

Node-RED

INDICE

1.	REVISIONI	3
2.	RINGRAZIAMENTI.....	3
3.	PREMESSA.....	3
4.	STORIA.....	4
5.	SISTEMI SUPPORTATI.....	4
6.	INSTALLAZIONE.....	4
6.1.	Raspian (Raspberry PI).....	4
6.1.	Windows.....	4
7.	AVVIO	5
7.1.	Introduzione ai nodi.....	5
8.	IL NOSTRO PRIMO PROGETTO	6
8.1.	Creazione	6
8.1.	Esportazione e Importazione.....	7
9.	LIBRERIE di NODI	9
9.1.	Installazione	9
10.	AUTENTICAZIONE.....	10
10.1.	Autenticazione basata su credenziali username / password	10
10.1.	Generazione hash password	10
11.	Sintassi JSON	11
11.1.	Introduzione al JSON.....	11
11.2.	Utilizziamo il json in node-red	14
11.3.	Decodifica di un json	16
12.	Comunicazione tra nodi (oggetto msg).....	16
13.	Dashboard node-red.....	17
13.1.	Introduzione alla dashboard.....	17
13.1.	Layout.....	19
13.1.1.	Tab.....	19
13.1.1.	Link.....	26
13.2.	Site	27
13.1.	Theme	28
14.	Enciclopedia dei nodi base.....	28
14.1.	Input.....	28
14.2.	Output	28
14.3.	Function	29
14.4.	Social	29
14.5.	Storage	29
14.6.	Analysis	30
14.7.	Advanced	30
14.8.	Raspberry Pi.....	30
14.9.	Dashboard	30
14.10.	Network	30

1. REVISIONI

Data	Revisione	Descrizione
21/03/2018	0.0	Prima emissione
14/07/2018	0.1	Aggiunto capitolo autenticazione
22/01/2019	0.2	Elaborazione payload, Enciclopedia dei nodi
09/01/2019	0.3	Aggiunto capitolo oggetto msg
02/02/2020	0.4	Aggiunto capitolo Dashboard

2. RINGRAZIAMENTI

Questo manuale è stato realizzato per chi vuole approcciarsi al grande mondo di Node-red e ogni contributo è ben accetto.

Vi invito a partecipare sul gruppo Telegram **@noderedIT** oppure **t.me/noderedIT**

Si ringraziano per la partecipazione gli utenti:

Folgore

Rossella

3. PREMESSA

Questa guida si rivolge a un'utenza che sappia utilizzare un PC, quindi le operazioni tipo aprire la shell e altre varie operazioni sul browser sono omesse. Saranno invece spiegate nel dettaglio le operazioni da effettuare su node-red.

Node-RED è un editor di flusso basato su Node.js che punta a un browser Web che consente agli utenti di collegare dispositivi hardware, API e servizi online in modi nuovi e interessanti.

All'interno del browser è possibile creare l'applicazione trascinando i nodi dalla tavolozza in un'area di lavoro e iniziare a collegarli tra loro. Con un solo clic, l'applicazione viene inviata nel runtime in cui viene eseguita.

La tavolozza dei nodi può essere facilmente estesa installando nuovi nodi creati dalla comunità e i flussi creati possono essere facilmente condivisi come file JSON.

I nodi di Node-RED sono come i pacchetti npm e puoi ottenerli allo stesso modo. E poiché Node-RED ha un editor di testo integrato, puoi rendere le applicazioni complesse come vuoi aggiungendo le funzioni JavaScript.

Poiché Node-RED è basato su Node.js e sfrutta il modello non bloccante e basato sugli eventi che può essere eseguito su hardware a basso costo come il Raspberry Pi o nel cloud.

Vantaggi dell'architettura

1. Essendo un thread asincrono assicura una velocità elevata di esecuzione.
2. Espandibilità
3. Flessibilità (domotica, elaborazione dati, gateway, interfaccia HMI)
4. Il server può essere installato su hardware obsoleto o con requisiti minimi.
5. Velocità di programmazione.
6. Visualizzazione grafica del codice.
7. Aggiornamenti, ogni giorno ci sono nuove librerie e aggiornamenti di quelle già presenti.

4. STORIA

Node-RED ha iniziato la sua vita all'inizio del 2013 come progetto secondario di Nick O'Leary e Dave Conway-Jones del gruppo Emerging Technology Services di IBM.

Quello che era iniziato come un proof-of-concept per visualizzare e manipolare le mappature tra gli argomenti MQTT, divenne rapidamente uno strumento molto più generale che poteva essere facilmente esteso in qualsiasi direzione.

È diventato open source a settembre 2013 e da allora è stato sviluppato, culminando in uno dei progetti fondatori della JS Foundation nell'ottobre 2016.

5. SISTEMI SUPPORTATI

Il server di Node-red può essere installato sui seguenti sistemi:

- Linux (testato su Ubuntu, raspbian, dietpi)
- Windows (testato personalmente su Windows 7 SP1 e Windows 10)
- Android (su simulatore di linux Termux)

6. INSTALLAZIONE

6.1. Raspian (Raspberry PI)

Dopo la configurazione di raspbian e la sua configurazione si può procedere con l'installazione di Node-red con questo comando

```
bash <(curl -sL https://raw.githubusercontent.com/node-red/raspbian-deb-package/master/resources/update-nodejs-and-nodered)
```

Esso esegue in automatico tutta la procedura di installazione, da node js a node-red

Al termine dell'installazione è possibile lanciare node-red :

```
Node-red-start
```

O eseguirlo come servizio e quindi all'avvio del sistema:

```
sudo systemctl enable nodered.service
```

6.1. Windows

Per prima cosa non possiamo usare il comando precedente che è esclusivo per la raspberry, quindi dobbiamo scaricare la versione consigliata di nodejs dal sito <https://nodejs.org/it/> e installiamo.

Al termine riavviamo e successivamente apriamo la shell di windows cliccando su start e digitando cmd.exe (per windows 10) apriamo con diritti amministrativi e lanciamo il seguente comando:

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

Al termine dell'installazione si avvia con il comando

```
node-red
```

7. AVVIO

Se siamo sicuri di averlo avviato apriamo il browser e puntiamo all'indirizzo IP del server, se siamo sulla stessa macchina <http://127.0.0.1:1880> . Se tutto è stato eseguito correttamente dovremmo vedere questa interfaccia.

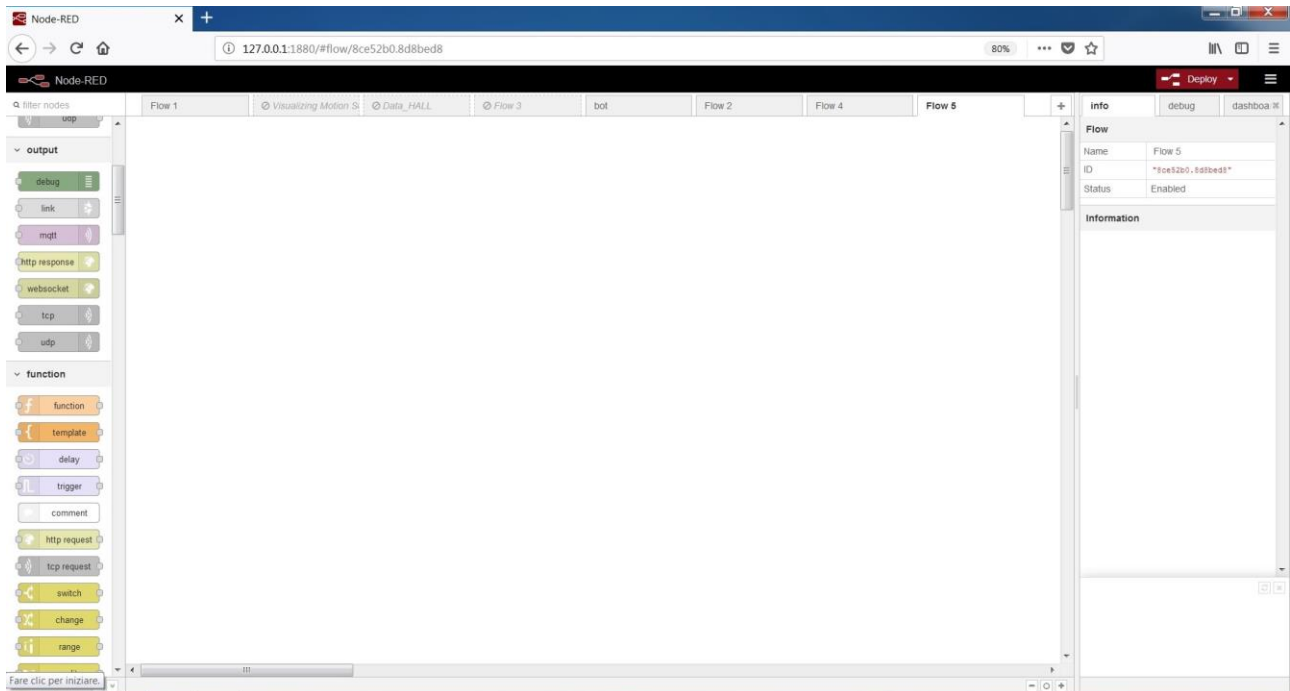



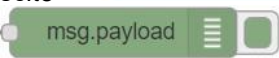
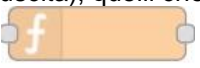
Figura 1 - Interfaccia web

L'interfaccia è suddivisa in tre sezioni, a sinistra ci sono le librerie dei **nodi**, al centro l'area di sviluppo suddivisa in varie pagine chiamate **flow** e a destra le tab di informazioni in base alla selezione, il debug e la dashboard.

