'umidità 68, dovremmo leggere:

T23H680000

... da cui estrarremo solo i dati:

T23 e H68

Gli altri zeri servono per eventuali dati di altri sensori o semplicemente per il padding.

Inviando questo payload saremo in grado di decodificare il pacchetto nella nostra applicazione su TTN, come vedremo fra poco.

Payload formatter per DHT11

Per la decodifica dei dati di temperatura e umidità è necessario modificare i formati del payload, come abbiamo fatto per la stringa "Hello, World!" (vedi paragrafo Payload formatters).

La funzione JavaScript del payload per DHT11 è stata modificata in questo modo:

```
function Decoder(bytes, port) {  
  var decoded = {};  
  
  decoded.d1 = bytes[0];  
  decoded.d2 = bytes[1];  
  decoded.d3 = bytes[2];  
  decoded.d4 = bytes[3];  
  decoded.d5 = bytes[4];  
  decoded.d6 = bytes[5];  
  decoded.d7 = bytes[6];  
  decoded.d8 = bytes[7];  
  
  return decoded;  
}
```

Come si può vedere la variabile decoded è un array che memorizza i byte del payload da 0 a 7 (totale 8 byte), per cui dovremo avere i seguenti campi decodificati:

"d1": 84 = T

"d2": 23 = 23

"d3": 72 = H
"d4": 68 = 68
"d5": 48 = 0
"d6": 48 = 0
"d7": 48 = 0
"d8": 48 = 0

Esattamente come li abbiamo spediti. La Figura 6.33 mostra la pagina con la nuova funzione che abbiamo appena visto.

Formatter parameter*

```
1 function Decoder(bytes, port) {  
2   var decoded = {};  
3   decoded.d1 = bytes[0];  
4   decoded.d2 = bytes[1];  
5   decoded.d3 = bytes[2];  
6   decoded.d4 = bytes[3];  
7   decoded.d5 = bytes[4];  
8   decoded.d6 = bytes[5];  
9   decoded.d7 = bytes[6];  
10  decoded.d8 = bytes[7];  
11  return decoded;  
12 }
```

Figura 6.33 Payload formatter parameter.

A questo punto non ci resta che inviare i dati dal nostro end device collegato al DHT11 e vedere cosa succede nella pagina *Live data* del dispositivo **device-001**.

Come si può vedere (Figura 6.34) il payload ricevuto è proprio...

54 17 48 44 30 30 30 30

... e i dati convertiti sono proprio quelli forniti dalla funzione:

"d1": 84,
"d2": 23,
"d3": 72,
"d4": 68,
"d5": 48,
"d6": 48,
"d7": 48,
"d8": 48

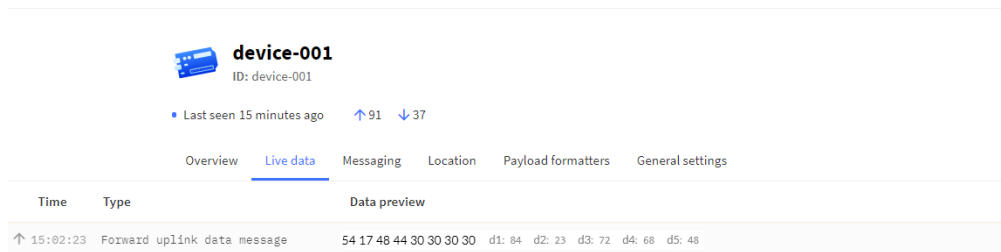


Figura 6.34 I dati convertiti nella pagina Payload Formats.

Aggiungere un LED di monitoraggio

Il nostro end device per il rilevamento della temperatura e dell'umidità è pronto, ma se vogliamo la ciliegina sulla torta, ci servirebbe un LED che ci indichi lo stato di invio dei dati.

Per fare questo bastano poche righe di codice e un LED con un resistore in serie da 120 Ω che collegheremo alla porta digitale 5 (pin fisico 11) del chip ATmega328, come illustrato in Figura 6.35. Il codice da aggiungere è questo:

```
#define LED_PIN 5 all'inizio sotto #define DHT11_PIN 4
digitalWrite(LED_PIN,1); sotto Serial.println("Packet queued");
pinMode (LED_PIN, OUTPUT); nel Setup()
digitalWrite(LED_PIN,0); prima di os_setTimedCallback(&sendjob, os_getTi-
me()+sec2osticks(TX_INTERVAL), do_send);
```

In questo modo vedremo il LED accendersi quando viene inviato un pacchetto e spegnersi quando è stato inviato.

Notare che avendo collegato il LED sul pin digitale 5, è possibile trascurare il resistore perché possiamo sfruttare la PWM del pin 5 e quindi scrivere, per esempio, `analogWrite(LED_PIN, 25)` o un valore anche più basso per avere meno tensione in uscita e non bruciare il LED.

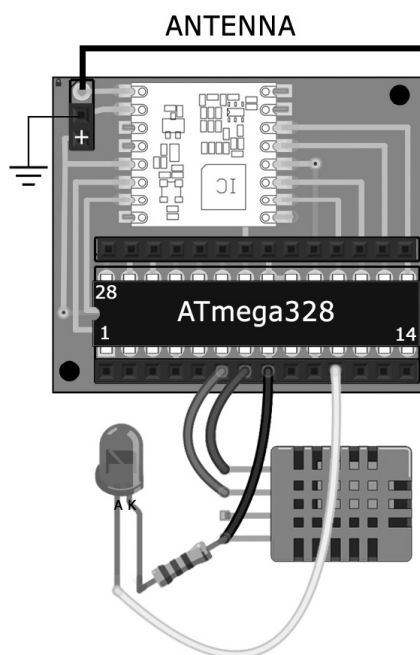


Figura 6.35 Il monitor seriale con i dati di temperatura e umidità del DHT11.

■ **Attenzione!**

Volendo aggiungere altri sensori a un solo end device, si potrebbe arrivare a creare un payload anche molto lungo. Consigliamo di non esagerare con la lunghezza del pacchetto da inviare, in quanto LoRa è un protocollo lento e per lunghe distanze si potrebbero perdere dati. Da considerare anche il fatto che il chip ATmega328 o Arduino è limitato in memoria e uno sketch di larghe dimensioni potrebbe causare problemi di blocco. Il consiglio è quello di creare più end device con al massimo 4 sensori ciascuno.

Il nostro sketch **ttn-abp-dht11** usa 15966 byte (51%) dello spazio disponibile per i programmi. Il massimo è 30720 byte. Le variabili globali usano 948 byte di memoria dinamica. Rimanendo in questi range di occupazione della memoria, non dovrebbero insorgere problemi.

Visualizzare i dati con Node-RED

Fino a questo momento abbiamo fatto tutto il possibile per una visualizzazione “human readable”, come dicono gli inglesi, dei dati su TTN. Siccome che non ci possiamo accontentare di questo, vorremmo proporre un’alternativa che a nostro parere è molto accattivante: la visualizzazione dei dati tramite Node-RED. Tanto per essere chiari, quello che otterremo alla fine di questo capitolo è un risultato simile alla Figura 6.36.

