Capire e usare LoRa®

4

di PIER CALDERAN

Vediamo come connettere il nostro End Device a The Things Network e come visualizzare i dati dei sensori sfruttando la piattaforma Node-RED e la rete wireless LoRaWAN.

Abbiamo imparato, nella scorsa puntata, come TTN visualizza i dati del gateway in modo per così dire "grezzo". Non essendo implementata nessuna conversione, tutto viene riportato sotto forma di numeri esadecimali, abbastanza scomodi da interpretare.
Per rendere questi dati, per così dire, *human-readable*, ovvero comprensibili all'occhio umano, abbiamo bisogno di adattarli per poterli usare con un'interfaccia grafica.
A questo scopo, abbiamo scelto Node-RED, uno strumento di facile programmazione in grado di mettere in connessione dispositivi hardware, API e servizi online.
Il suo editor è basato su un browser che, grazie ai suoi numerosi moduli grafici, semplifica il collegamento dei flussi fra hardware e software. Questi moduli grafici utilizzano un'ampia gamma di nodi nella tavolozza dell'editor che possono essere distribuiti con un solo clic.
Tenete in considerazione il fatto che in questa sede verranno usati solo gli elementi essenziali di Node-RED finalizzati al nostro scopo; consigliamo a chi volesse ulteriori informazioni, disporre di tutorial e di altro materiale, di visitare il sito ufficiale *https://nodered.org*.

INSTALLAZIONE DI NODE-RED

Prima di installare Node-RED localmente, avrete bisogno di una versione di Node.js. Per l'installazione di Node.js si vedano le istruzioni per tutti i sistemi operativi supportati sul sito ufficiale *https://nodejs.org*.
Dopo aver installato Node.js è possibile installare Node-RED tramite il comando *npm* fornito con node.js.
Digitare il comando seguente da una finestra del terminale Linux/OS X o del command prompt di Windows:

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

Se si utilizza Windows, non impartite il comando con *sudo* ma con il command prompt eseguito in modalità amministratore.

Fig. 1 - Esempio di interfaccia grafica del modulo dashboard di Node-RED.

Tale comando installerà Node-RED come modulo globale insieme alle sue dipendenze. Ulteriori informazioni sull'installazione di Node-RED su Windows sono disponibili presso il sito Internet ufficiale di Node-RED, cui rimandiamo per gli approfondimenti del caso.
L'installazione sarà riuscita se alla fine della finestra di output del terminale comparirà qualcosa di simile a questo:

```
+ node-red@1.1.0
added 332 packages from 341 contributors in 18.494s
found 0 vulnerabilities
```

Notare che per la piattaforma Raspberry Pi, Node-RED è già installato per impostazione predefinita nel sistema Raspberry Pi OS (ex Raspbian) e quindi si può saltare l'installazione.

INSTALLAZIONE DEL MODULO NODE-RED DASHBOARD

Per la visualizzazione dei dati sotto forma di grafici e di misuratori abbiamo scelto il modulo *dashboard*, che fornisce un ricco pannello di strumenti come quelli che vedete nella **Fig. 1**. Questo modulo fornisce una serie di nodi pronti all'uso che permettono di creare rapidamente una dashboard.
Per installare l'ultima versione stabile, avviare il server Node-RED e utilizzare il menu *Manage palette* e l'opzione *Palette->Install*. Nel pannello che apparirà, digitare nella casella di ricerca il testo "node-red-dashboard" e installare l'ultima versione disponibile.
Per avviare il server Node-RED è sufficiente eseguire il seguente comando dal terminale o dal command prompt:

```
node-red
```

Una volta avviato il server Node-RED, si vedrà qualcosa di simile alla **Fig. 2**. Aprendo un browser internet l'interfaccia di Node-RED sarà disponibile all'indirizzo *http://localhost:1880* oppure *http://127.0.0.1:1880*.
Per l'installazione del modulo dashboard, si può eseguire il seguente comando *npm* dalla directory utente di Node-RED. In genere questa si trova in ~/.node-red in Linux/OS X e C:\Users\User\.node-red in Windows:

```
npm i node-red-dashboard
```

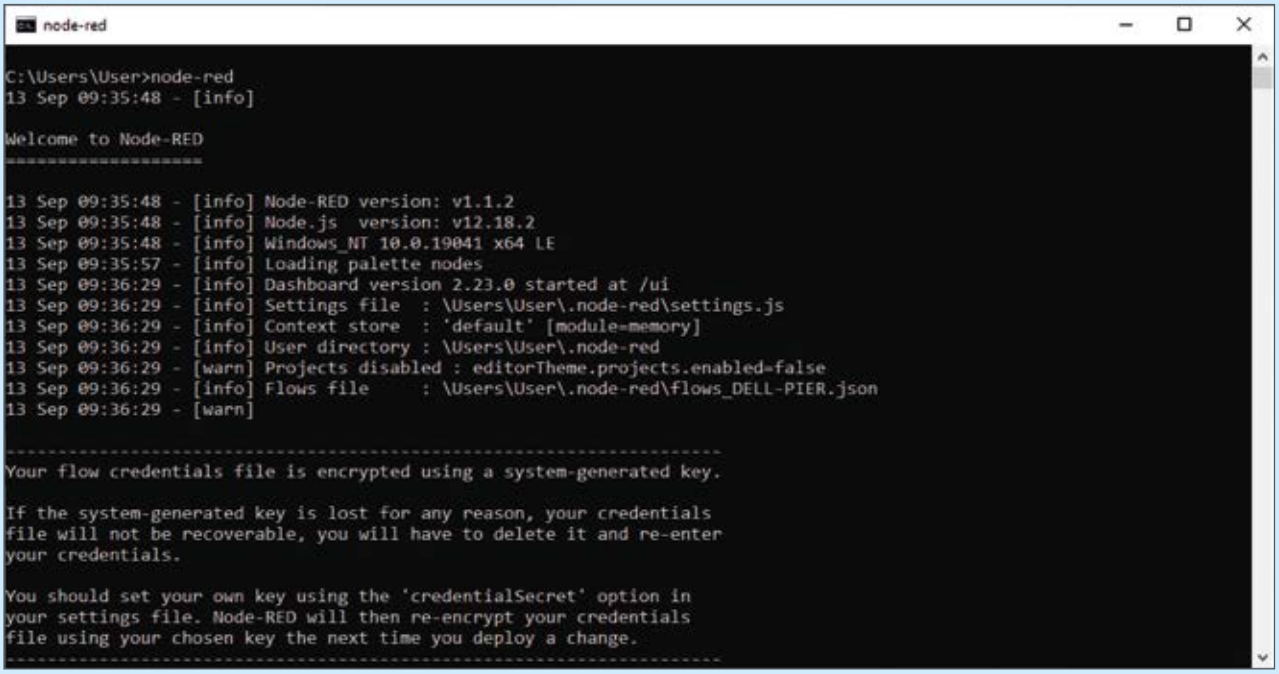


Fig. 2 - Il server Node-RED avviato.

