CONOSCERE E USARE Node-RED 2

di FRANCESCO FICILI

Iniziamo a scoprire Node-RED, un tool di flow-based programming, molto orientato all'IoT ed alla connettività, originariamente sviluppato dall'IBM Emerging Technology Services team e adesso parte della JS Foundation. In questa puntata ci addentriamo maggiormente nell'uso del tool, esaminando i nodi di utilizzo più comune, vediamo come aggiungerne di nuovi e scopriamo cosa sono i context in Node-RED. Seconda puntata.

Abbiamo iniziato questo corso introducendo, nella prima puntata il tool di flow-based programming Node-RED, descrivendone i concetti di base, quindi passando alla pratica installando l'ambiente ed analizzando le caratteristiche dell'editor grafico. Come di consueto, ci siamo quindi spostati sul lato pratico, realizzando anche un semplice esempio di utilizzo (il classico esempio di Hello World usato nei sistemi embedded, ossia il lampeggio di un LED) su Raspberry Pi. Ritenendo di avervi dato le basi per proseguire proficuamente la trattazione, in questa seconda puntata approfondiremo la nostra conoscenza dell'ambiente concentrandoci sulle principali categorie di nodi a disposizione spieghiamo come aggiungere nuovi nodi installandoli sulla palette ed introduciamo il concetto dei context, che ci permettono di immagazzinare informazioni con diversi gradi di visibilità all'interno dei nostri flow.

FLUSSI, NODI E MESSAGGI

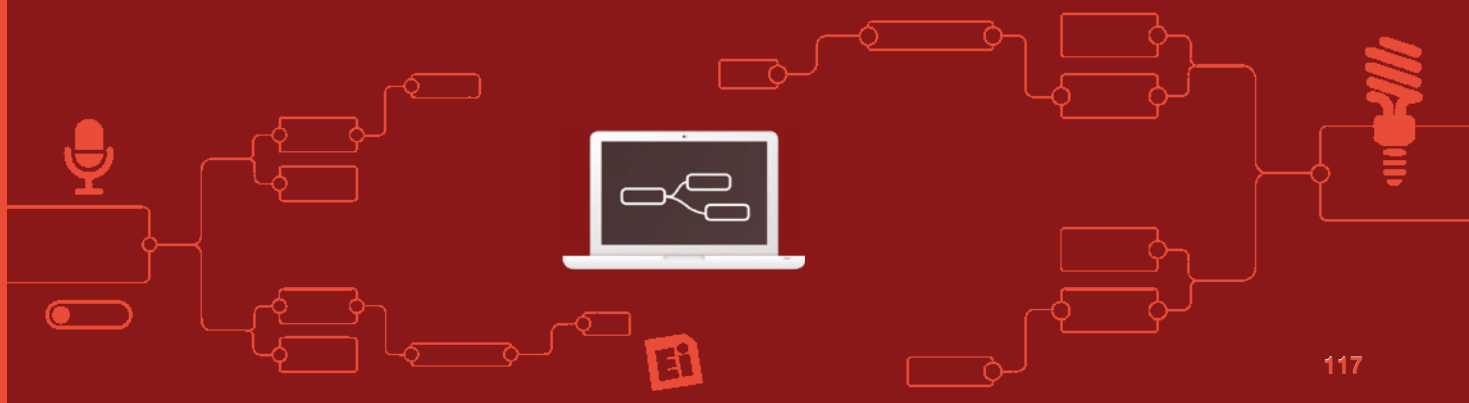
Come abbiamo brevemente accennato nella puntata precedente, gli elementi fondamentali che costituiscono un programma Node-RED sono:

- Flussi;
- Nodi;
- Messaggi.

Espandiamo questi concetti di base, in modo da comprendere meglio queste fondamentali strutture ed il loro funzionamento.

Flussi

Un flusso (o flow) è il principale meccanismo per contenere una struttura connessa di nodi. Al livello dell'editor di Node-RED è rappresentato dal workspace, all'interno del quale i nodi possono essere trascinati (prelevandoli dalla palette) e connessi tra di loro per costituire il codice grafico. Una singola istanza di Node-RED può contenere diversi flows,