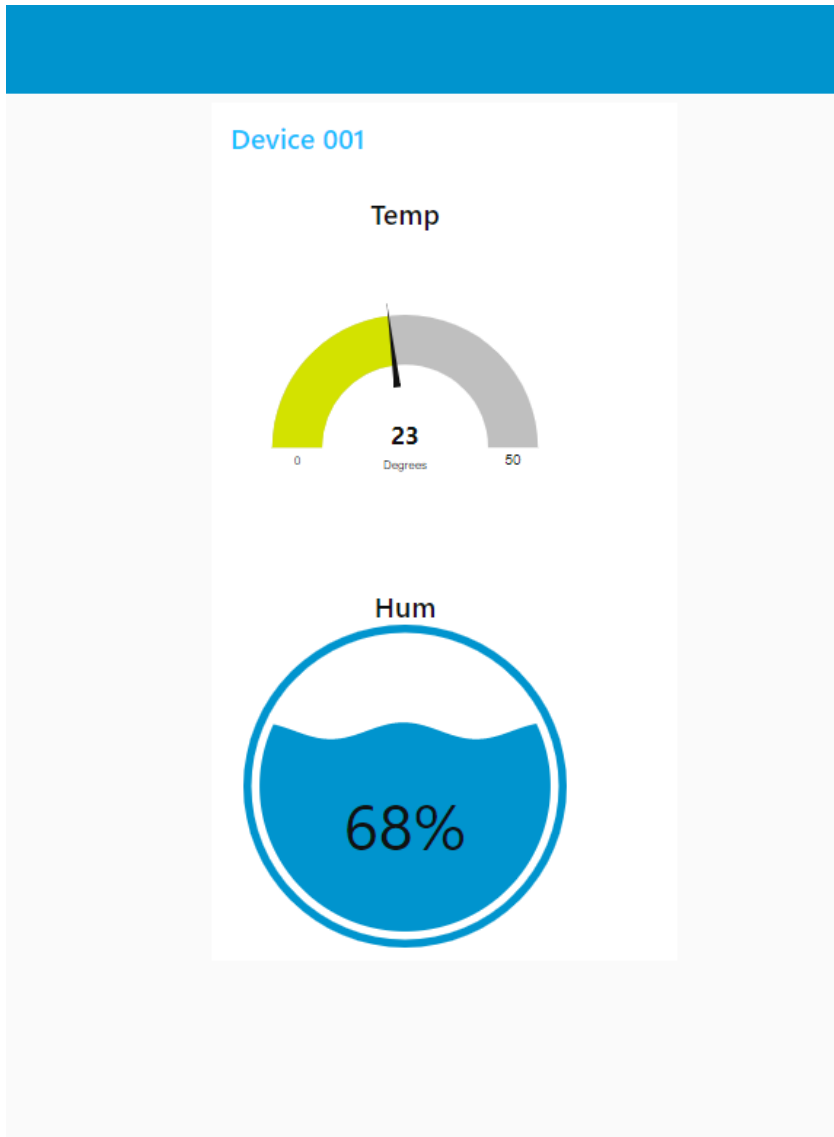


Figura 6.36 I dati visualizzati con Node-RED.

Node-RED

Node-RED (sito ufficiale: <https://nodered.org>) è uno strumento di programmazione per collegare insieme dispositivi hardware, API e servizi online, con un modus operandi molto facile e accattivante. Fornisce un editor basato su browser che semplifica il collegamento dei flussi utilizzando l'ampia gamma di nodi nella dashboard (tavolozza) che può essere distribuita online in un solo clic.

Prima di iniziare

Node-RED è basato su **Node.js**, sfruttando appieno il suo modello non bloccante basato sugli eventi. Ciò lo rende ideale per l'esecuzione su hardware come Raspberry Pi e nel cloud.

Per installare Node-RED localmente è necessaria una versione supportata di Node.js.

Versioni del nodo supportate

Node-RED attualmente consiglia Node 10.x LTS.

Si consiglia di rimanere aggiornati con le versioni di Node.js.

Versione	Livello di supporto
<7.x	non supportato
8.x	supportato
9.x	non supportato
10.x	Consigliato
11.x	non supportato
12.x	provvisorio

Installazione di Node.js

Il sito ufficiale di Node.js (<https://nodejs.org/it>) offre le guide per l'installazione di Node.js su una vasta gamma di sistemi operativi. Per chi usa Windows, per esempio, deve scaricare dal sito ufficiale la versione supportata al momento in cui scriviamo (Figura 6.37). Una volta scaricato l'installer, basta lanciarlo e seguire le istruzioni del wizard per l'installazione guidata.

Una volta installato, sarà disponibile nel menu dei programmi. Qui è presente anche il *Node.js command prompt* che ci servirà per l'installazione di Node-RED.

Download per Windows (x64)

14.17.1 LTS Consigliata	16.4.0 Corrente Ultime funzionalità
Altri download Changelog Documentazione API	Altri download Changelog Documentazione API

Dai un'occhiata alla [tabella di marcia LTS](#).

Figura 6.37 Versione consigliata LTS per Windows 64 di Node.js.

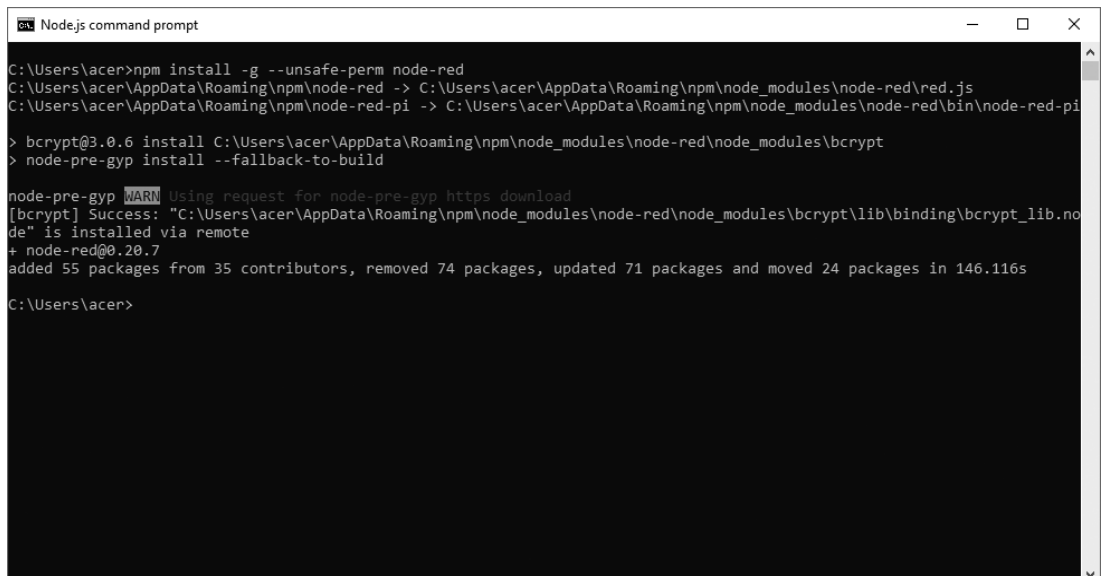
Installazione di Node-RED con npm

Per installare Node-RED si può utilizzare il comando **npm** fornito con Node.js.

Per fare questo bisogna aprire il **Node.js command prompt** come illustrato in Figura 6.38 e digitare:

npm install -g --unsafe-perm node-red

Questo comando installerà Node-RED come un modulo globale insieme alle sue dipendenze.



```
Node.js command prompt
C:\Users\acer>npm install -g --unsafe-perm node-red
C:\Users\acer\AppData\Roaming\npm\node-red -> C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\red.js
C:\Users\acer\AppData\Roaming\npm\node-red -> C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\bin\node-red-pi

> bcrypt@3.0.6 install C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt
> node-pre-gyp install --fallback-to-build

node-pre-gyp WARN Using request for node-pre-gyp https download
[bcrypt] Success: "C:\Users\acer\AppData\Roaming\npm\node_modules\node-red\node_modules\bcrypt\lib\binding\bcrypt_lib.no
de" is installed via remote
+ node-red@0.20.7
added 55 packages from 35 contributors, removed 74 packages, updated 71 packages and moved 24 packages in 146.116s

C:\Users\acer>
```

Figura 6.38 Finestra del Node.js command prompt.