7.1. Introduzione ai nodi





Vediamo come sono strutturati i nodi, i nodi non sono messi alla rinfusa, come vediamo sono suddivisi in librerie. Nell'immagine **Figura 1 - Interfaccia web** vediamo la libreria output e function.

I nodi si possono distinguere (le diciture seguenti le ho date io per una maggiore comprensione):

- Nodi di IN (ingresso), i quali hanno solo una o più uscite .
- Nodi di OUT (uscita), i quali prevedono un ingresso .
- Nodi di IN-OUT (ingresso-uscita), quelli che eseguono delle operazioni e generano un risultato (spesso, ma non sempre) .

Attenzione che i nodi non possono avere più ingressi, solo più uscite.

I nodi presentano graficamente delle altre segnalazioni:


-  un nodo appena inserito o modificato avrà un pallino blu, oltre a questo si attiva il pulsante . Il nodo non è ancora caricato sul server.
-  il triangolo rosso evidenzia che c'è un errore da correggere.
-  alcuni nodi presentano inoltre delle segnalazioni aggiuntive tipo stato della connessione.

Prestiamo sempre attenzione prima di fare il deploy con nodi in errore, potrebbe comportare il crash di node-red.

8. IL NOSTRO PRIMO PROGETTO

8.1. Creazione

Ora che siamo davanti all'interfaccia grafica, creiamo il nostro primo progetto per testarne il funzionamento.

Selezioniamo il nodo **inject**  e con un drag&drop lo mettiamo nel nostro flow. Con un doppio click.

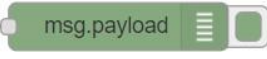

A questo punto aggiungiamo con la stessa procedura il nodo di **debug** . Ora per farli funzionare assieme dobbiamo collegarli. Clicchiamo con il mouse sul quadratino grigio di uno dei due e lo colleghiamo con l'altro quadrino grigio.



Figura 2 - Primo progetto

Come possiamo vedere hanno i pallini blu, non sono ancora salvati sul server, quindi sul  e aspettiamo la conferma.


Perfetto ora possiamo cliccare sul pulsante azzurro dell'inject  e vedere cosa succede nella tab debug sulla destra.



Figura 3 – Debug

Essendo che di default l'inject node è impostato come timestamp, viene visualizzato il tempo in formato UTC.

Ora proviamo a modificare cosa deve inviare, doppio click su nodo inject per aprire la seguente videata:

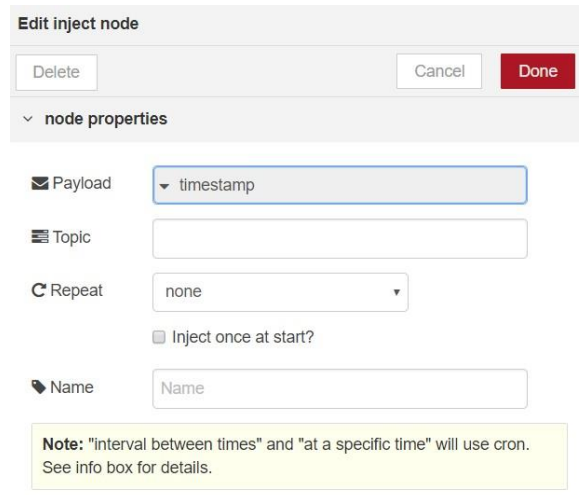


Figura 4 - Configurazione nodo Inject

Al posto di timestamp selezioniamo string e scriviamo Hello world!, clicchiamo su Done e rifacciamo il deploy.

Riproviamo la procedura precedente e controlliamo il debug. Se tutto è corretto visualizzeremo:




Figura 5 - Debug 'Hello World!'

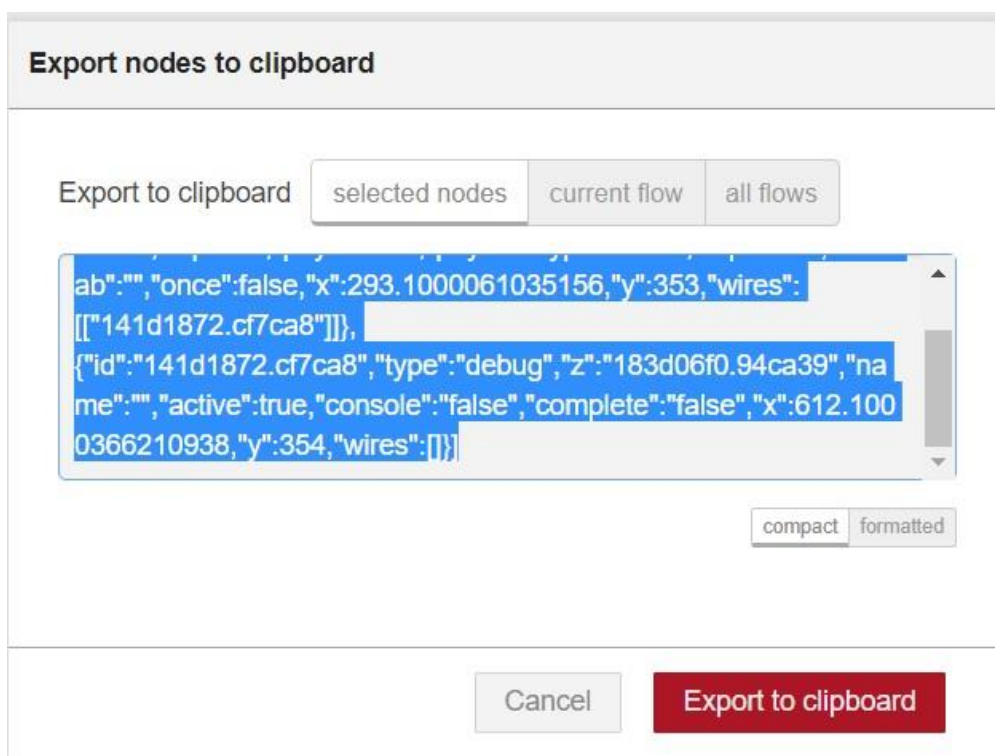
Dal debug notiamo anche una cosa interessante. C'è una variabile utilizzata per lo scambio dati: **msg.payload** che nella Figura 3 – Debug era un formato Numero e nella Figura 5 - Debug 'Hello World!' è una stringa.

8.1. Esportazione e Importazione

Ora che il nostro progetto è funzionante è il caso di salvarne una copia (per usi futuri o per backup). Per

prima cosa selezioniamo tutto. Dopodiché andiamo in alto a destra sull'hamburger  (si chiama così) a destra del tasto deploy e selezioniamo export e successivamente clipboard.

Viene mostrata la seguente finestra:

**Figura 6 – Esportazione**

Qui possiamo scegliere se esportare i nodi selezionati precedentemente, il flow corrente o tutti i flow.

Cliccando poi sul pulsante 'Export to clipboard' viene copiato il codice. Creiamo un nuovo file di testo e incolliamo. Salviamo il file ed avremmo il nostro codice.

```
[{"id":"e009842","type":"inject","z":"183d06f0.94ca39","name":"","topic":"","pay  
load":"","payloadType":"date","repeat":"","crontab":"","once":false,"x":293.1000  
061035156,"y":353,"wires":[["141d1872.cf7ca8"]]}, {"id":"141d1872.cf7ca8","type":  
"debug","z":"183d06f0.94ca39","name":"","active":true,"console":"false","complet  
e":"false","x":612.1000366210938,"y":354,"wires":[]}]
```

Ecco cosa abbiamo copiato nel file di testo. E possiamo già fare la procedura di importazione con la procedura precedente, ma al posto di selezionare Export, selezioniamo Import.

Import nodes

Paste nodes here

Import to

Figura 7 – Importazione

Incolliamo nel rettangolo grigio e selezioniamo import, scegliendo se importare nel flow corrente oppure in un nuovo flow, e decidiamo dove posizionarlo.

ATTENZIONE i nodi con dati sensibili tipo API non vengono salvate nel processo di esportazione.

9. LIBRERIE di NODI


Come abbiamo visto finora sembra che le possibilità siano abbastanza limitate o che ci sia un'enorme mole di lavoro per poter sviluppare delle logiche in base a quello che vogliamo fare. Per questo si dovrà quasi necessariamente aggiungere delle librerie che ci semplificheranno il compito da eseguire.

Faccio notare che quotidianamente vengono aggiunte sul sito nuove librerie e update di quelle esistenti. Le librerie di nodi sono raggiungibili a questo indirizzo <https://flows.nodered.org>

Le librerie permettono di aggiungere funzionalità a node-red, dalla comunicazione alle statistiche o semplicemente aggiungere nuove funzioni senza che dobbiamo costruircele noi.