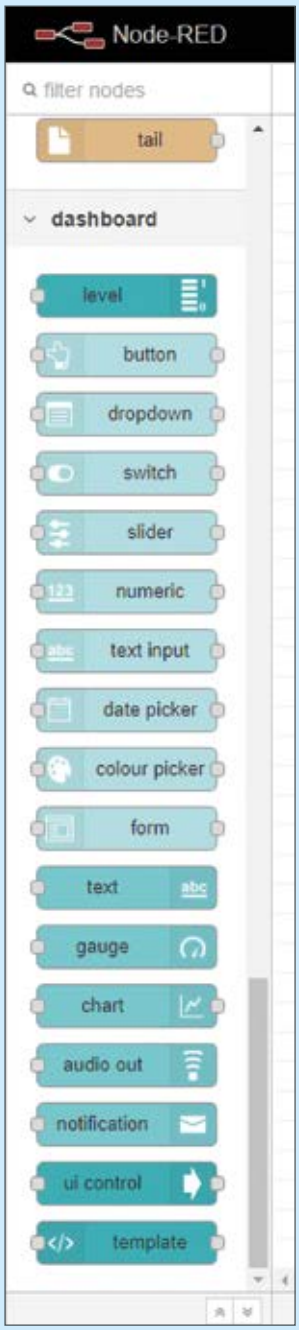
Dopo aver avviato il server Node-RED si dovrebbero vedere i nodi disponibili nella scheda *dashboard* situata nel pannello laterale sinistro dell'interfaccia, come illustrato nella **Fig. 3**.
Ricordiamo che per accedere alla dashboard basta digitare il seguente indirizzo nel browser: *http://localhost:1880/ui* oppure *http://127.0.0.1:1880/ui*
Il layout della dashboard deve essere considerato come una griglia. Ogni layout di Node-RED viene chiamato *Flow*.
Ogni widget del gruppo ha una larghezza per impostazione predefinita di 6 "unità" (un'unità è larga 48 px per impostazione predefinita con uno spazio di 6 px).
Ogni widget nel gruppo ha una larghezza predefinita "auto". Significa che riempirà la larghezza del gruppo in cui si trova, ma si può impostare un numero fisso di unità.
L'algoritmo del layout della dashboard cerca sempre di posizionare i widget il più in alto e a sinistra possibile all'interno del loro contenitore. Questo vale per il modo in cui i gruppi sono posizionati nella pagina e per come i widget sono posizionati in un gruppo.
Per esempio, in un gruppo con larghezza 6 unità, se si aggiungono sei widget, ciascuno con una larghezza di 2 unità, verranno disposti su due righe, cioè tre widget in ciascuna riga. Se si aggiungono due gruppi di larghezza 6 unità, a condizione che la finestra del browser sia sufficientemente ampia, si troveranno uno accanto all'altro. Se si riduce il browser, a un certo punto il secondo gruppo si sposterà sotto il primo, in una colonna.
Si consiglia di utilizzare più gruppi, se possibile, piuttosto che un gruppo grande, in modo che la pagina possa ridimensionarsi dinamicamente quando si usano schermi più piccoli.

I widget della dashboard

Non potendo entrare nel dettaglio della spiegazione di questo modulo, diamo comunque una breve descrizione dei nodi disponibili che si possono aggiungere al Flow:

- **button**: aggiunge un pulsante;
- **dropdown**: aggiunge un menu di selezione a discesa;
- **switch**: aggiunge un interruttore;
- **numeric**: aggiunge un widget di input numerico;
- **text_input**: aggiunge un campo di immissione di testo;
- **date_picker**: aggiunge un widget di selezione della data;
- **colour_picker**: aggiunge un selettore di colori;
- **form**: aggiunge un modulo di immissione;
- **text**: con esso viene visualizzato un campo di testo non modificabile;
- **gauge**: aggiunge un widget di tipo indicatore (ago, bussola, livello, ecc.);
- **chart**: traccia i valori di input su un grafico. Può essere un grafico a linee basato sul tempo, un grafico a barre (verticale o orizzontale) o un grafico a torta;
- **audio out**: riproduce audio o sintesi vocale (TTS) dalla dashboard;
- **notification**: mostra msg.payload come una notifica popup o un messaggio di dialogo OK/Cancel;

Fig. 3
Il pannello dashboard con tutti i nodi disponibili.



- **ui control**: consente il controllo dinamico della dashboard (la funzione predefinita è modificare la scheda attualmente visualizzata);
- **template**: questo widget può contenere qualsiasi direttiva HTML e Angular valida; questo nodo può essere utilizzato per creare un elemento dell'interfaccia utente dinamica che cambia il suo aspetto in base al messaggio di input e può inviare messaggi di ritorno a Node-RED.