Si può verificare che l'installazione ha avuto esito positivo se la fine dell'output del comando è simile a questo:

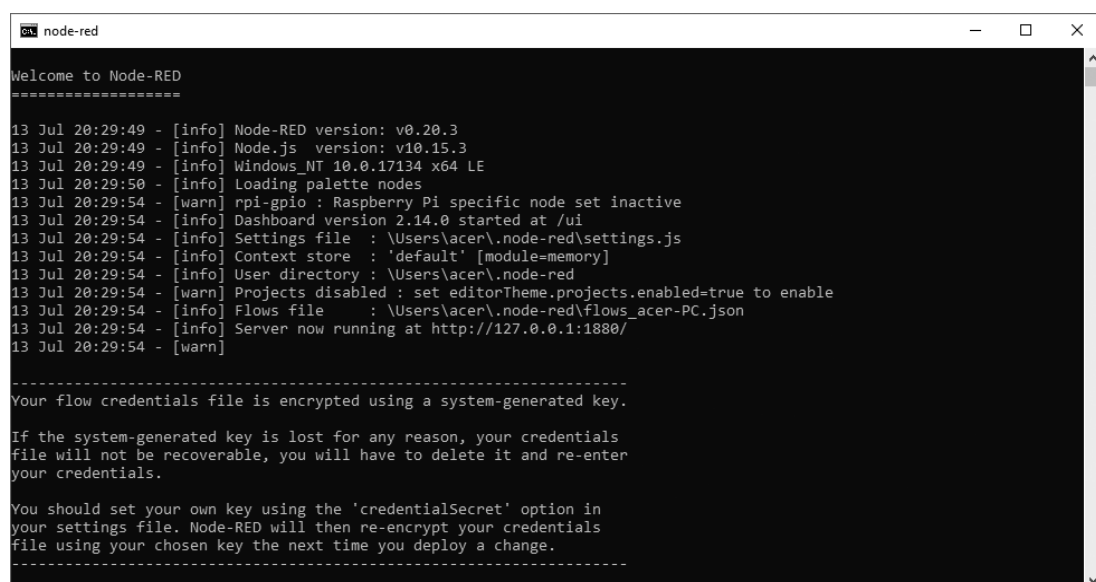
+ node-red@0.20.6

added 55 packages from 25 contributors and audited 1493 packages in 16.606s
found 0 vulnerabilities

Eseguire Node-RED

Una volta installato come modulo globale, si può usare il comando `node-red` per avviare Node-RED nel terminale (Figura 6.39).

node-red



```
node-red

Welcome to Node-RED
=====
13 Jul 20:29:49 - [info] Node-RED version: v0.20.3
13 Jul 20:29:49 - [info] Node.js version: v10.15.3
13 Jul 20:29:49 - [info] Windows_NT 10.0.17134 x64 LE
13 Jul 20:29:50 - [info] Loading palette nodes
13 Jul 20:29:54 - [warn] rpi-gpio : Raspberry Pi specific node set inactive
13 Jul 20:29:54 - [info] Dashboard version 2.14.0 started at /ui
13 Jul 20:29:54 - [info] Settings file : \Users\acer\.node-red\settings.js
13 Jul 20:29:54 - [info] Context store : 'default' [module=memory]
13 Jul 20:29:54 - [info] User directory : \Users\acer\.node-red
13 Jul 20:29:54 - [warn] Projects disabled : set editorTheme.projects.enabled=true to enable
13 Jul 20:29:54 - [info] Flows file : \Users\acer\.node-red\flows_acer-PC.json
13 Jul 20:29:54 - [info] Server now running at http://127.0.0.1:1880/
13 Jul 20:29:54 - [warn]

-----
Your flow credentials file is encrypted using a system-generated key.

If the system-generated key is lost for any reason, your credentials
file will not be recoverable, you will have to delete it and re-enter
your credentials.

You should set your own key using the 'credentialSecret' option in
your settings file. Node-RED will then re-encrypt your credentials
file using your chosen key the next time you deploy a change.
-----
```

Figura 6.39 Avvio di Node-RED.

È possibile utilizzare Ctrl-C chiudere la finestra del terminale e interrompere l'esecuzione di Node-RED.

Per accedere localmente all'editor Node-RED basta digitare sul browser questo indirizzo:

http://localhost:1880 oppure **http://127.0.0.1:1880**.

Apparirà una finestra simile alla Figura 6.40. Si fa notare che per l'accesso remoto da Internet sarà necessario impostare un *port forwarding* sul router di casa e accedere tramite il DNS prescelto.

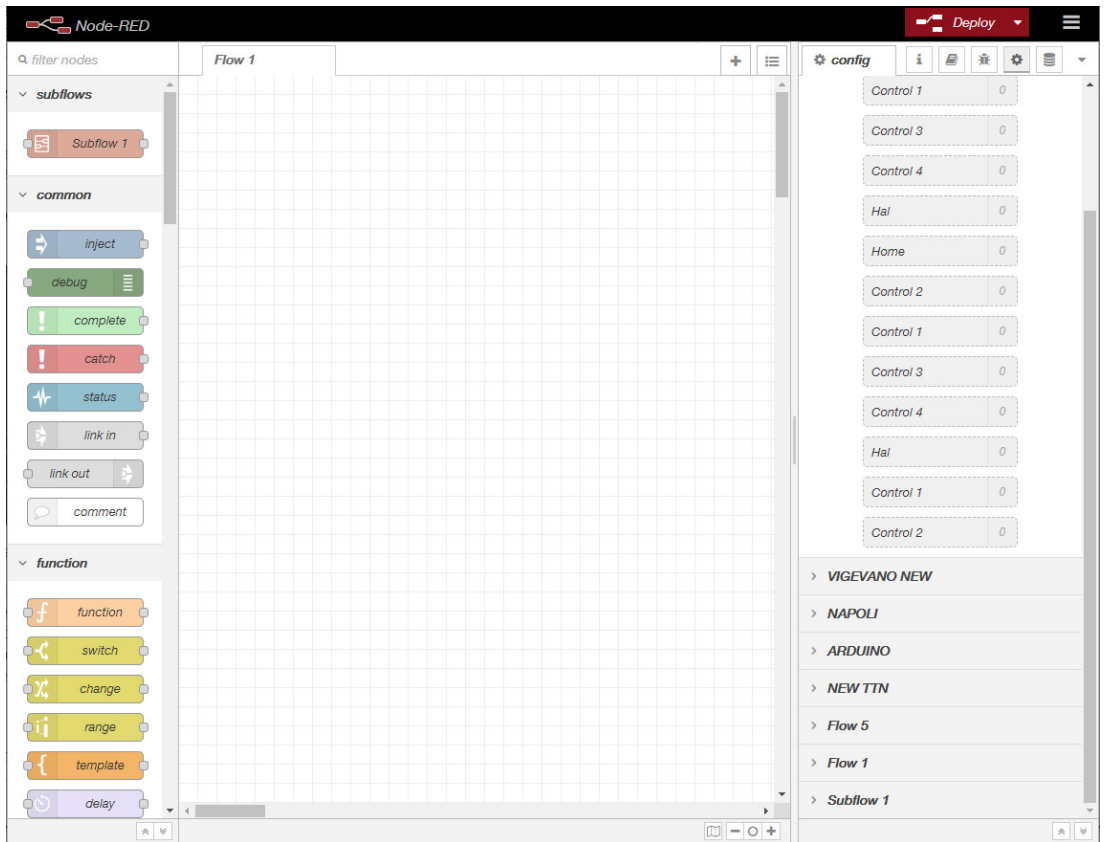


Figura 6.40 L'editor di Node-RED.

Creare il primo flow con Node-RED

Innanzitutto diciamo che Node-RED si basa sui cosiddetti **flow**, traducibile come "flussi", ma che lasceremo in inglese.

Un flow è un insieme di oggetti grafici programmabili e collegabili fra loro con semplici drag and drop.

Per capire i flow che andremo a usare per visualizzare da TTN i dati di temperatura e umidità (o qualsiasi altra cosa), è necessario introdurre l'argomento con questo breve tutorial.

Questo tutorial introduce l'editor di Node-RED per creare un flow che illustra l'uso dei nodi *Inject*, *Debug* e *Function*.

- 1. Accesso all'editor:** con Node-RED in esecuzione sul terminale, aprire l'editor in un browser web. Se si utilizza un browser sullo stesso computer su cui è in esecuzione Node-RED, è pos-

sibile accedervi con l'URL: `http://localhost:1880`. Se si utilizza un browser su un altro computer, sarà necessario utilizzare l'indirizzo IP del computer che esegue Node-RED: `http://<indirizzo-ip>:1880`.