9.1. Installazione

L'installazione delle nuove librerie possiamo operare tramite comando in shell (sempre segnalato nella

pagina di descrizione della libreria) o direttamente da node-red. Clicchiamo sull'hamburger  e selezioniamo Manage Palette. A questo punto dovrebbe mostrarci due schede le quali mostrano le librerie installate e quelle installabili. Selezioniamo Install e cerchiamo la libreria che ci serve. A quel punto selezioniamo Install e aspettiamo la conferma di installazione. Si consiglia sempre di riavviare node-red al termine dell'installazione.

10. AUTENTICAZIONE

In questo capitolo spieghiamo come proteggere node-red, aggiungendo l'autenticazione tramite nome utente e password.

Per poter abilitare l'autenticazione, bisogna editare il file setting :

Windows : C:\Users\nome_utente\.node-red\settings.js (nome_utente da sostituire con il nome di accesso di Windows)

Linux (raspberry): ./node-red/settings.js

10.1. Autenticazione basata su credenziali username / password

Per abilitare l'autenticazione utente su Editor e Admin API, aggiungi o rimuovi i commenti quanto segue al tuo file settings.js:

```
adminAuth: {
  type: "credentials",
  users: [{
    username: "admin",
    password:
"$2a$08$zZWtXTja0fB1pzD4sHcMyOCMyzZ6dNbM6tl8sJogENOMcxWV9DN.",
    permissions: "*"
  }]
}
```

Questa configurazione di esempio definisce un singolo utente chiamato admin che ha il permesso di fare tutto all'interno dell'editor.

All'interno della chiamata users si possono inserire diversi utenti ciascuno dei quali può avere autorizzazioni diverse.

```
type: "credentials",
users: [ /* list of users */ ],
default: {
  permissions: "read"
```

Nota: nelle versioni precedenti di Node-RED, l'impostazione httpAdminAuth potrebbe essere utilizzata per abilitare l'autenticazione di base HTTP Basic Authentication. Questa opzione è deprecata e non dovrebbe essere utilizzata.

Per generare la password Hash per accedere all'area di node-red installare il pacchetto

node-red \$ npm install -g node-red-admin

Nota: sudo è richiesto se si esegue come utente non root su Linux / OS X. Se è in esecuzione su Windows, sarà necessario eseguire una shell di comandi come Amministratore, senza il comando sudo.

10.1. Generazione hash password

Per generare un hash di password adatto, è possibile utilizzare lo strumento da riga di comando node-red-admin digitando:

node-red-admin hash-pw

Lo strumento chiederà la password che si desidera utilizzare e quindi stamperà l'hash che può essere copiato nel file delle impostazioni (setting.js)
esempio password:

"\$2a\$08\$zZWtXTjx0fB1pzD4xHCMYOCMYz2Z6dNbM6tl8xJogENOMxxWV9DN."

Se tutto è stato eseguito correttamente, quando ci collegheremo tramite browser a node-red ci verrà richiesta l'autenticazione come mostrato nella figura successiva.



Figura 8 - Autenticazione di node-red

11. Sintassi JSON

11.1. Introduzione al JSON

Il JSON (**JavaScript Object Notation**) è un semplice formato per lo scambio di dati. Per le persone è facile da leggere e scrivere, mentre per le macchine risulta facile da generare e analizzarne la sintassi. Si basa su un sottoinsieme del Linguaggio di Programmazione JavaScript.

JSON è un formato di testo completamente indipendente dal linguaggio di programmazione, ma utilizza convenzioni conosciute dai programmatori di linguaggi della famiglia del C, come C, C++, C#, Java, JavaScript, Perl, Python, e molti altri. Questa caratteristica fa di JSON un linguaggio ideale per lo scambio di dati.

Un JSON è strutturato così:

- Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
- Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.



In JSON, assumono queste forme:

Un oggetto è una serie non ordinata di nomi/valori. Un oggetto inizia con { (parentesi graffa sinistra) e finisce con } (parentesi graffa destra). Ogni nome è seguito da : (due punti) e la coppia di nome/valore sono separata da , (virgola).

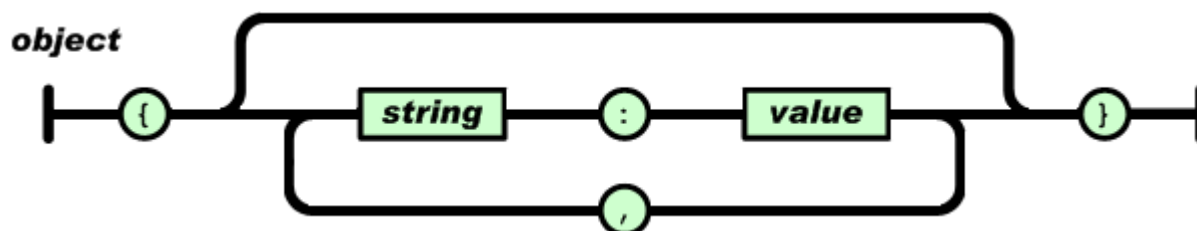


Figura 9 - Struttura dell'oggetto

Un array è una raccolta ordinata di valori. Un array comincia con [(parentesi quadra sinistra) e finisce con] (parentesi quadra destra). I valori sono separati da , (virgola).

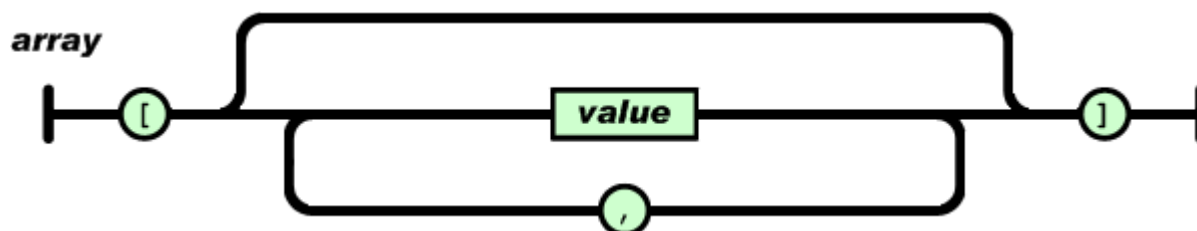


Figura 10 - Struttura dell'array

Un valore può essere una stringa tra virgolette, o un numero, o vero o falso o nullo, o un oggetto o un array. Queste strutture possono essere annidate.

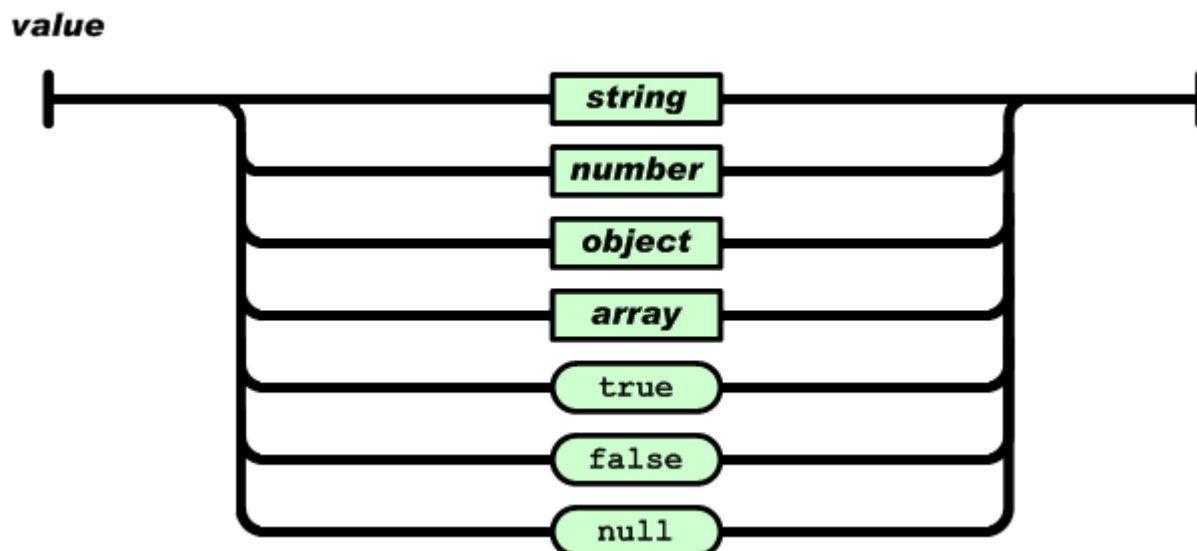


Figura 11 - Tipo di dati

Una stringa è una raccolta di zero o più caratteri Unicode, tra virgolette; per le sequenze di escape utilizza la barra rovesciata. Un singolo carattere è rappresentato come una stringa di caratteri di lunghezza uno. Una stringa è molto simile ad una stringa C o Java.