INSTALLAZIONE DEL MODULO NODE-RED TTN

Per connettere i widget ai device e alle applicazioni di TTN

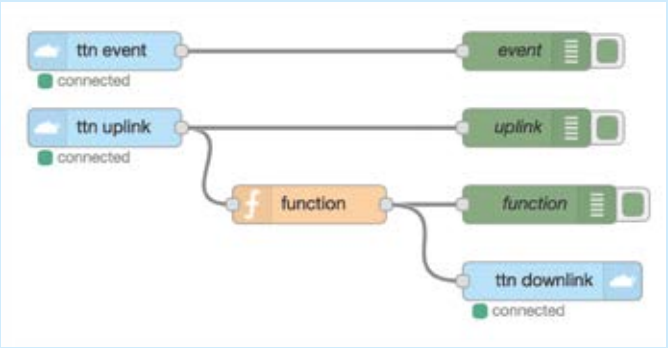


Fig. 4 - Esempio di utilizzo del modulo node-red-contrib-ttn.

dovremo installare anche il modulo *node-red-contrib-ttn* il cui sito ufficiale è *https://www.npmjs.com/package/node-red-contrib-ttn.*
Come spiegato per il modulo dashboard, installare il modulo dall'interno di Node-RED o eseguire il seguente comando dal terminale (nella directory .node-red):

```
npm install node-red-contrib-ttn
```

Un esempio di utilizzo dei nodi ttn è visibile in **Fig. 4**.
Una volta installato il modulo *node-red-contrib-ttn* saranno disponibili i seguenti nodi nella sezione *input* del pannello di Node-RED:

- **ttn event**: questo nodo serve a ricevere eventi dai dispositivi su The Things Network. L'applicazione e l'evento devono essere configurati nel nodo. Il messaggio di output: *dev_id*, ovvero l'ID del dispositivo che ha inviato il messaggio. *payload*, ovvero l'argomento originale.
- **ttn uplink**: questo nodo serve per ricevere messaggi di uplink dai dispositivi su The Things Network. L'applicazione deve essere configurata nel nodo. Il messaggio di output: *dev_id*, ovvero l'ID del dispositivo che ha inviato il messaggio, *payload*, un oggetto di campi o il campo specificato se l'applicazione è configurata con una funzione di payload del decodificatore o un buffer se non lo è. A meno che non sia stato specificato un campo, il messaggio viene esteso con il topic originale.

CREAZIONE DEL NODO TTN

A questo punto basta inserire un nodo *ttn uplink* nel nostro Flow; allo scopo basta trascinare il nodo dal pannello.
Con un doppio clic sul nodo si aprirà l'editor del nodo, come visualizzato nella **Fig. 5**; qui dovete operare come segue.

- Nel campo **App** selezionare un'app esistente o selezionare **Add new ttn app...** e fare clic sull'icona matita.
- Nella nuova finestra che si aprirà (**Fig. 6**) copiare e incollare le seguenti informazioni prese dall'applicazione di The Things Network:
  - nel campo **App ID** copiare l'ID dell'applicazione dalla sezione *Overview* dell'applicazione. Nel nostro caso inseriamo il nome scelto nella scorsa puntata: **futura_devices**.
  - nel campo **Access Key**, incollare la chiave di acces-

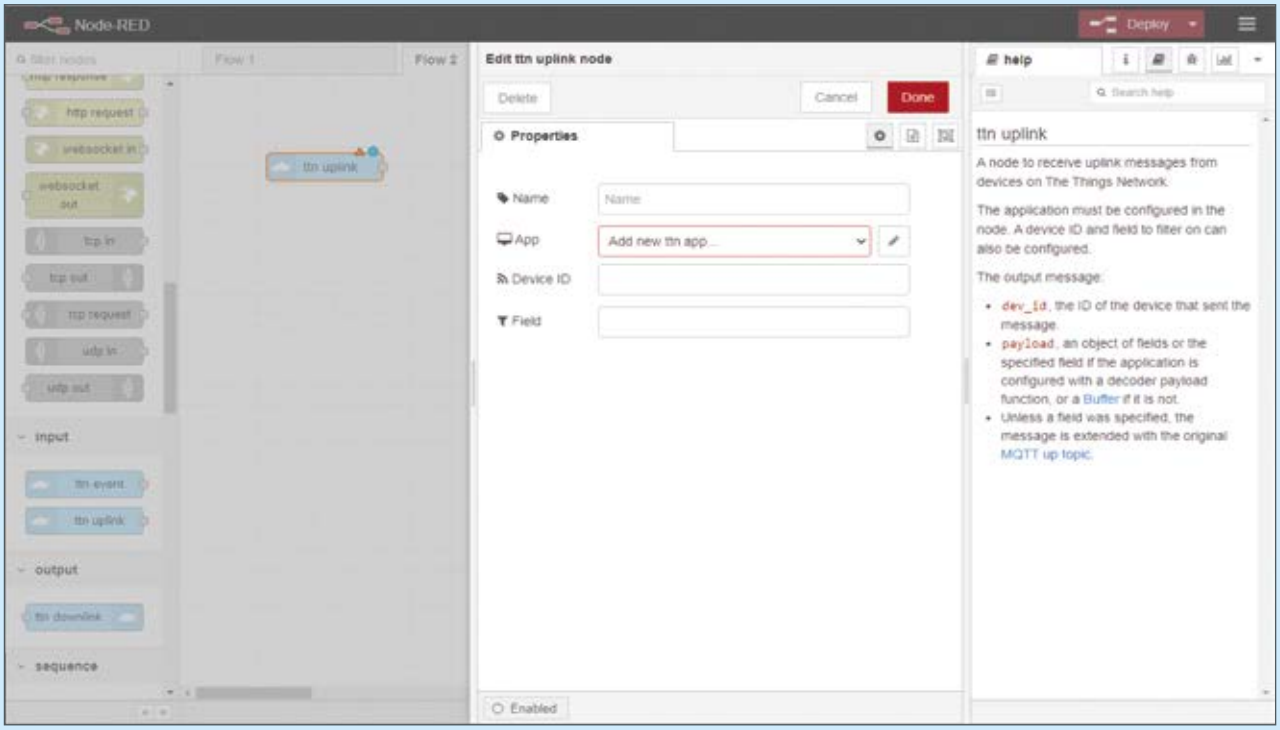


Fig. 5 - Finestra di modifica del nodo ttn uplink.