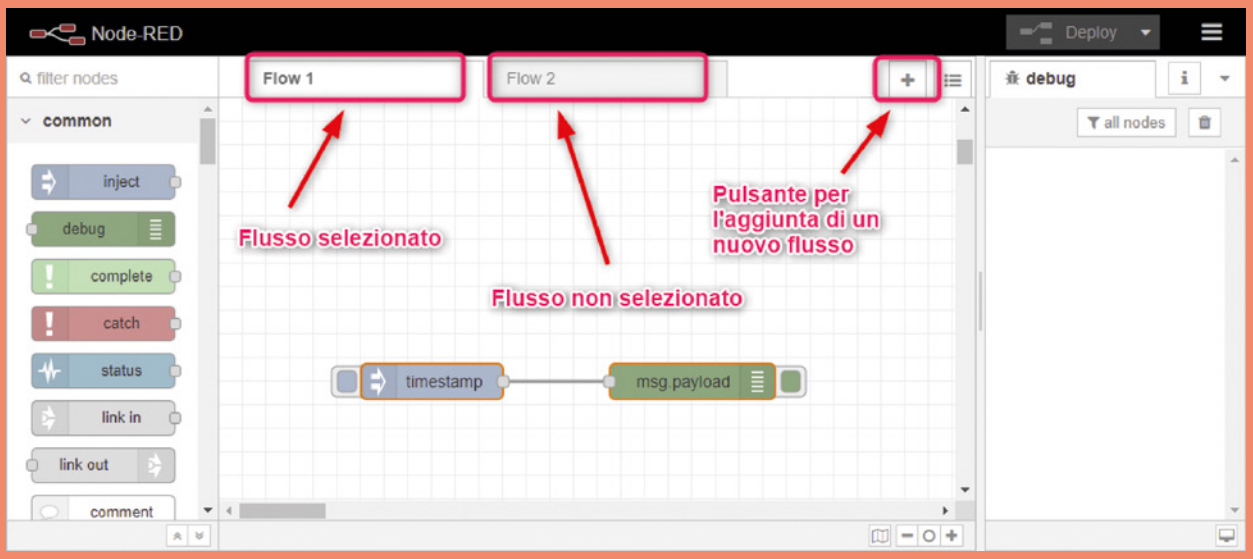


Fig. 1 - Tab flussi ed inserimento di un nuovo flusso.

rappresentati come una serie di tab nella parte alta del workspace stesso. Node-RED mostra sul workspace, di volta in volta, i nodi e le connessioni del flow attivo.
Per aggiungere un nuovo flow al workspace bisogna premere il tasto '+' situato nell'angolo in alto a destra del workspace (alternativamente può essere utilizzata l'opzione Flows → Add del menu principale), come illustrato in Fig. 1.
Il termine flusso può anche essere utilizzato per indicare informalmente un insieme atomico di nodi connessi, di conseguenza un flusso (tab) può essere costituito da un insieme di flussi (intesi come molteplici set i nodi interconnessi).

In Node-RED un flusso può essere salvato come un file JSON (Javascript Object Notation). Questo permette di memorizzare e scambiare molto facilmente i flows come file JSON. I flussi possono essere esportati ed importati tramite le due opzioni "Import" ed "Export" presenti sul menu principale (Fig. 2). Eseguendo un doppio click sul tab di un flusso si accede alla sua finestra delle proprietà, dalla quale è possibile eseguire una serie di operazioni aggiuntive, come rinominare il flusso, abilitarlo o disabilitarlo o anche cancellarlo (Fig. 3). Sempre dalla finestra delle proprietà è possibile inserire una descrizione del flusso, in modo da

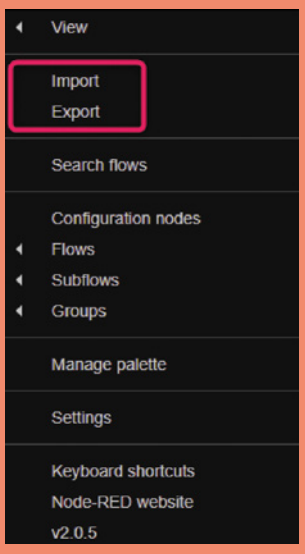


Fig. 2 - Opzioni di Import ed Export di Node-RED.

Fig. 3 - Finestra delle proprietà di un flow.

creare una utile documentazione utente del programma che stiamo sviluppando. La descrizione usa la sintassi di MarkDown e compare nella sezione info della sidebar.

Subflows

I flow possono anche essere collassati in un singolo nodo ed in questo caso si parla di subflows. I subflows sono usati per ridurre la complessità di un flow o per impacchettare un flow (inteso come gruppo di nodi) all'interno di un singolo nodo da riutilizzare più volte in altri flows. Una volta creati, i subflows appaiono nella palette come un nodo disponibile e possono essere aggiunti al workspace come i nodi standard.

Nodi

Un flow Node-RED è popolato da Nodi (l'elemento principale di Node-RED) e relative connessioni. I nodi costituiscono le parti eseguibili di un flow e si scambiano messaggi tramite wires (fili) che vengono collegati alle porte di ingresso e di uscita di un nodo. I nodi sono triggerati sia ricevendo un messaggio su una porta d'ingresso o tramite un evento interno (come ad esempio lo scadere di un timer, il cambio di stato di una line digitale, o una richiesta proveniente da un canale di comunicazione). Quando viene triggerato, il nodo elabora il messaggio ricevuto o l'evento interno, e successivamente, può inviare un messaggio ai nodi successivi del flow. In qualche caso i nodi hanno degli elementi integrati nella loro icona, con i quali l'utente può interagire direttamente dal flow. Questi elementi possono essere pulsanti, elementi di output, eccetera.
Ci sono tre modi per aggiungere un nodo ad un flow:

- Prelevandolo dal Node Palette
- Tramite la quick-add dialog box

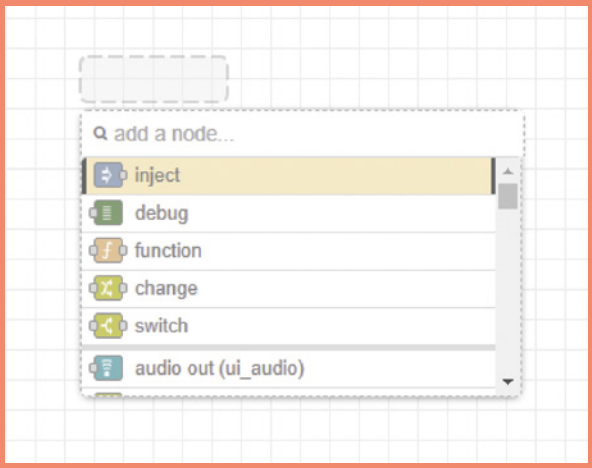


Fig. 4 - Quick-add dialog box.