- 2. Aggiungere un nodo Inject:** il nodo *Inject* consente di iniettare messaggi in un flow, facendo clic sul pulsante sul nodo o impostando un intervallo di tempo tra le iniezioni. Trascinarne uno nell'area di lavoro dalla tavolozza. Selezionare il nodo Inject appena aggiunto e fare doppio clic per visualizzare le sue proprietà e una descrizione di cosa fa nel riquadro della barra laterale di informazioni. Come "iniezione" predefinita lasceremo *"timestamp"* (data e ora) anche se si può iniettare praticamente qualsiasi cosa (Figura 6.41).

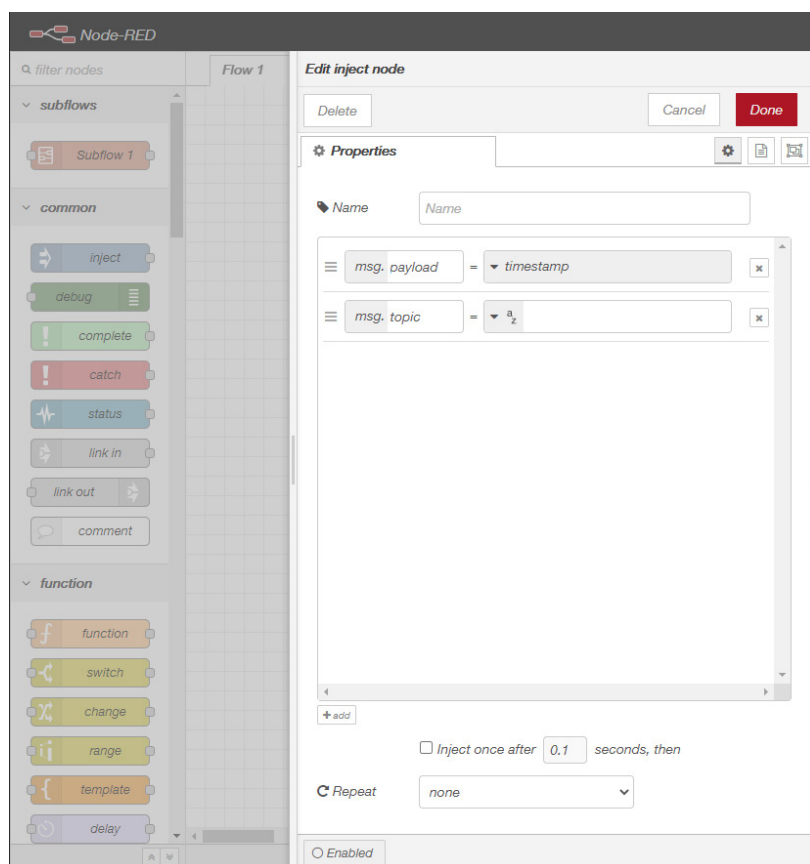


Figura 6.41 Il nodo inject timestamp.

- 3. Aggiungere un nodo Debug:** il nodo *Debug* permette la visualizzazione di qualsiasi messaggio nella barra laterale di

debug. Per impostazione predefinita, mostra solo il payload del messaggio, ma è possibile visualizzare l'intero oggetto del messaggio.

- 4. Collegare i due nodi insieme:** collegare i nodi *Inject* e *Debug* insieme trascinando tra la porta di output di uno e la porta di input dell'altro (Figura 6.42).

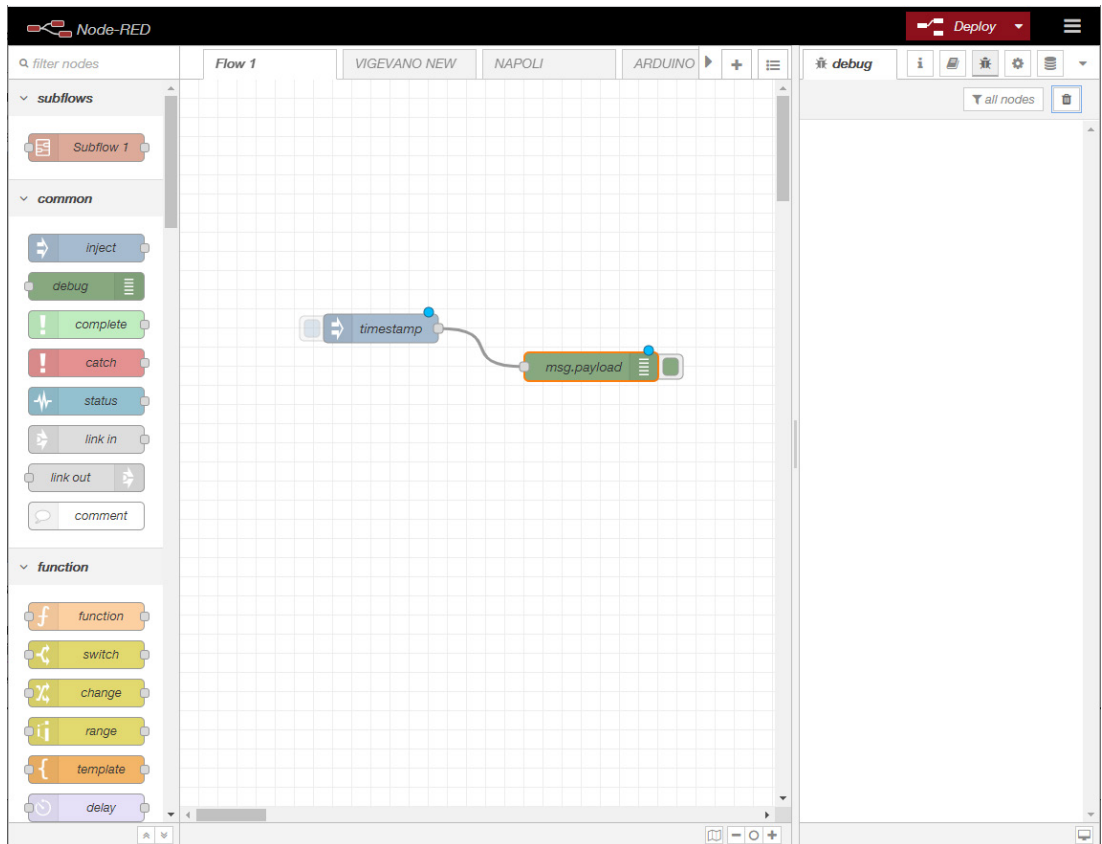


Figura 6.42 Il nodo *inject* e *debug* collegati insieme.

- 5. Distribuire il flow:** a questo punto, i nodi esistono solo nell'editor e per funzionare devono essere "distribuiti" (*Deploy*) sul server. Fare clic sul pulsante *Deploy*.
- 6. Visualizzare i dati di debug:** con la scheda *Debug* della barra laterale selezionata, fare clic sul pulsante *Inject* (timestamp). Dovrebbero apparire i numeri nella barra laterale. Per impostazione predefinita, il nodo *Inject* utilizza il numero di millisecondi dal 1° gennaio 1970 come payload, tradizionalmente in informatica, detto **epoch** o **epoca**.

Per esempio: 1563038172646 corrisponde alla data 2019-07-13 e orario 10:16:13-07:00. La funzione di conversione viene effettuata da una semplice funzione in JavaScript.

7. Aggiungere un nodo Function: il nodo *Function* consente di passare ogni messaggio attraverso una funzione JavaScript. Eliminare il filo esistente selezionandolo e premendo “cancel-la” sulla tastiera. Collegare il nodo *Function* tra i nodi *Inject* e *Debug* (figura 6.43). Fare doppio clic sul nodo *Function* per visualizzare la finestra di modifica. Copiare il seguente codice nel campo funzioni:

```
var date = new Date (msg.payload);  
msg.payload = date.toString ();  
return msg;
```

8. La prima riga crea un oggetto *Date* dal payload.
La seconda cambia il payload in stringa formattata.
La terza restituisce il messaggio.

9. Fare clic su *Done* per chiudere la finestra di modifica del nodo *Function* e sul pulsante *Deploy* per distribuirla.