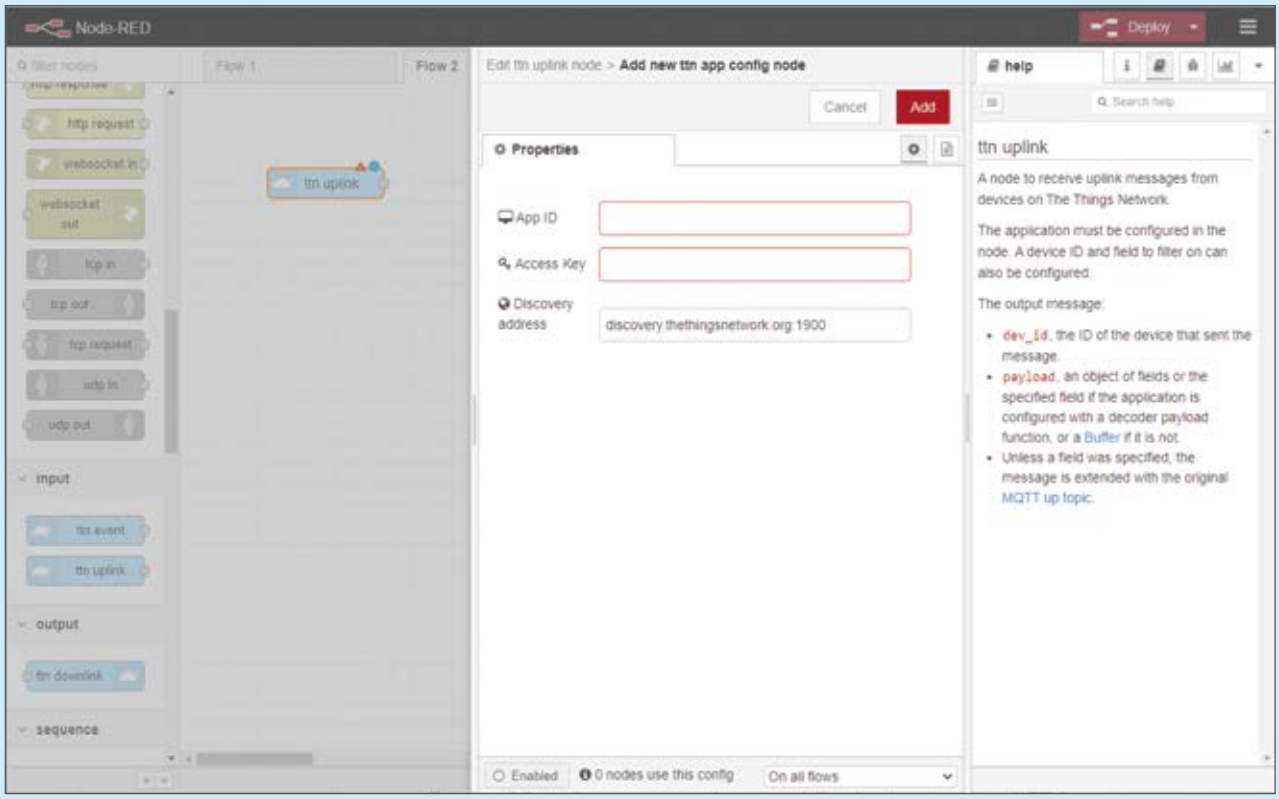


Fig. 6 - Finestra dell'editor del nodo Add new ttn app config node.

so, che si trova nella stessa pagina della sezione *Overview* (la chiave di accesso è qualcosa di simile a questo: **ttn-account-v2.FMmmk63apA4DrdBPPVf2OZSKhfCNQ36klgfNWhsDqc0**).

- Nel campo **Discovery address** è possibile lasciare l'indirizzo di rilevamento predefinito ovvero *discovery.thethingsnetwork.org:1900*.

Fate clic su **Add** per salvare la configurazione dell'applicazione, che adesso è possibile riutilizzare in tutti i nodi TTN di Node-RED. Tornando alla finestra precedente si vedrà l'ID dell'app come illustrato in **Fig. 7**.
Nel campo *name* si può dare opzionalmente un nome al nodo. Nel nostro caso abbiamo scelto **FUTURA TTN UPLINK**.
Facendo clic sul pulsante **Done** e successivamente sul pul-

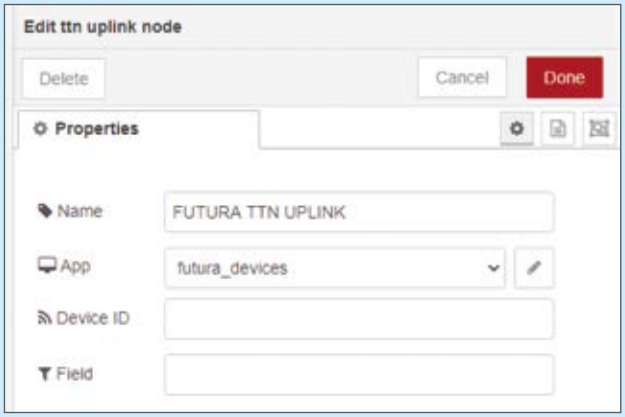


Fig. 7 - Il nodo futura_devices a cui è stato il nome FUTURA TTN UPLINK.

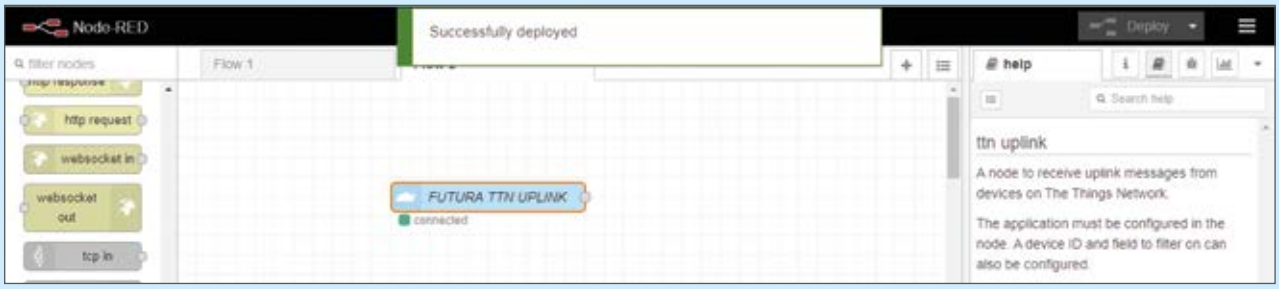


Fig. 8 - Il nodo FUTURA TTN UPLINK connesso.