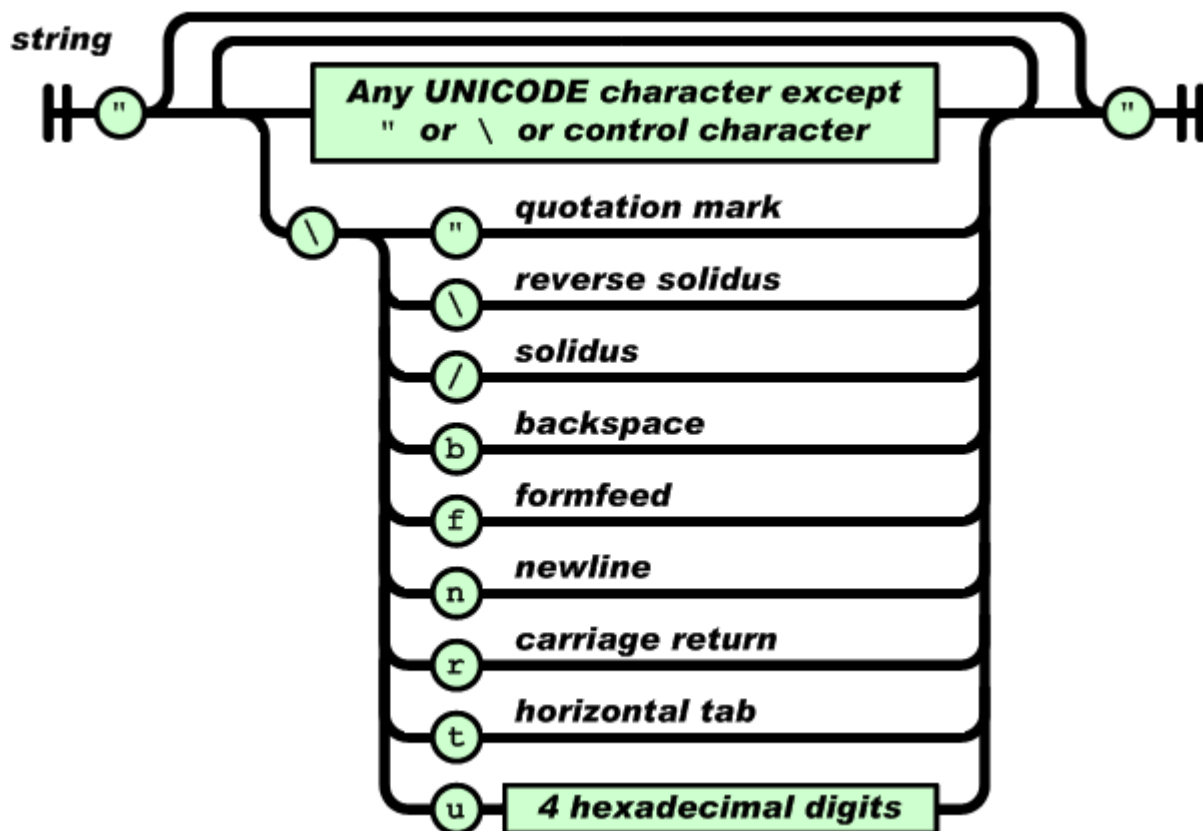


Figura 12 - Stringa

Un numero è molto simile ad un numero C o Java, a parte il fatto che i formati ottali e esadecimali non sono utilizzati.

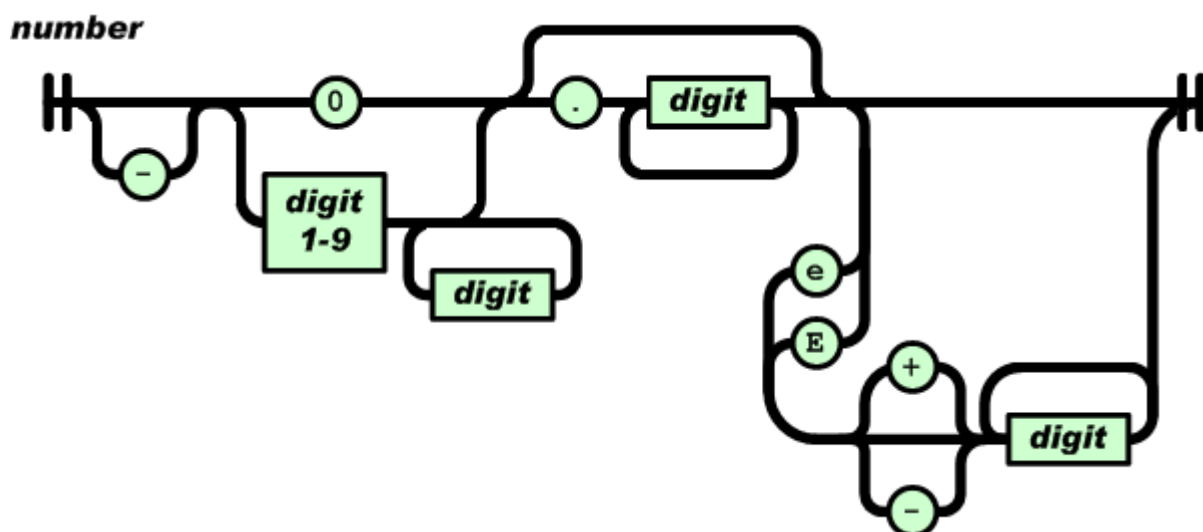


Figura 13 - Tipo number

I caratteri di spaziatura possono essere inseriti in mezzo a qualsiasi coppia di token.

A parte alcuni dettagli di codifica, questo descrive totalmente il linguaggio.

Fonte (<https://www.json.org/json-it.html>)

11.2. Utilizziamo il json in node-red

Spieghiamo ora come funziona il collegamento tra i nodi i nodi e cosa ci dobbiamo aspettare. Innanzitutto il nostro nodo principale di diagnostica, il debug.

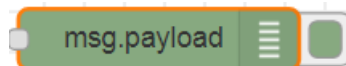


Figura 14 - Nodo debug

Esso effettua un print di quello che arriva in ingresso e lo mostra nella tab debug, nelle impostazioni può mostrare l'msg.payload oppure tutto il msg. L'interruttore a destra permette di abilitare o disabilitare il print.

Accennando al msg, esso è il messaggio di interconnessione tra i vari nodi. Diciamo che i nodi si parlano tramite una variabile msg. Dentro di essa ci può essere di tutto, la possiamo costruire noi oppure già costruita da sviluppatori terzi.

Vediamo un esempio. Inseriamo un inject e un debug e colleghiamoli insieme.

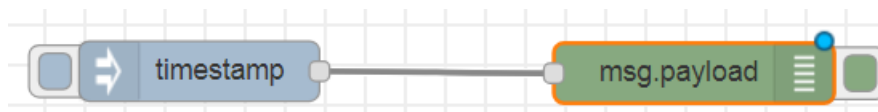


Figura 15 - Nodi debug e debug

Quando premiamo il pulsante sul nodo inject viene scritta la variabile msg.payload e il nodo debug la stampa nella tab debug.

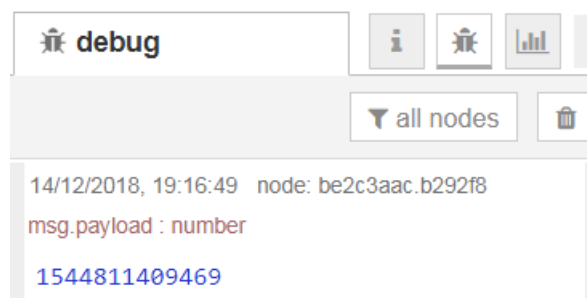


Figura 16 - Tab debug

Dal debug riceviamo il valore del variabile in quel punto (il timestamp in questo caso), vediamo il tipo di dato (number) e la variabile di riferimento (msg.payload). Di default è filtrato sul msg.payload, vediamo ora se impostiamo il complete message su nodo debug.

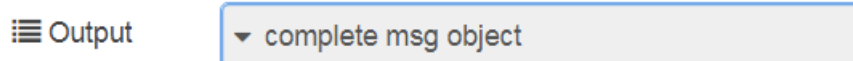


Figura 17 - Impostazione del debug

Il risultato sarà un oggetto, si vede sempre il valore del payload ma in più viene stampato un id del messaggio e il topic. Il topic è scritto dal nodo inject e in questo caso è stato lasciato vuoto e lo invia come stringa vuota.

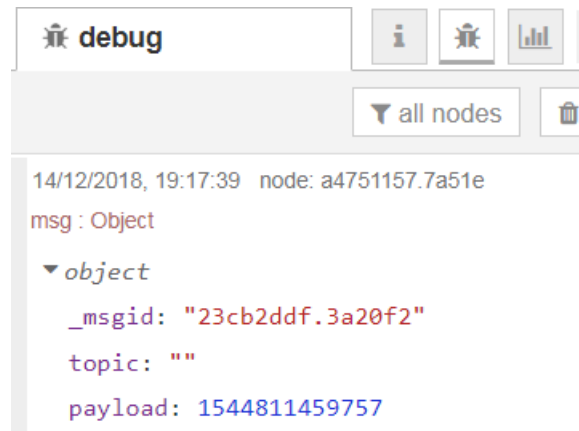


Figura 18 - Debug del msg

Aggiungiamo la stringa “data/ora” nel campo topic nel nodo inject. Il risultato è nell’immagine successiva.