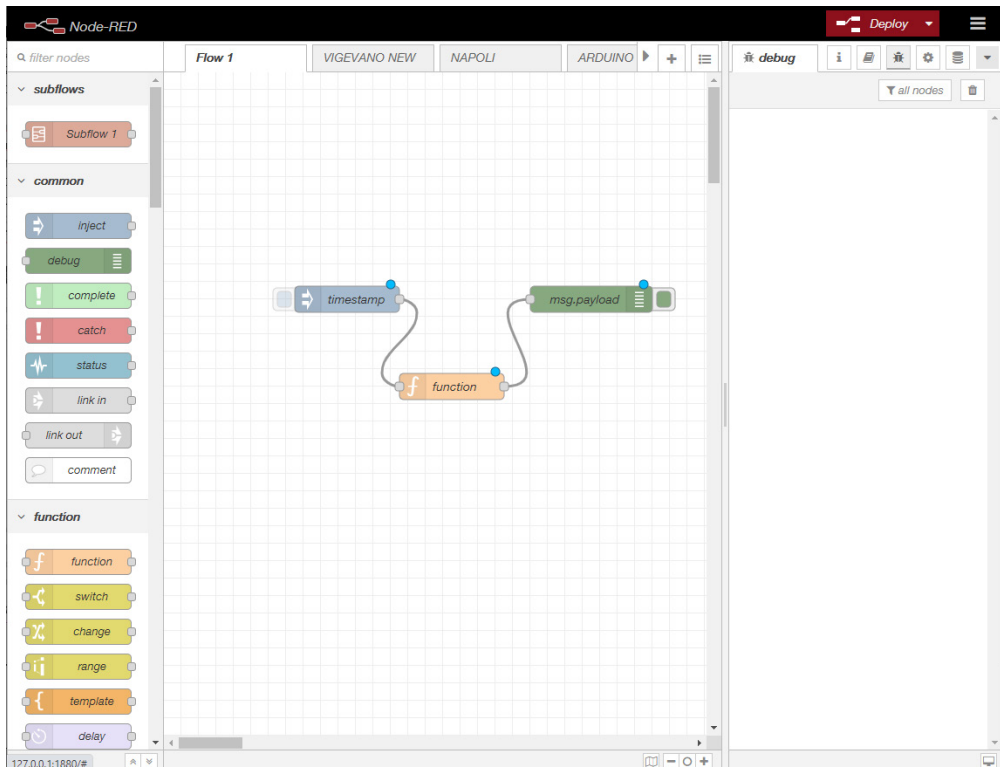


Figura 6.43 Il nodo *Function* interposto fra il nodo *Inject* e *Debug*.

- 10. Visualizzare la data e l'ora:** quando si preme il pulsante *Inject*, i messaggi nella barra laterale verranno formattati con un timestamp leggibile.

Questo flow dimostra il concetto di base della creazione di un flow. Mostra come il nodo *Inject* può essere utilizzato per attivare manualmente un flow e come il nodo *Debug* possa visualizzare i messaggi nella barra laterale. Mostra anche come il nodo *Function* può essere utilizzato per scrivere funzioni in JavaScript di qualsiasi tipo.

Codice sorgente del flow di esempio

Il flow creato in questo tutorial è rappresentato dal seguente file in formato json:

```
[{"id": "58ffae9d.a7005", "type": "debug", "name": "", "active": true, "complete": false, "x": 640, "y": 200, "wires": []}, {"id": "17626462.e89d9c", "type": "inject", "name": "", "topic": "", "payload": "", "repeat": "", "once": false, "x": 240, "y": 200, "wires": [{"id": "2921667d.d6de9a"}]}, {"id": "2921667d.d6de9a", "type": "function", "name": "Format timestamp", "func": "// Create a Date object from the payload\nvar date = new Date(msg.payload);\n// Change the payload to be a formatted Date string\nmsg.payload = date.toString();\n// Return the message so it can be sent on\nreturn msg;", "outputs": 1, "x": 440, "y": 200, "wires": [{"id": "58ffae9d.a7005"}]}
```

Per esportare il flow dall'editor, basta copiarlo negli appunti e incollarlo nella finestra di un altro flow. Per fare questo, usare il menu *Export* dalla barra degli strumenti.

Per importare un flow nell'editor, basta copiarlo dagli appunti e incollarlo nella finestra di un flow, usando il menu *Import* dalla barra degli strumenti (Figura 6.44).

NOTA IMPORTANTE

Avendo provveduto a fornire nelle risorse del libro il file json del flow che andremo a usare per TTN, non verranno spiegati tutti i passi per costruirlo. Ci limiteremo ad usarlo fornendo comunque i commenti necessari alla loro comprensione.

Per gli approfondimenti sulla creazione dei flow e l'uso in generale di Node-RED fare riferimento alle guide presenti sul sito ufficiale <https://nodered.org>.

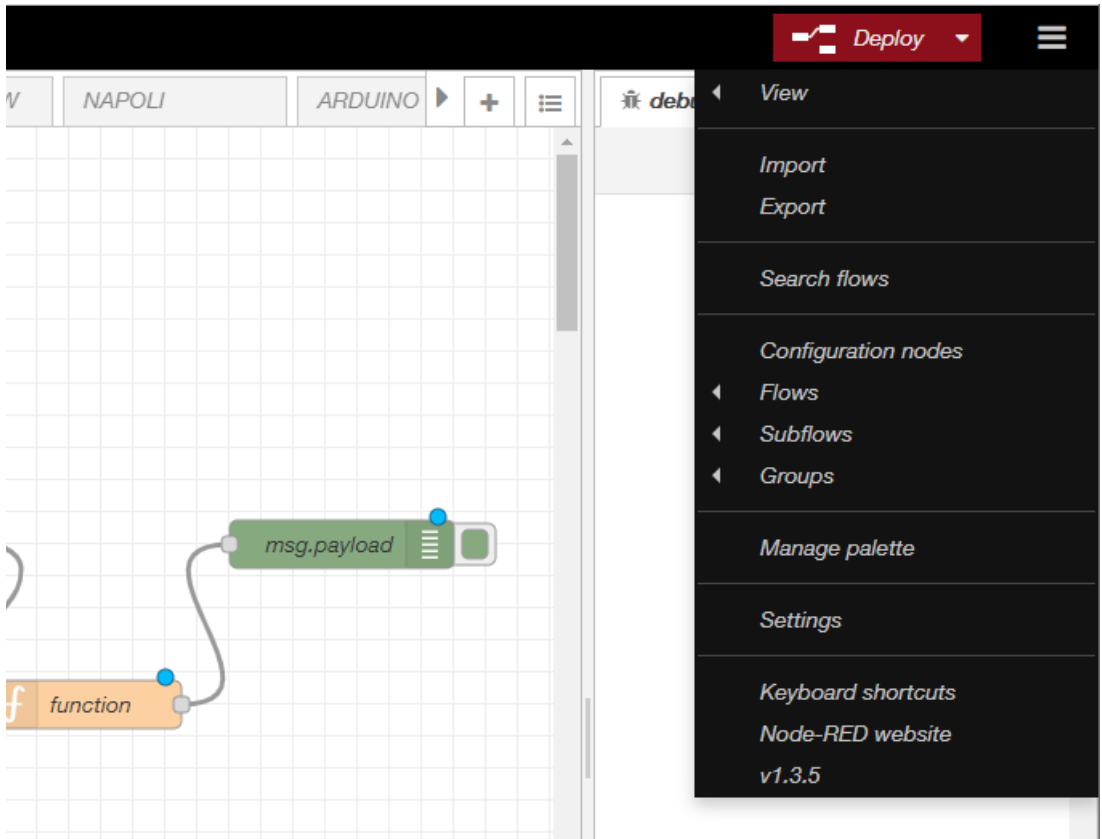


Figura 6.44 La barra degli strumenti Import ed Export di Node-RED.