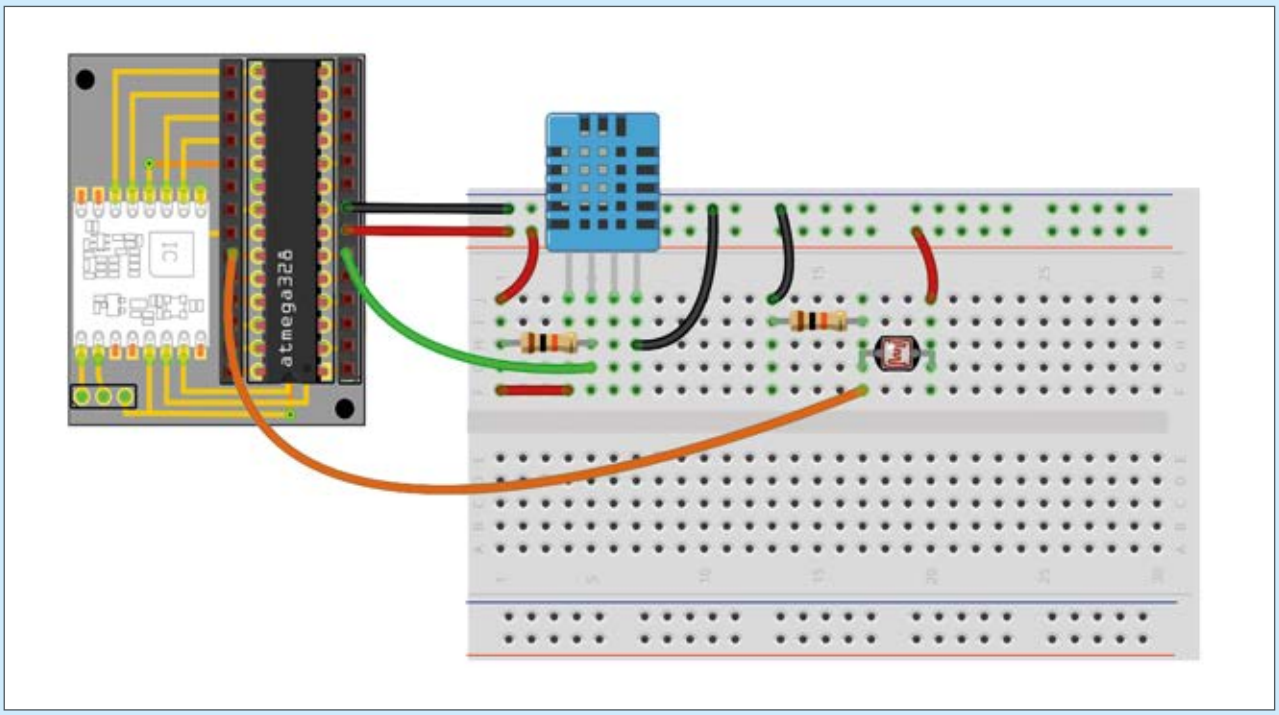


Fig. 9 - Il circuito di prova con un DHT11 e un LDR.

sante **Deploy** di Node-RED si vedrà qualcosa di simile alla **Fig. 8**, in cui si vede il nodo FUTURA TTN UPLINK connesso all'applicazione futura_devices di TTN.
Tenete presente che con questa operazione abbiamo effettuato solo una connessione con l'applicazione TTN e finché non verranno attivati il gateway e il dispositivo finale non si potranno ricevere i dati.

CREARE UN CIRCUITO DI PROVA

Nella puntata scorsa vi abbiamo spiegato come trasmettere e visualizzare su TTN il messaggio "Hello world!". Se tutto è andato bene, oggi possiamo spingerci un po' più là e aggiungere al nostro dispositivo finale dei sensori per la rilevazione, per esempio, di temperatura, umidità e illuminamento ambientale. Per fare questo abbiamo affidato il compito di rilevare la temperatura e l'umidità al diffusissimo modulo DHT11 e la luminosità a un comune fotoresistore (LDR). Il semplice circuito è visibile in **Fig. 9**.
Come si può vedere dallo schema, il pin dei dati del DHT11 è collegato al pin digitale 4 e l'uscita del fotoresistore all'ingresso analogico A0. I resistori di pull-up sono entrambi da 10 kΩ.

CREARE UN PROTOCOLLO DI COMUNICAZIONE

Il codice che abbiamo scritto per la lettura e l'invio dei dati dei sensori si basa sullo stesso esempio usato per il messaggio di "Hello world!" che abbiamo usato nella scorsa puntata. Abbiamo aggiunto solamente qualche riga per creare un piccolo protocollo di comunicazione per TTN.

Per semplificare la spiegazione, nel **Listato 1** mostriamo solo il codice aggiunto e i relativi commenti.
Il nostro protocollo per la trasmissione del payload è stato concepito in questo modo.
Una variabile *mydata[]* viene dichiarata come una stringa di 6 byte in cui scrivere i seguenti dati:

- byte 0 = lettera T
- byte 1 = dato di temperatura
- byte 2 = lettera H
- byte 3 = dato di umidità
- byte 4 = lettera L
- byte 5 = dato di luminosità

In questo modo, una volta inviata la stringa, si potrà visualizzare ogni singolo byte in base alla sua posizione all'interno della stringa. Nella funzione *do_send* dello sketch di esempio abbiamo inserito le funzioni di lettura dei sensori:
dht_read()
light_read()

Queste memorizzano i dati dei sensori nelle posizioni pari della stringa in questo modo:
mydata[1]=tem; //dato di temperatura
mydata[3]=hum; //dato di umidità
mydata[5]=lux; //dato di luminosità

I valori ASCII corrispondenti alle lettere T, H e L vengono memorizzati nelle posizioni dispari della stringa:
mydata[0]=84; //valore ASCII lettera T