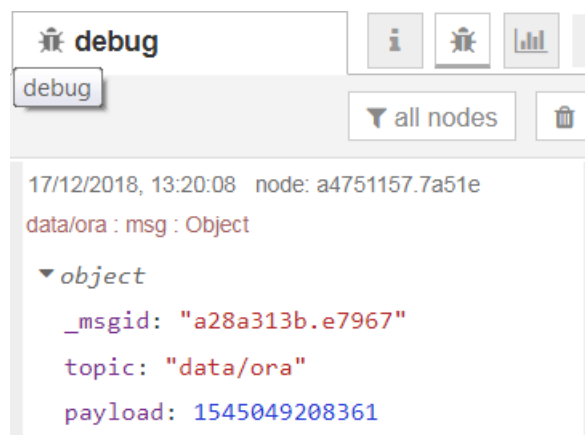


Figura 19 - Debug del msg

A cosa serve il topic? Possiamo usare il topic e/o inoltre creare il nostro json personalizzato (vedi capitolo successivo).

Un esempio pratico è nell’esempio scaricabile dal gruppo **9-multichart**. Il topic serve al nodo per distinguere i vari msg.payload dei valori in arrivo per identificare a quale penna appartengono (l’msg.topic).

11.3. Decodifica di un json

Molte volte abbiamo dei json che dobbiamo estrarre i dati singolarmente, anche perché molti nodi accettano in ingresso solo l'`msg.payload`. Di conseguenza dobbiamo decodificare il json, vediamo qualche esempio partendo dalle immagini precedenti.

Prendiamo come esempio questo json:

```
msg.payload={"1":1234,"2":5678}
```

Decodifichiamo il valore dal nome "1".

In questo caso lo trattiamo come fosse un array, `msg.payload= msg.payload[1]`

Decodifichiamo il valore dal nome "2".

In questo caso lo trattiamo come fosse un array, `msg.payload= msg.payload[2]`

Prendiamo un altro esempio questo json:

```
msg.payload={"cmd":"report",  
  "model":"magnet",  
  "sid":"158b0005c5548",  
  "short_id":3005,  
  "data":{"status":"open"}}
```

Questo è il json di lettura in uscita dal nodo xiaomi-in del magnete per porte finestre. A noi serve lo stato della porta: aperta o chiusa, e si trova nel valore del nome `status`.

```
msg.payload = msg.payload.data.status
```

12. Comunicazione tra nodi (oggetto msg)

Ora che sappiamo usare un json, dobbiamo fare chiarezza su come si passano i dati da un nodo ad un altro. Tutto questo è l'oggetto **msg**.

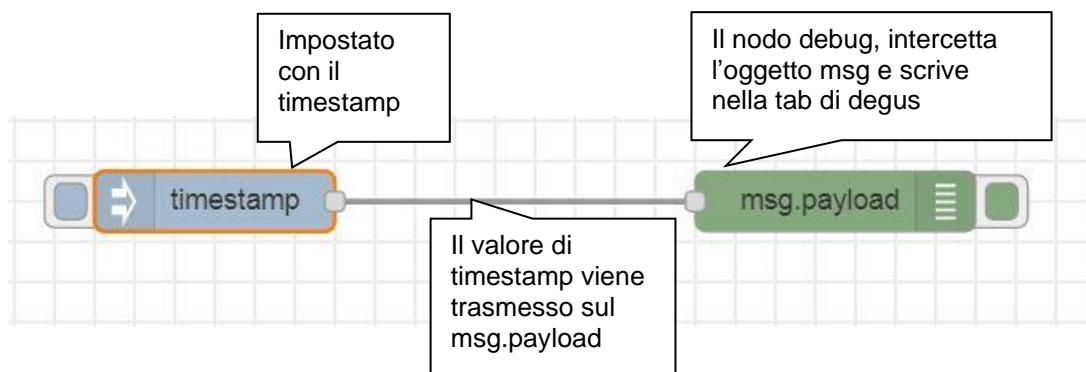


Figura 20 - oggetto msg

Qualunque cosa viene scritta in uscita da una function, sarà sempre scritto nell'oggetto `msg`. Vediamo, infatti il prossimo esempio.

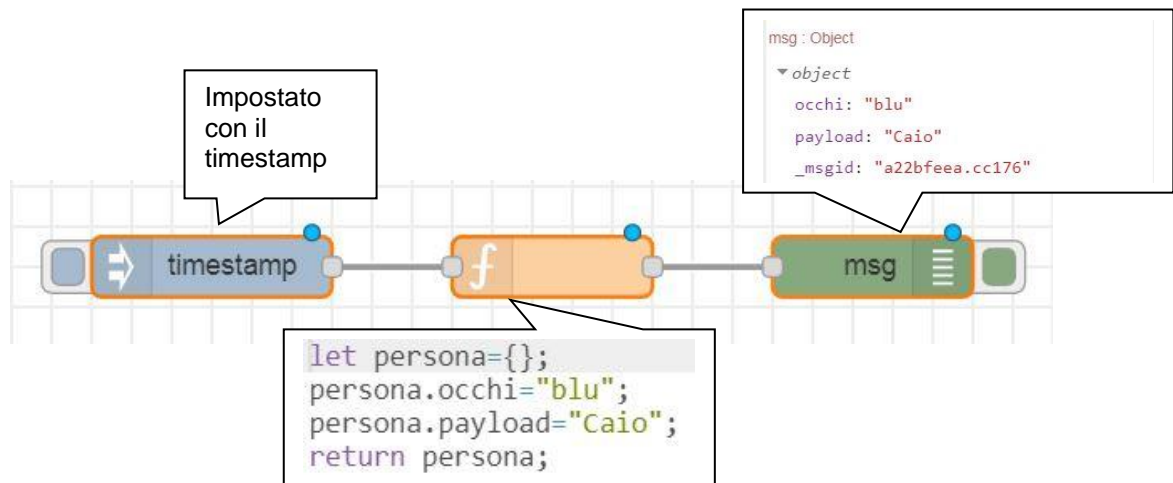
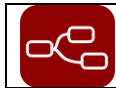


Figura 21 - oggetto msg

Come possiamo notare, l'inject invia di nuovo il timestamp, nella funzione però non viene minimamente utilizzato. In questo caso usiamo questo metodo solo per "attivare" la function che abbiamo creato (P.S, il nodo inject si può attivare all'avvio di node-red e ad intervallo di tempo).

La funzione crea un oggetto **persona**, e imposta le proprietà occhi e payload ed infine restituisce (con il return) l'oggetto persona appena creato. Quello che vediamo nel debug (ho impostato il messaggio completo per vedere tutto l'oggetto msg), vediamo:

1. persona.payload diventa msg.payload;
2. persona.occhi diventa msg.occhi.

Questo perché la variabile di scambio tra nodi è sempre l'oggetto msg.

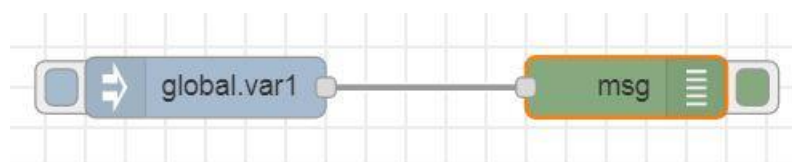


Figura 22 - oggetto msg

Anche se usiamo una variabile globale, il suo valore lo troveremo sempre su msg.payload.

13. Dashboard node-red

13.1. Introduzione alla dashboard

La dashboard di node-red si installa come un qualunque altro nodo (**node-red-dashboard**), o dal manage palette oppure manualmente come descritto dalla pagina web del nodo, giunta alla stesura di questo manuale alla versione 2.19.3.



Figura 23 - Tab Dashboard

La dashboard di node-red aggiunge una tab di fianco al debug.

La dashboard è suddivisa in 3 livelli:

- la **tab** , possiamo considerarla come la pagina web;
- il **gruppo** (la sezione dove sono raggruppati i vari oggetti grafici) ;
- l'**oggetto grafico** (praticamente il nodo che usiamo nel drawer, che può essere il pulsante, il testo,ecc)

Quando inseriamo un oggetto, siamo obbligati ad inserirlo in un gruppo e in una dashboard (se non esistono, bisogna crearli).

A questo punto, ci sono due metodi che si possono usare per creare la nostra dashboard:


- Passando dal nodo
- Usando la tab di destra (consigliato)


Prendiamoci 5 minuti di tempo per analizzare la dashboard Tab. La dashboard tab è suddivisa in Layout, Site e Theme.

13.1. Layout

13.1.1. Tab

Iniziamo con Layout (**Figura 23** - Tab Dashboard), da qua possiamo gestire e capire come è articolata la nostra dashboard, all'inizio è semplice gestire le varie sezioni, ma grazie a questo strumento visivo ad albero abbiamo la possibilità di avere sott'occhio tutto.

Abbiamo un pulsante per aggiungere Tab, uno per aggiungere link e  per aprire la dashboard.

Facciamo un esempio, creiamo una tab aggiungendo .