Installare la dashboard in Node-RED

La grande community open source di Node-RED ci permette di usufruire gratuitamente di un cospicuo numero di strumenti aggiuntivi, chiamati appunto "nodi". Uno di questi è senza dubbio la *dashboard* o, in italiano, tavolozza di strumenti (Figura 6.45). La dashboard di Node-RED, in pratica, è una *user interface* (ui) che consente di aggiungere una serie di elementi grafici ai nostri flow per visualizzare sul browser oggetti di vario tipo come quelli elencati qui sotto:

- **ui_button**
- **ui_dropdown**

- `ui_slider`
- `ui_numeric`
- `ui_text_input`
- `ui_date_picker`
- `ui_colour_picker`
- `ui_form`
- `ui_text`
- `ui_gauge`
- `ui_chart`
- `ui_audio`
- `ui_toast`
- `ui_ui_control`
- `ui_template`

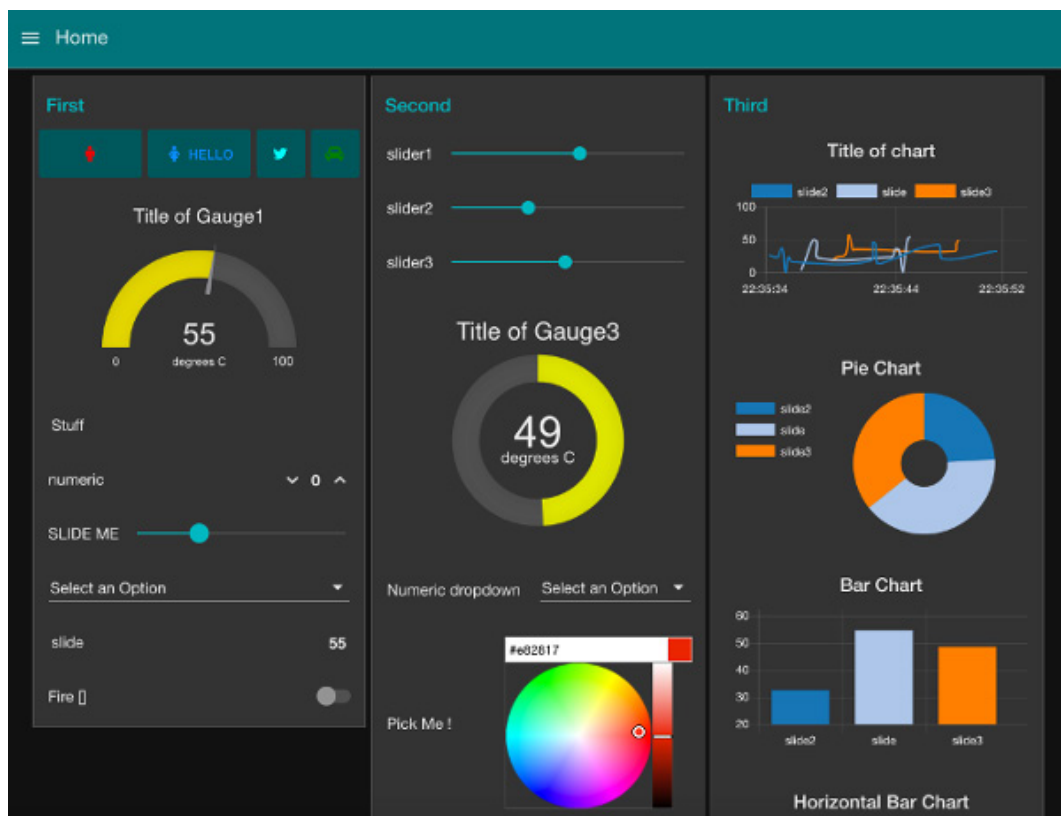


Figura 6.45 I nodi della dashboard di Node-RED.

Per approfondire l'uso della dashboard, rimandiamo al link seguente: <https://flows.nodered.org/node/node-red-dashboard>.

Per l'installazione della dashboard, indispensabile per la visualizzazione dei flow del libro, procedere in questo modo.

Aprire il command prompt nella directory Windows:
`cd C:\Users\nome_utente\.node-red`

... oppure, il command prompt nella directory Mac/Linux:
`cd ~/.node-red`

Digitare sul terminale:
`npm i node-red-dashboard`

Dopo l'installazione, riavviando Node-RED sempre dal Node.js command prompt, dovrebbe essere disponibile la sezione dashboard con tutti i nodi UI nel pannello laterale sinistro.

Notare che per aprire localmente l'interfaccia grafica della dashboard bisogna digitare sul browser l'indirizzo:

`http://localhost:1880/ui`

ATTENZIONE ttn-node

Fra i molti contributi della community di Node-RED c'è anche questo nodo che non è più supportato con lo stack V3 di TTN. Pertanto è perfettamente inutile installarlo.

Importare i flow forniti con il libro

Come già detto abbiamo provveduto a fornire nelle risorse del libro il file json del flow che andremo a usare.

Una volta decompresso il file risorse_libro_lora.zip, nella cartella NODE-RED ci sarà il file **NEW_TTN.json**. Basterà importarlo nell'editor di Node-RED in questo modo.

1. Avviare il server Node-RED
2. Aprire l'editor di Node-RED dal browser.
 1. Dalla barra del menu in alto a destra selezionare *Import*.
 2. Si aprirà una finestra "*Import nodes*" simile alla Figura 6.46.
 3. Fare clic su "*select a file to import*".
 4. Individuare il file NEW_TTN.json sul computer.
 5. Fare clic su "*Import to current flow*".

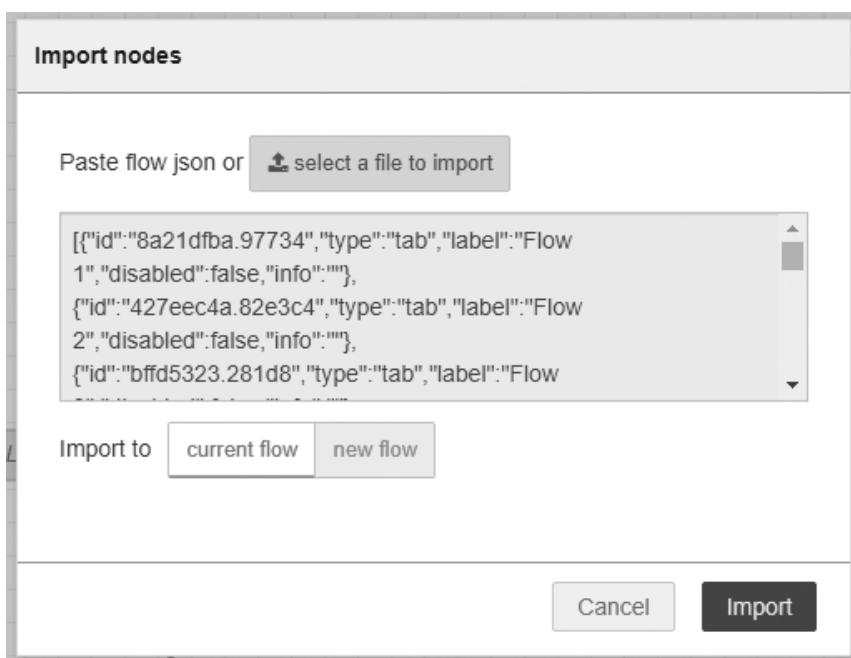


Figura 6.46 La finestra *Import nodes* di Node-RED.

Dopo l'importazione è disponibile un flow:

- **NEW TTN:** connessione a TTN con un end device

Flow NEW TTN: connessione a TTN con un end device

Per vedere in funzione questo flow si presuppone che siano state portate a termine con successo le seguenti operazioni:

1. L'end device è acceso e comunica con il gateway.
2. Il gateway riceve correttamente i dati dall'end device e li invia al dispositivo **device-001** dell'applicazione **ttn-v3-pier-2021**.
3. Nella pagina **Live data** si vedono arrivare i dati di temperatura e umidità.

Integrazione MQTT

Con la nuova versione V3 di TTN è necessario attivare l'integrazione MQTT per poter comunicare con Node-RED.

Dalla sezione Integrations aprire MQTT. Si aprirà una pagina (Figura 6.47) in cui si possono vedere i parametri da usare per la connessione MQTT.