Listato 1

```
#include <dht.h>
dht DHT;
#define DHT11_PIN 4
#define PIN_LDR A0

float temperature, humidity; //valori float di lettura dal DHT11
int tem, hum, lux; //variabili da inviare a TTN
uint8_t mydata[] = "000000"; //payload da inviare

float valR = 0;      //lettura dell'ingresso analogico (resistenza)
float Vout = 0.0;    //voltaggio in uscita del partitore
float Vin = 5.0;     //tensione di alimentazione
float R_nota = 10000.0;  //valore della resistenza nota (10 Kohm)
float val_lux;           //valore lux
float Ldr = 0.0;         //valore calcolato della resistenza Ldr
float Ldr_1 = 75000.0;   //valore ldr di illuminamento unitario
float slope  = 0.66;     //valore gamma fotoresistenza (pendenza)
double Lux = 0.0;        //lux calcolati

void dht_read()
{
void dht_read()
{
      temperature = DHT.read11(DHT11_PIN); //rilevazione della temperatura
      tem = DHT.temperature;
      humidity = DHT.read11(DHT11_PIN); //rilevazione dell'umidità
      hum = DHT.humidity;
      mydata[0]=84; //lettera T
      mydata[1]=tem; //gradi temperatura
      mydata[2]=72; //lettera H
      mydata[3]=hum; //percentuale umidità

      Serial.println("Temperature - Humidity:");
      Serial.print(tem);
      Serial.print("°C");
      Serial.print(" - ");
      Serial.print(hum);
      Serial.print("%");
      Serial.println();
}

void light_read(){
  valR = 0;
  for ( int i = 0; i<= 4; i++)  //lettura di 5 valori di LDR
  {
    int val_luxLdr = analogRead(PIN_LDR);
    delay (40);
    valR = valR + val_luxLdr;
  }
   valR = valR/5; // media aritmetica dei valori letti
  //calcolo del valore della resistenza LDR
  Vout = (Vin/1024.0 * valR);    //conversione del valore in Volt
  Ldr = ((R_nota * Vin/Vout )- R_nota);    //calcolo della resistenza

  //calcolo dei Lux
  Lux = pow((Ldr/Ldr_1), (1.0/-slope));
  lux = Lux; //double to int
  mydata[4]=76; //lettera L
  mydata[5]=lux; //invio dei lux
}

void do_send(osjob_t* j){
    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println("OP_TXRXPEND, not sending");
    } else {

        dht_read(); //lettura DHT11
        light_read(); //lettura LDR

        LMIC_setTxData2(1, mydata, sizeof(mydata), 0);
    }
}
```

mydata[2]=72; //valore ASCII lettera H
mydata[4]=76; //valore ASCII lettera L

Dopo aver inviato il payload al *device_ 1* dell'applicazione *futura_devices* che abbiamo precedentemente creato su TTN, quello che si vedrà nella sezione *Data* del *device_ 1* sarà simile alla **Fig. 10**.
Nell'esempio si possono distinguere i valori ASCII seguiti rispettivamente dai dati dei sensori:
54 1B 48 2D 4C 3D 00

Fig. 10 - Il payload ricevuto dal device_1.

Fig. 11 - Node-RED con il nodo debug e il payload ricevuto.

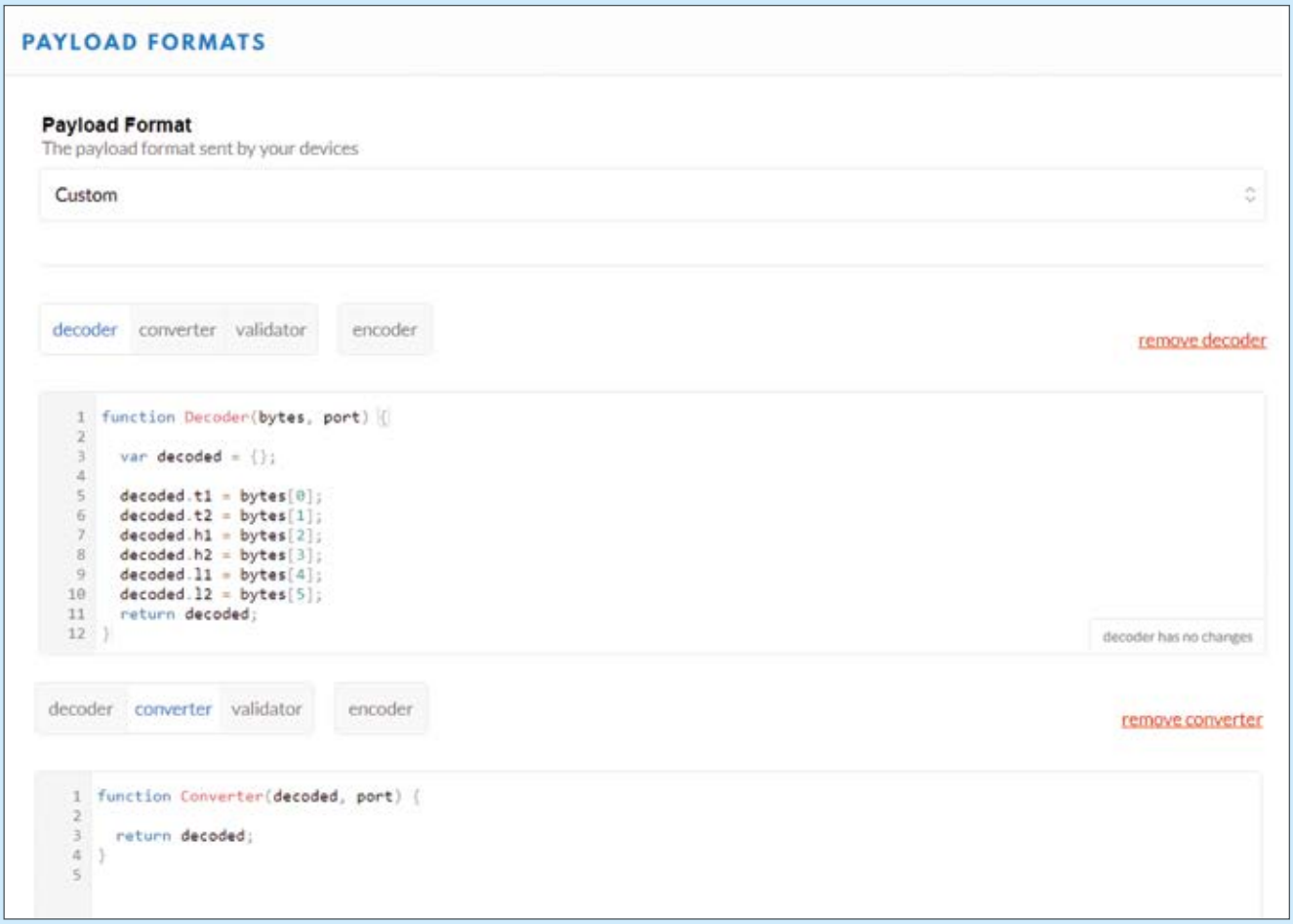


Fig. 12 - Le funzioni Decoder e Converter.