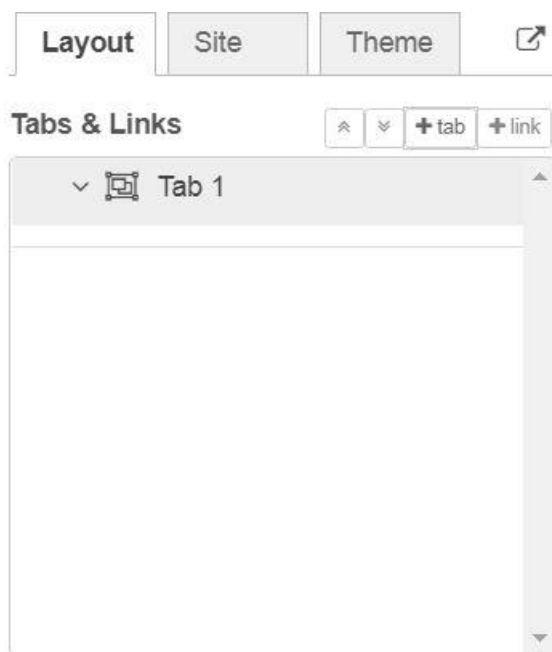


Figura 24 - Inseriamo una tab

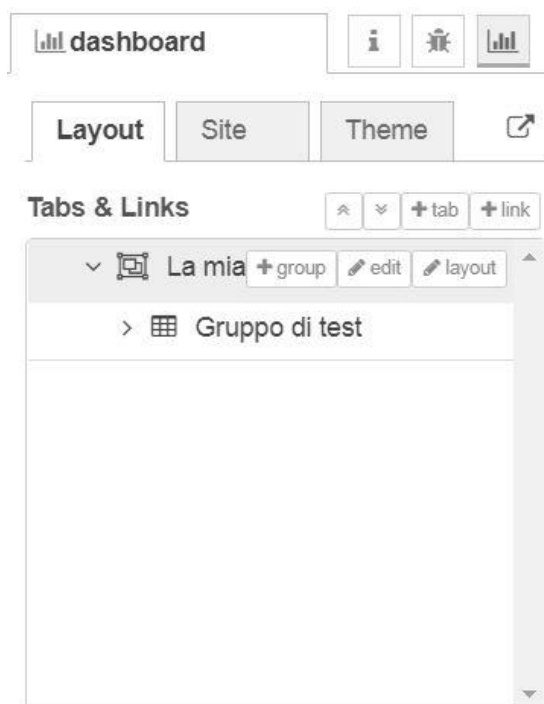


Figura 25 - Se passiamo sopra con il mouse vediamo i pulsanti di edit e aggiunta gruppo e layout

Clicchiamo su edit e si apre la finestra di configurazione.

Edit dashboard tab node

Delete
Cancel
Update

Properties

Name
Tab 1

Icon
dashboard

State
☒ Enabled

Nav. Menu
☒ Visible

The **Icon** field can be either a Material Design icon (e.g. 'check', 'close') or a Font Awesome icon (e.g. 'fa-fire'), or a Weather icon (e.g. 'wi-wu-sunny').

Figura 26 - Configurazione Tab

Inseriamo il nostro nome a piacimento, se abilitare la dashboard, e se renderla visibile a menu. Possiamo anche assegnargli un'icona.

Dopo aver aggiunto la tab e configurata, aggiungiamo un gruppo.

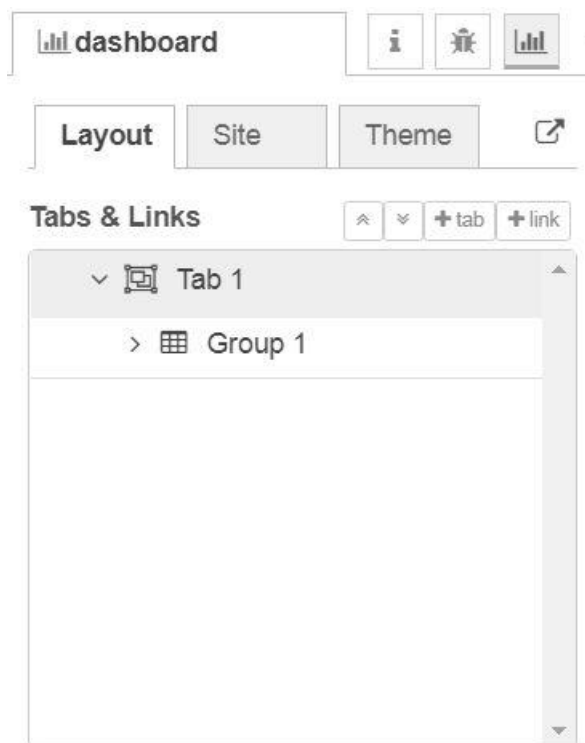


Figura 27 - Inserimento gruppo

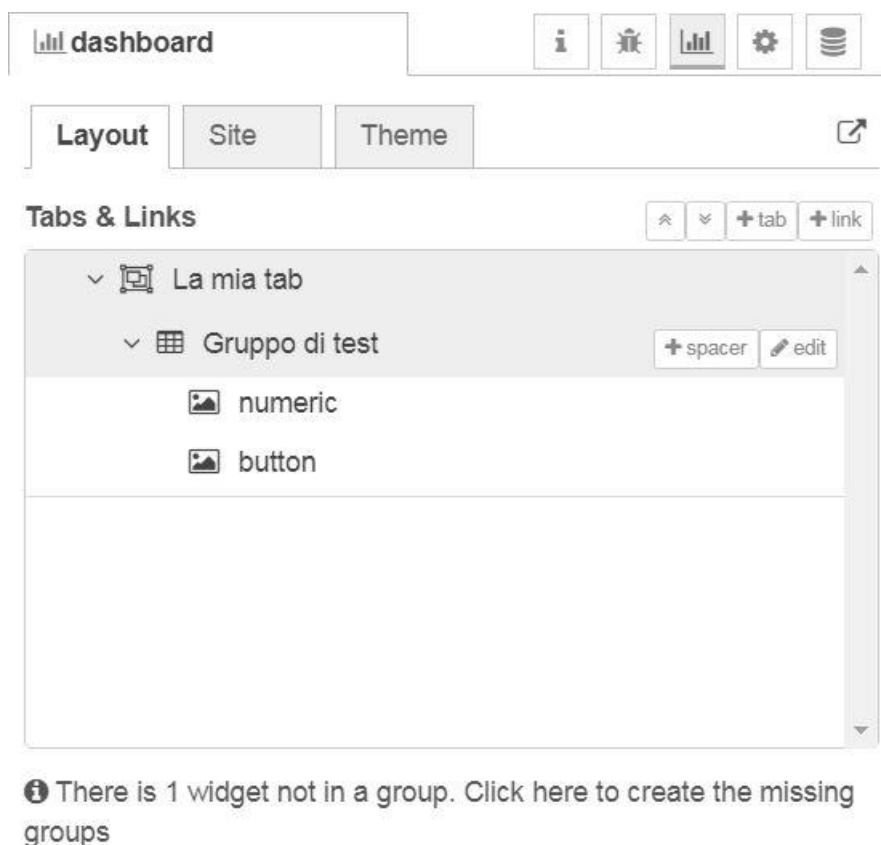


Figura 28 - Passando sopra con il mouse si vedono i pulsanti spacer ed edit

Nell'immagine si vede che ho già nominato il gruppo e i pulsanti visibili, lo spacer serve se vogliamo degli elementi vuoti nel gruppo.



Figura 29 - Configurazione Gruppo

Nella configurazione inseriamo il nome da visualizzare in grafica (in questo caso “gruppo di test”), l’assegnazione della tab (possiamo usarlo anche nel layout) la dimensione in larghezza del gruppo e le opzioni di chiudere a icona il gruppo e visualizzarne il nome.

Ho inserito anche due nodi, un numeric e un button , il risultato del layout della Figura 28 - Passando sopra con il mouse si vedono i pulsanti spacer ed edit e è visibile qua sotto.



Figura 30 - Dashboard online

Ora, se volessimo scambiare in ordine il pulsante e il campo numerico, andiamo nel layout, e con un semplice trascinamento (drag&drop) spostiamo il pulsante in alto.

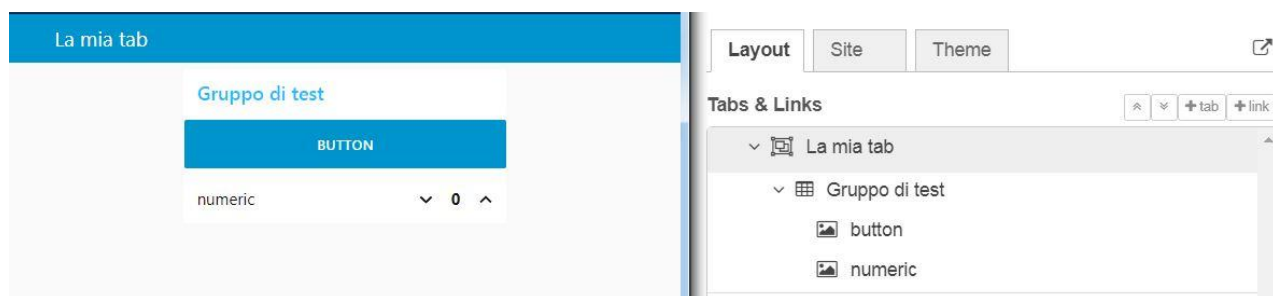


Figura 31 -Scambio posizione dei due oggetti

E se invece li vogliamo affiancati? Torniamo nell'edit del gruppo e vediamo Figura 29 - Configurazione Gruppo. Abbiamo una larghezza di default di 6. Questo significa che gli oggetti, in AUTO (valore di default), avranno una larghezza di 6.

Ora per affiancarli possiamo impostare la larghezza del button a 3 e del numeric a 3.