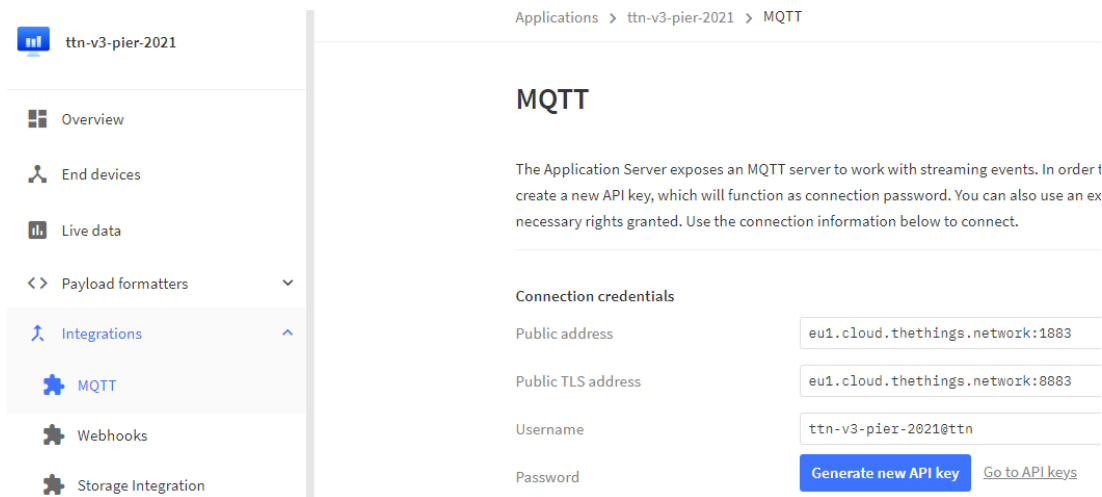


Figura 6.47 La pagina MQTT della sezione Integrations.

Da questa pagina noi useremo:

- **Public address:** eu1.cloud.thethings.network:1883
- **User name:** ttn-v3-pier-2021@ttn (notare che il nome utente è il nome dato all'applicazione più il suffisso @ttn)
- **Password:** ottenuta con Generate new API key.

Facendo clic su Generate new API key si aprirà una pagina per generare la chiave di accesso e facendo clic su Add API key si aprirà una ulteriore finestra in cui si potrà vedere per una sola volta la chiave generata (Figura 6.48). Per comodità, si consiglia di copiare la chiave e incollarla provvisoriamente su un foglio di notepad. Ci servirà più avanti.

Add API key

Name

Rights*

Granted rights



- ✓ All application rights

Please copy newly created API key

You won't be able to view the key afterward

Your API key has been created successfully. Note: After closing this window, the value of the key secret will not be accessible anymore. Make sure to copy and store it in a safe place now.

API key

☐ Write uplink application traffic

Create API key

✓ I have copied the key

Figura 6.48 La finestra della creazione della API key per l'accesso a TTN.

Tornare su Node-RED e aprire l'editor del nodo MQTT TTN UPLINK. Aprire il tab Security, incollare il nome utente (per noi è ttn-v3-pier-2021@ttn) e incollare la chiave che è stata copiata nel notepad (Figura 6.49).

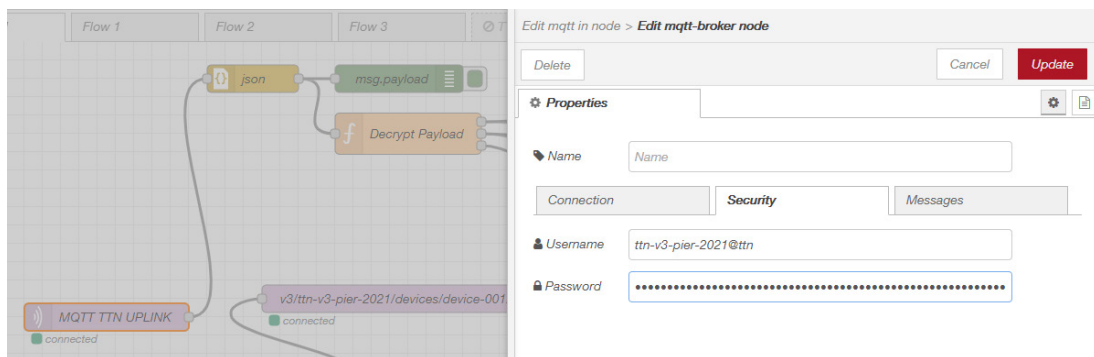


Figura 6.49 La finestra Edit MQTT node di Node-RED.

Una volta chiuso l'editor e fatto il deploy, dovremmo vedere il pallino verde *connected* sotto il nodo MQTT TTN UPLINK, come illustrato nella Figura 6.50.

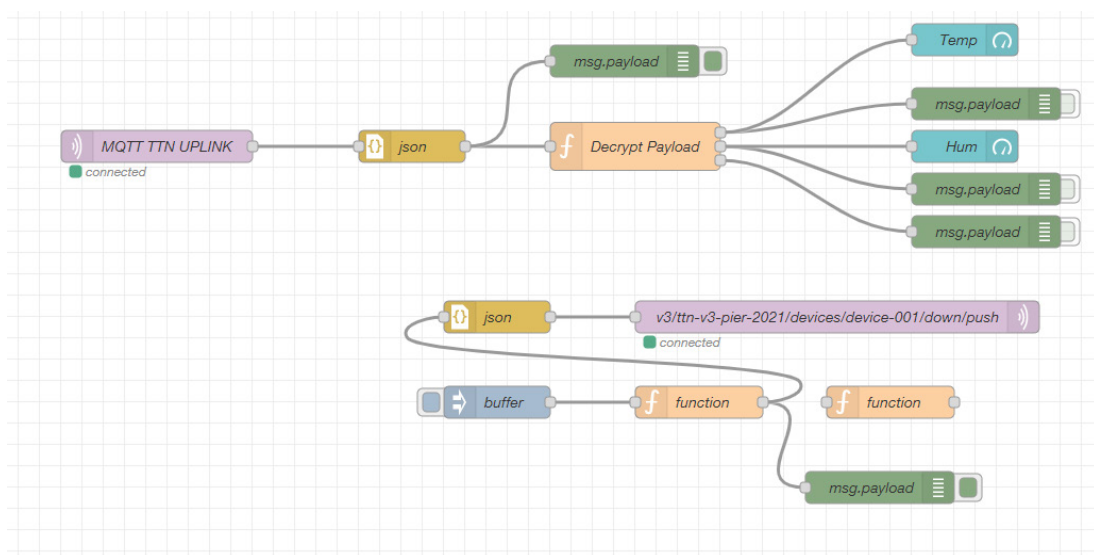


Figura 6.50 La finestra Edit MQTT node di Node-RED.

Nella finestra di debug di Node-RED dovremmo vedere i dati di temperatura [23] e umidità [68]. Si fa notare che, per motivi pratici, manterremo la coerenza con questi due dati (23 e 68) per tutto il capitolo. Ovviamente, sono puramente indicativi e nella

realtà cambieranno costantemente.

■ **Attenzione!**

Il file in formato json che è stato fornito come esempio contiene i nomi delle applicazioni e dei device usati dall'autore a scopo solo didattico. Quindi, il nodo *MQTT TTN UPLINK* funzionerà solo se si cambiano il nome utente e la password del nodo MQTT.

Nodo della funzione Extract Data

Abbiamo chiamato così questo nodo *Function* perché ci serve proprio una funzione per estrarre i dati dal payload inviato da TTN. Il payload, lo ricordiamo, è composto da questi dati:

54 17 48 44 30 30 30 30 che, convertiti, sono:

```
"d1": 84,  
"d2": 23,  
"d3": 72,  
"d4": 68,  
"d5": 48,  
"d6": 48,  
"d7": 48,  
"d8": 48
```

Facendo doppio clic sul nodo si aprirà la finestra con la nostra funzione JavaScript (Figura 6.51):

```
var obj =  
{  
  0:msg.payload.uplink_message.decoded_payload.bytes[0],  
  1:msg.payload.uplink_message.decoded_payload.bytes[1],  
}  
  
var res_temp = obj[0]  
var res_hum = obj[1]  
  
var msg1 = { payload:res_temp };  
var msg2 = { payload:res_hum };  
return [msg1,msg2]
```

Senza entrare nei particolari di programmazione JavaScript, daremo solo qualche indicazione del significato delle variabili e delle funzioni.