Fig. 13 - La conversione effettuata in base al Decoder del Payload Format.

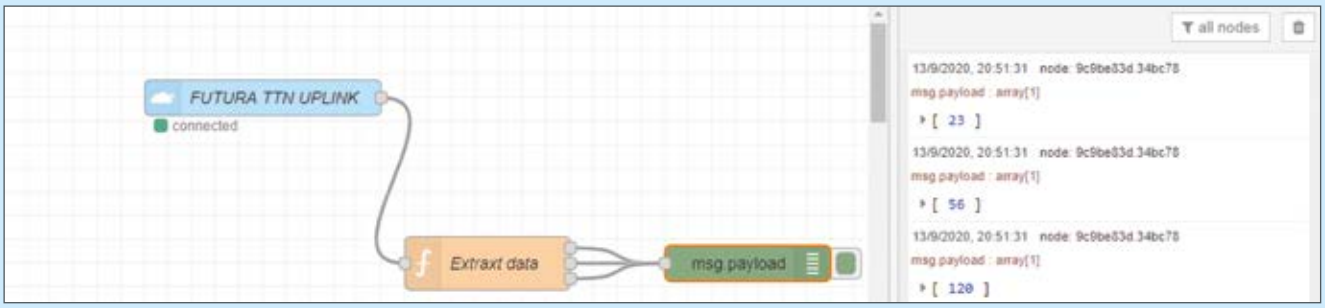


Fig. 14 - Il nodo Function.

84 27 72 45 76 61 0 (in decimale)

Per vedere gli stessi dati ricevuti dal nodo *ttn uplink* basterà aggiungere un nodo *debug* come illustrato in **Fig. 11**.
Nella finestra di *debug* si vedranno i dati ricevuti dal nodo, convertiti in valori decimali:

[84, 27, 72, 45, 76, 61, 0]

A questo punto si rende necessaria la cosiddetta "formattazione del payload" in modo da poter filtrare successivamente valori numerici e lettere in base alla loro posizione nel payload.

PAYLOAD FORMAT

Nella sezione *Payload Formats* dell'applicazione sono presenti le funzioni *Decoder*, *Converter*, *Validator* e *Encoder*. Per impostazione predefinita sono vuote e pertanto si dovrà scrivere un piccolo Javascript, rispettivamente per la funzione *Decoder* e *Converter*, come illustrato in **Fig. 12.**

```
function Decoder(bytes, port) {
  var decoded = {};
  decoded.t1 = bytes[0];
  decoded.t2 = bytes[1];
  decoded.h1 = bytes[2];
  decoded.h2 = bytes[3];
  decoded.l1 = bytes[4];
  decoded.l2 = bytes[5];
  return decoded;
}
```

Fig. 15 - L'editor del nodo Function.

Listato 2

```
var obj =
{ 0:msg.payload.t1,
  1:msg.payload.t2,
  2:msg.payload.h1,
  3:msg.payload.h2,
  4:msg.payload.l1,
  5:msg.payload.l2
}

var result = Object.keys(obj).map(function(key)
{
      return [Number(key), obj[key]];
});

var res_temp = result[1].slice(1);
var res_humi = result[3].slice(1);
var res_light = result[5].slice(1);
console.log("Temp",res_temp);
console.log("Humi",res_humi);
console.log("Light",res_light);

var msg1 = { payload:res_temp };
var msg2 = { payload:res_humi };
var msg3 = { payload:res_light };
return [ [msg1], msg2,msg3];
```

```
function Converter(decoded, port) {
  return decoded;
}
```

Così facendo il payload verrà formattato in modo simile alla **Fig. 13,** in cui si possono vedere i dati chiaramente suddivisi nei rispettivi valori numerici decimali e caratteri ASCII.

Dopo la conversione si potrà leggere:

h1: 72 //lettera H
h2: 56 //dato di umidità
l1: 76 //lettera L
l2: 120 // dato di luminosità
t1: 84 //lettera T
t2: 23 // dato di temperatura

ESTRAZIONE DEI DATI

Giunti a questo punto basterà aggiungere nel Flow una funzione per l'estrazione dei dati del payload così formattato (**Fig. 14**). L'uscita del nodo funzione andrà poi collegata al nodo debug per visualizzare i dati in arrivo. La funzione di estrazione è un semplice Javascript che legge i dati del payload formattato. Il **Listato 2** riporta il codice da scrivere all'interno del nodo funzione per implementarla.
Facendo doppio clic sul nodo Function si aprirà l'editor della funzione (**Fig. 15**). Qui si può notare che le variabili *res_temp*, *res_humi* e *res_light* sono il risultato della funzione *slice* che splitta i valori della variabile *result*, restituendo solo i valori numerici memorizzati nelle posizioni 1, 3 e 5, escludendo le lettere (caratteri ASCII) memorizzate nelle posizioni 0, 2 e 4.
Si fa notare che l'uscita del nodo possiede tre uscite verso le quali vengono indirizzati rispettivamente i valori di ritorno delle variabili *msg1*, *msg2* e *msg3*, corrispondenti ai valori delle variabili *res_temp*, *res_humi* e *res_light*.
Collegando queste tre uscite al nodo debug si potranno vedere i valori nella finestra di debug relativi a temperatura, umidità e luminosità.