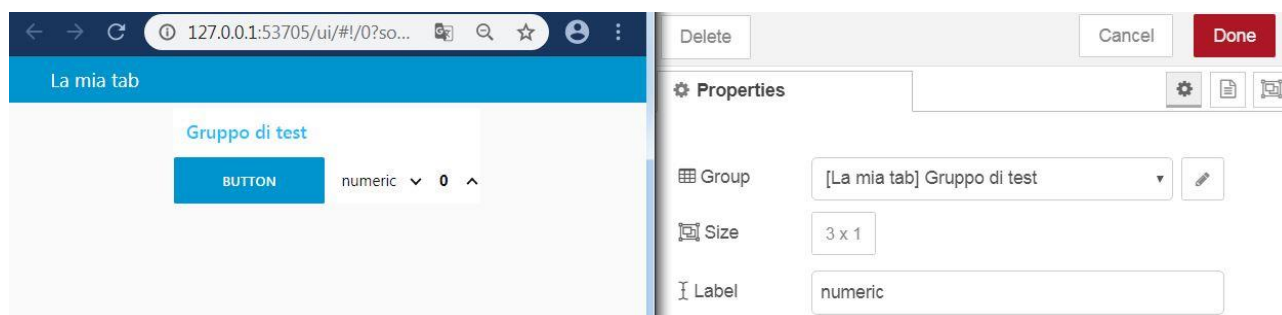


Figura 32 - Oggetti affiancati

Da questo si deduce che.

$$\text{Oggetto 1 (3)} + \text{oggetto2(3)} = \text{width del gruppo (6)}$$

Se la somma supera il valore massimo della larghezza del gruppo l'oggetto successivo sarà messo a capo.
Se aggiungiamo lo Spacer, possiamo decidere uno spazio trasparente di separazione tra oggetti.

Ad aiutarci in questo, c'è la finestra layout raggiungibile dal pulsante visibile in Figura 25 - Se passiamo sopra con il mouse vediamo i pulsanti di edit e aggiunta gruppo e layout.



Figura 33 - Layout

Oltre a essere visibile, possiamo modificare a piacimento le dimensioni degli oggetti e del gruppo.

Vediamo questo esempio, spostato solo il numeric a capo, e lo ingrandisco. Dopo che ho fatto DONE, apro il gruppo di test e vedo che in automatico ha aggiunto degli spacer, ovvero ha riempito le varie righe con quello che mancava per completare la larghezza di 6 unità.

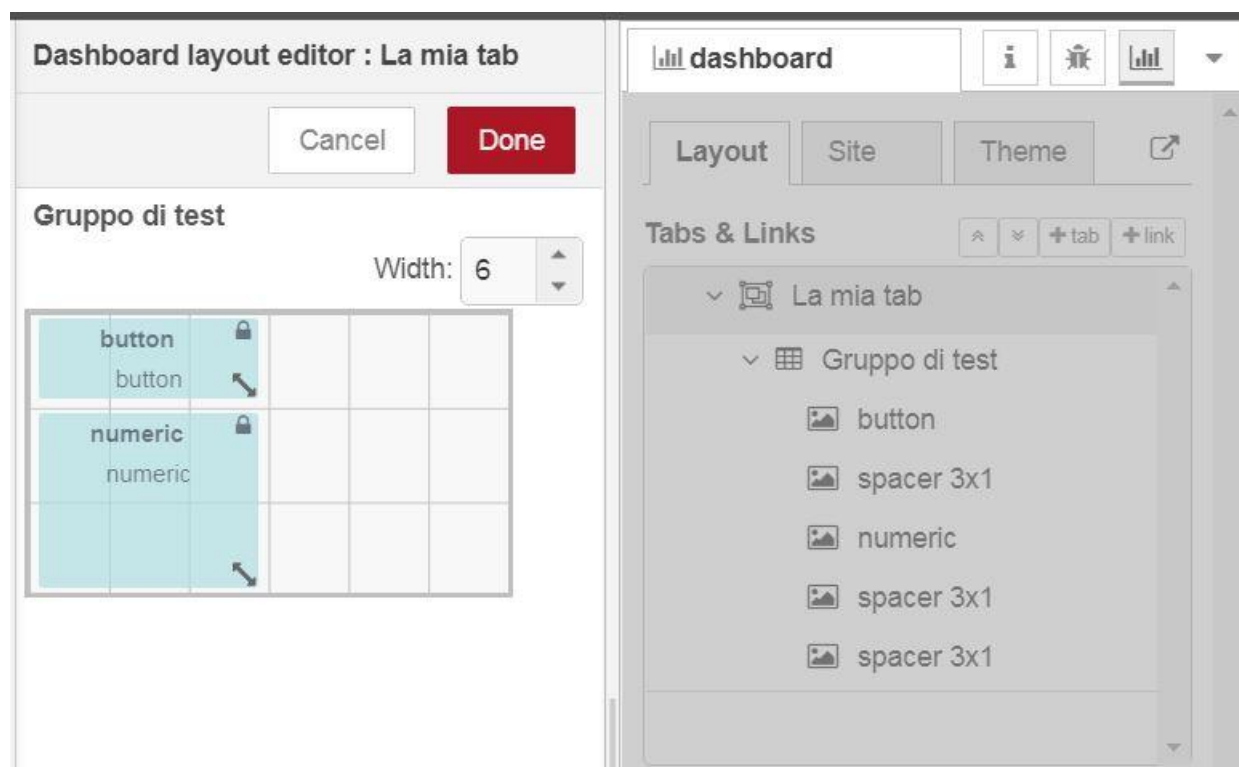
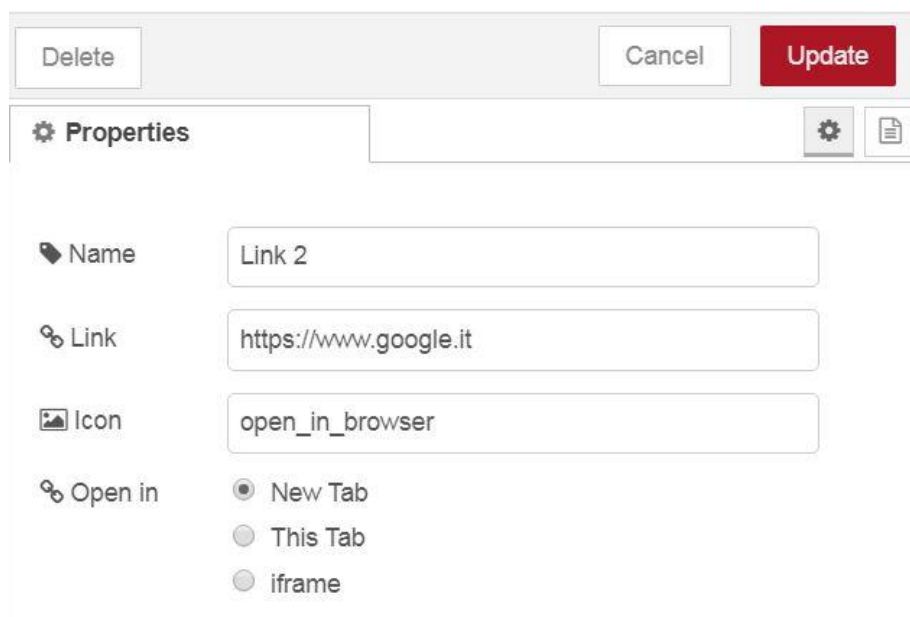


Figura 34 - Esempio layout

Questa implementazione ci permette di costruire più facilmente le nostre dashboard.

13.1.1. Link

Il pulsante Link permette di inserire un collegamento ad un URL esterno.



The screenshot shows a 'Properties' dialog box for configuring a link. At the top, there are three buttons: 'Delete', 'Cancel', and 'Update'. Below the title bar, the 'Properties' section is visible. It contains four fields: 'Name' with the value 'Link 2', 'Link' with the value 'https://www.google.it', 'Icon' with the value 'open_in_browser', and 'Open in' with three radio button options: 'New Tab' (selected), 'This Tab', and 'iframe'.

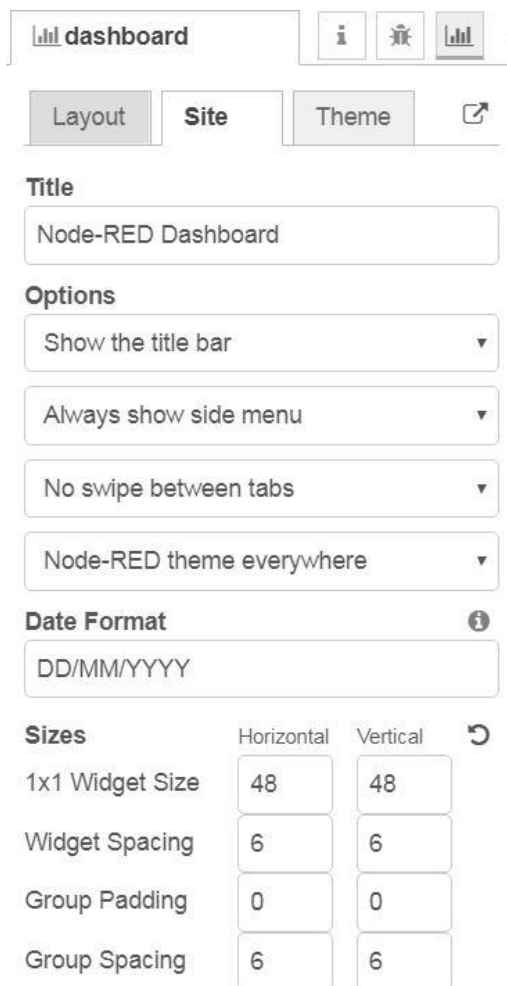
Figura 35 - Configurazione Link

Ho inserito come Link, la pagina iniziale di google. Possiamo editare il nome del link, aggiungere una icona e la modalità di apertura:

- New Tab, apre una nuova tab del browser;
- This Tab, usa la stessa tab della dashboard e apre il link;
- iFrame, apre la pagina web all'interno della dashboard.