- Dovendo estrarre solo i primi 4 byte del payload, cioè i dati relativi a temperatura [23] e umidità [68], l'oggetto *obj* crea quattro variabili.
- Le variabili *msg1* e *msg2* raccolgono rispettivamente i parametri del *payload:res_temp* e *payload:res_hum* che vengono ritornati dalla funzione con *return [msg1, msg2]*.
- Nella stessa figura possiamo vedere che sono stati impostati due *output*. Significa che la funzione ha due uscite separate a cui vengono inviati rispettivamente il valore di *msg1*, cioè [23], e il valore di *msg2*, cioè [68].
- A queste due uscite possiamo collegare un termometro e un igrometro, come vedremo fra poco.

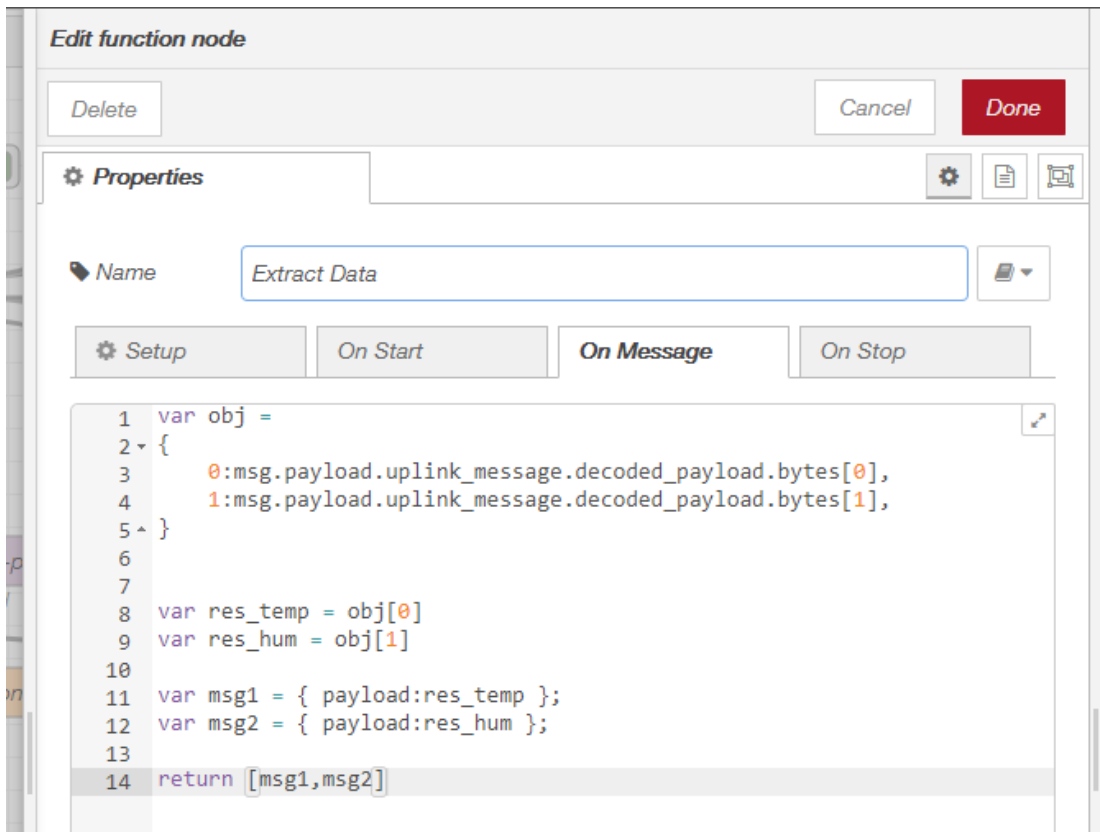


Figura 6.51 La finestra Edit MQTT node di Node-RED.

Nodo termometro

La dashboard di Node-RED offre una serie di misuratori molto belli. Per visualizzare la temperatura abbiamo pensato a un nodo di tipo *Gauge*, cioè un misuratore ad ago.

Figura 6.52 La finestra *Edit gauge node (Gauge)* di Node-RED.

Nodo igrometro

Per visualizzare l'umidità abbiamo pensato a un nodo di tipo *Level*, cioè un misuratore del livello dell'acqua (Figura 6.53).

Edit gauge node

Buttons: Delete, Cancel, Done

Properties

- Group: [NEW TTN] Device 001
- Size: 5 x 5
- Type: Level
- Label: Hum
- Value format: {{value}}
- Units: %
- Range: min 0 max 100
- Name: Hum

Figura 6.53 La finestra *Edit gauge node (Level)* di Node-RED.

La visualizzazione sul browser

Aprendo localhost:1880/ui, tutto il nostro lavoro sarà visibile in tempo reale sul browser, come illustrato in Figura 6.54.

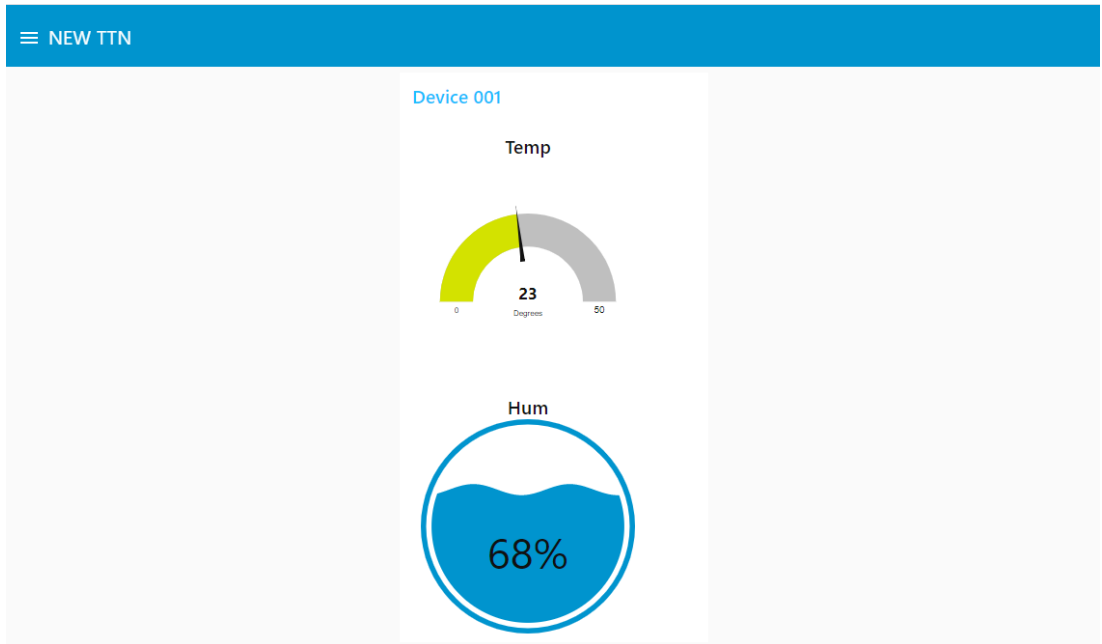


Figura 6.54 La finestra del browser con i Gauge di temperatura e umidità.

Come proseguire?

Abbiamo dotato l'end device con un sensore DHT11 e un LED e siamo riusciti a visualizzare i dati dell'applicazione TTN con Node-RED. Ma non finisce qui...

Il prossimo capitolo è interamente dedicato al gateway multicanale fatto con il concentratore iC880A e Raspberry Pi. Una volta assemblato il nostro gateway, lo potremo utilizzare esattamente come il Dragino HAT ma con la differenza che avremo a disposizione tutti gli 8 canali LoRaWAN e pertanto una maggiore efficienza. Buona continuazione!