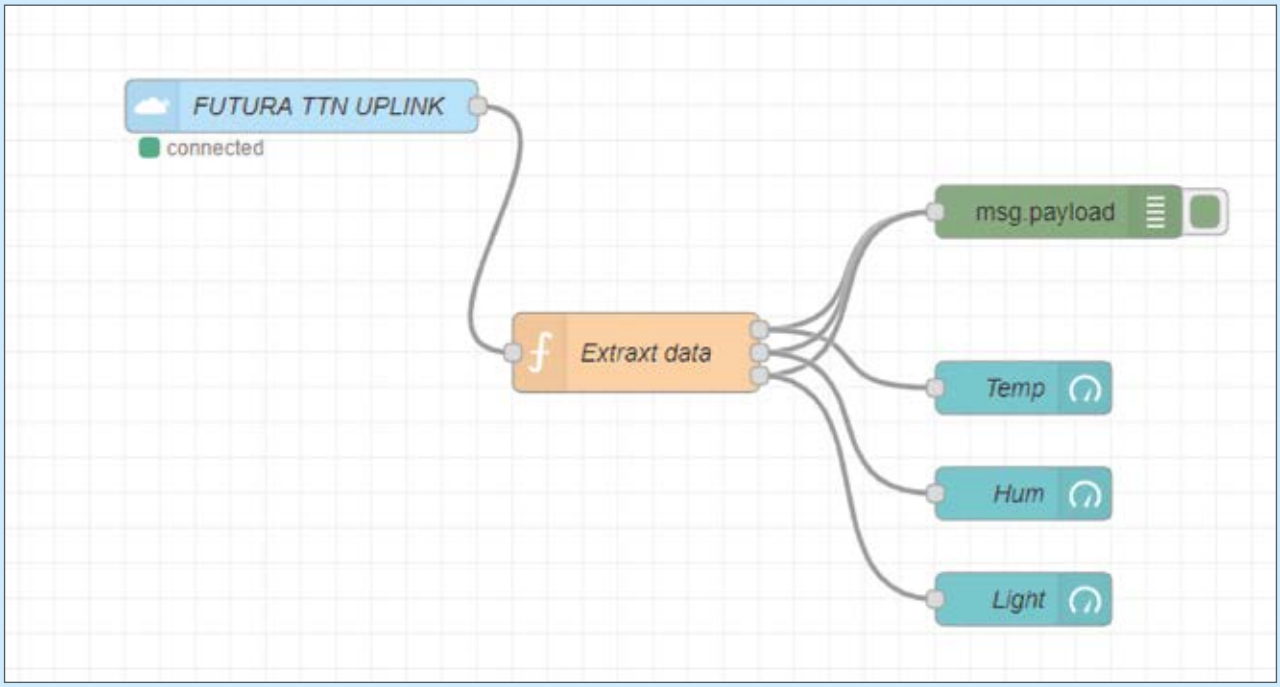


Fig. 16 - I tre widget collegati alle uscite del nodo funzione.

Opzionalmente si può dare anche un nome al nodo Function, che nel nostro caso è stato scelto come *Extract data*.

AGGIUNGERE I WIDGET DELLA DASHBOARD

Ora che abbiamo tutti i dati estratti con la funzione *Extract data*, possiamo collegare alle tre uscite del nodo funzione i tre widget della dashboard, come illustrato in **Fig. 16**. Come si può vedere possiamo lasciare collegato anche il nodo debug. Senza entrare nel dettaglio, spieghiamo brevemente le operazioni per modificare i tre widget che funzioneranno come misuratori di temperatura, umidità e luminosità.
Facendo doppio clic su un nodo *Gauge* si aprirà il relativo editor (***Fig. 17***) in cui è possibile modificare l'aspetto grafico e la tipologia del misuratore.

- Nel campo *Group* si può creare il gruppo di appartenenza del widget ovvero la sua posizione nella pagina del browser. Facendo clic sull'icona matita si può dare un nome al gruppo. Nel nostro caso il nome scelto è *FUTURA DEVICE 1*. Il gruppo predefinito è *Home*, ma si può digitare un nome qualsiasi.
- Nel campo *Size* si può decidere la dimensione del widget come spiegato in precedenza. Di default il parametro è *auto*.
- Nel campo *Type* si può scegliere il tipo del misuratore tra *Gauge*, *Donut*, *Compass* e *Level*.
- Nel campo *Label* si può dare un nome all'etichetta che appare nel misuratore.
- Il campo *Value* si può lasciare inalterato perché assumerà di volta in volta il valore del messaggio corrispondente al dato del sensore.
- Nel campo *Units* si può scrivere il tipo di unità, per esempio, gradi, %, lux e così via.
- Nel campo *Range* si stabiliscono i valori minimi e massimi della misurazione.
- Il parametro *Color gradient* è disponibile per i tipi *gauge* e *donut* e imposta il colore del gradiente per l'anello del misuratore.
- Nel campo *Sectors* si possono decidere opzionalmente i settori di suddivisione.
- Nel campo *Name* si può dare opzionalmente il nome al widget.

Una volta modificati i tre widget basterà distribuire il Flow con il pulsante *Deploy* per vedere il risultato nel browser web, puntando all'indirizzo **127.0.0.1:1800/ui**. Se non avete commesso errori, vedrete qualcosa di simile a quanto mostrato nella **Fig. 18**.

CONCLUSIONI

Per il momento ci fermiamo qui. Con la speranza di essere stati sufficientemente chiari, vi diamo appuntamento alla prossima puntata, nella quale approfondiremo l'uso di Node-RED per altre utili funzioni che vi descriveremo nell'occasione.

Fig. 17 - L'editor del widget gauge.

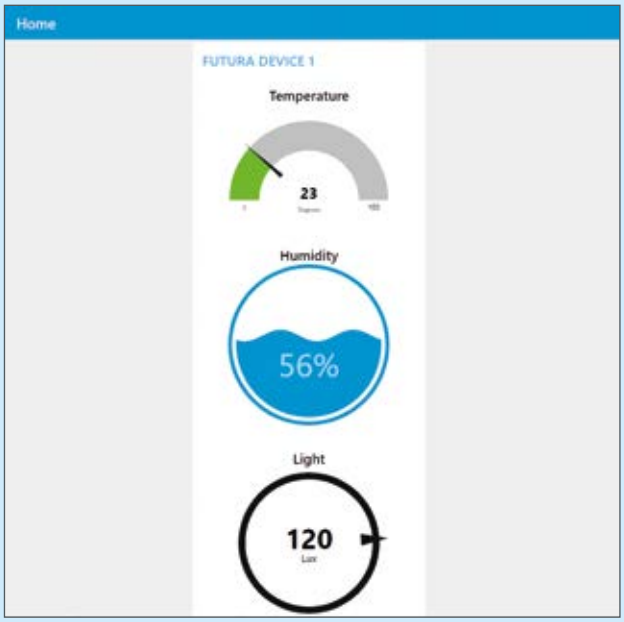


Fig. 18 - I tre widget con le misurazioni in tempo reale di temperatura, umidità e luminosità.