13.2. Site

In questa tab ci sono le opzioni.



dashboard [info] [refresh] [dashboard icon]

Layout **Site** Theme [external link]

Title

Node-RED Dashboard

Options

Show the title bar ▼

Always show side menu ▼

No swipe between tabs ▼

Node-RED theme everywhere ▼

Date Format [info]

DD/MM/YYYY

Sizes	Horizontal	Vertical
1x1 Widget Size	48	48
Widget Spacing	6	6
Group Padding	0	0
Group Spacing	6	6

Figura 36 - Impostazioni dashboard

Si imposta il titolo, se mostrare la barra del titolo e menu, se il menu è a scomparsa o meno, se con lo swipe del dito o mouse si passa da una tab all'altra, il tema di node-red (ammetto che non so come funzioni), il formato data e le dimensioni e spazi.

13.1. Theme

In questa tab possiamo decidere quale tema usare.

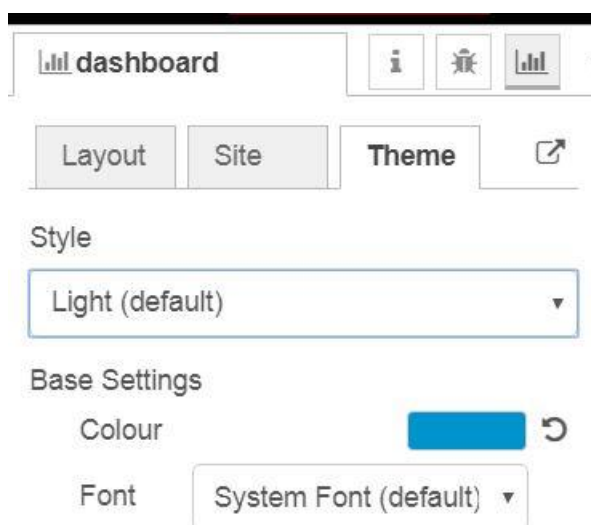


Figura 37 - Tema dashboard

14. Enciclopedia dei nodi base

14.1. Input

Inject: innietta nel flusso manualmente o ad intervalli regolari un messaggio, che può essere di vario tipo.

Catch: cattura gli errori nel flusso in cui è inserito

Status: restituisce lo stato dagli altri nodi nello stesso tab

Link: crea connessioni virtuali tra i vari flussi

MQTT: si connette ad un broker MQTT e si sottoscrive ai messaggi di uno specifico topic

HTTP: crea un endpoint HTTP per creare servizi web

WebSocket: un websocket input

TCP: Fornisce una scelta di input TCP. Può connettersi a una porta TCP remota o accettare connessioni in entrata

UDP: Un nodo di input UDP, che produce un msg.payload contenente una stringa codificata Buffer, string o base64. Supporta multicast

Serial: Legge i dati da una porta seriale locale

14.2. Output

Debug: Visualizza le proprietà del messaggio selezionate nella scheda della barra laterale di debug e facoltativamente il registro di runtime. Per impostazione predefinita visualizza msg.payload

Link: Creare cavi virtuali tra i flussi.

MQTT: Si collega a un broker MQTT e pubblica messaggi

HTTP response: Invia le risposte alle richieste ricevute da un nodo di input HTTP

WebSocket: Nodo WebSocket output

TCP: Fornisce una scelta di output TCP. Può connettersi a una porta TCP remota, accettare connessioni in entrata o rispondere ai messaggi ricevuti da un nodo TCP In

UDP: Questo nodo invia msg.payload all'host e alla porta UDP designati. Supporta multicast

Play audio: Un nodo per riprodurre l'audio nel browser. Per funzionare la pagina web dell'editor deve essere aperta

Serial: Fornisce una connessione a una porta seriale in uscita

14.3. Function

Function: Un blocco funzione JavaScript da eseguire rispetto ai messaggi ricevuti dal nodo

Template: Imposta una proprietà in base al modello fornito

Delay: Ritarda ogni messaggio che passa attraverso il nodo o limita la velocità con cui possono passare

Trigger: Quando attivato, può inviare un messaggio e quindi facoltativamente un secondo messaggio, a meno che non venga esteso o ripristinato

Comment: Un nodo che puoi usare per aggiungere commenti ai tuoi flussi

HTTP request: Invia richieste HTTP e restituisce la risposta

TCP request: Un semplice nodo di richiesta TCP: invia msg.payload a una porta TCP del server e si aspetta una risposta

Switch: Instradare i messaggi in base ai loro valori di proprietà o posizione di sequenza

Change: Imposta, modifica, elimina o sposta le proprietà di un messaggio, il contesto del flusso o il contesto globale

Range: Esegue il mapping di un valore numerico su un intervallo diverso

Split: Divide un messaggio in una sequenza di messaggi

Join: Unisce sequenze di messaggi in un singolo messaggio

Sort: Una funzione che ordina la proprietà del messaggio o una sequenza di messaggi

Batch: Crea sequenze di messaggi basati su varie regole

CSV: Converte tra una stringa formattata CSV e la sua rappresentazione di oggetto JavaScript, in entrambe le direzioni

HTML: Estrae elementi da un documento HTML contenuto in msg.payload usando un selettore CSS

JSON: Unisce sequenze di messaggi in un singolo messaggio

XML: Converte tra una stringa XML e la sua rappresentazione di oggetto JavaScript, in entrambe le direzioni

YAML: Converte tra una stringa formattata YAML e la sua rappresentazione di oggetto JavaScript, in entrambe le direzioni

Random: Genera un numero casuale tra un valore basso e alto

Smooth: Un nodo semplice per fornire varie funzioni su diversi valori precedenti, inclusi i filtri max, min, media, alta e bassa

Rbe: Segnala per nodo Eccezione – trasmette solo i dati se il carico utile è cambiato

14.4. Social

Email input: Recupera ripetutamente una singola email da un server IMAP e inoltra come msg se non è già stata vista

Email output: Invia msg.payload come email, con oggetto di msg.topic

Twitter input: Nodo di input di Twitter. Può essere utilizzato per effettuare una ricerca

Twitter output: Nodo out di Twitter. Tweetta il msg.payload

14.5. Storage

Tail: Tails (cerca le cose da aggiungere) al file configurato. (SOLO Linux / Mac)

File input: Legge il contenuto di un file come una stringa o un buffer binario

File output: Scrive msg.payload in un file, aggiungendo alla fine o sostituendo il contenuto esistente. In alternativa, può cancellare il file

14.6. Analysis

Sentiment: Analizza la proprietà scelta, di default msg.payload ed aggiunge un oggetto sentiment

14.7. Advanced

Watch: Controlla una directory o un file per le modifiche

FeedParse: Monitora un feed RSS / atom per le nuove voci

Exec: Esegue un comando di sistema e restituisce il suo output

14.8. Raspberry Pi

Rpi GPIO input: Genera un msg.payload con uno 0 o 1 a seconda dello stato del pin di input

Rpi GPIO output: Raspberry Pi nodo di uscita. Può essere utilizzato in modalità digitale o PWM

Rpi mouse: Nodo del pulsante del mouse Raspberry Pi. Richiede un mouse USB

Rpi keyboard: Nodo di gestione della tastiera Raspberry Pi. Richiede una tastiera USB

14.9. Dashboard

Button: Aggiunge un pulsante all'interfaccia utente

Dropdown: Aggiunge una casella di selezione a discesa all'interfaccia utente

Switch: Aggiunge un interruttore on/off all'interfaccia utente

Slider: Aggiunge un widget di scorrimento all'interfaccia utente

Numeric: Aggiunge un widget di input numerico all'interfaccia utente

Text input: Aggiunge un campo di immissione del testo all'interfaccia utente. La modalità può essere testo normale, email o selettore colori

Date picker: Aggiunge un widget di selezione data all'interfaccia utente

Colour picker: Aggiunge un selettore di colori alla dashboard

Form: Aggiunge un modulo all'interfaccia utente

Text: Visualizzerà un campo di testo non modificabile sull'interfaccia utente

Gauge: Aggiunge un widget di tipo indicatore all'interfaccia utente

Chart: Traccia i valori di input su un grafico. Questo 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) nella dashboard

Notification: Mostra msg.payload come notifica popup o la richiesta OK / Annulla sull'interfaccia utente

UI Control: Consente il controllo dinamico della Dashboard

Template: Il widgetTemplate può contenere qualsiasi direttiva valida in html e Angular / Angular-Material

14.10. Network

Ping: Effettua il ping su una macchina e restituisce la latenza in millisecondi come msg.payload