- Tramite la funzione Import

La prima opzione è quella più comune: si seleziona il nodo dalla Node Palette premendo e mantenendo premuto il pulsante sinistro del mouse e lo si trascina sul workspace, rilasciando il pulsante per posizionare il nodo.
Per utilizzare la quick-add dialog box invece è sufficiente tenere premuto il tasto Ctrl e cliccare con il tasto sinistro all'interno del workspace. Questo farà comparire una finestra di selezione come quella riportata in Fig. 4 (la quick-add dialog box, appunto...) dalla quale è possibile selezionare il nodo da aggiungere. L'opzione di import infine permette di importare un nodo (o più comunemente un intero flow)

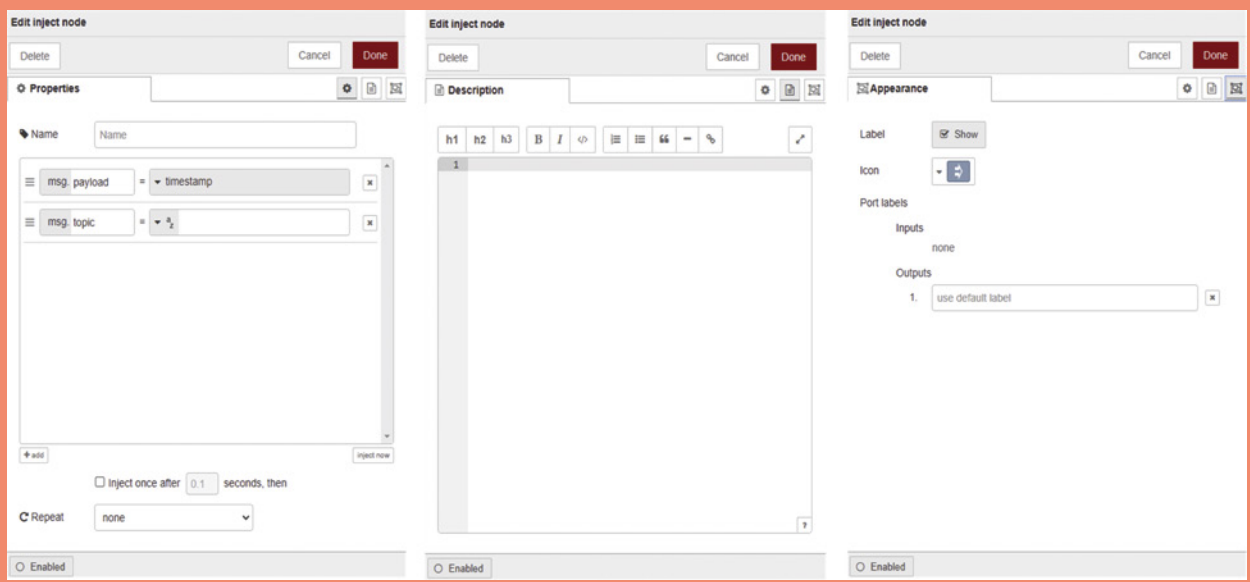


Fig. 5 - Finestra delle proprietà (sezioni Properties, Description ed Appearence) per un nodo Inject.

precedentemente salvato sottoforma di file JSON (tramite la funzione duale export). Esattamente come i flows, anche i nodi dispongono di una finestra delle proprietà, alla quale si può accedere eseguendo un doppio click sul nodo specifico (o alternativamente premendo il tasto invio quando il nodo ha il focus). La finestra delle proprietà di un nodo è divisa in tre sezioni principali, che descriviamo qui di seguito.

- **Properties**: da questa sezione è possibile editare le proprietà specifiche del nodo in questione. Questa sezione varia da nodo a nodo, poiché nodi differenti hanno differenti opzioni di configurazione e proprietà.

- **Description**: da questa sezione è possibile editare la documentazione del nodo, formattata con sintassi MarkDown. La descrizione inserita in questa sezione comparirà nella sezione info della sidebar, come documentazione del nodo. Esattamente come per i flows, questa funzionalità permette di creare una documentazione utente molto ben fatta dei nostri programmi.

- **Appearance**: da questa sezione è possibile customizzare l'aspetto grafico del nodo. Ad esempio è possibile definire se visualizzare o meno la label del nodo, è possibile selezionarne l'icona (nel caso non si voglia usare quella di default) ed inoltre è possibile definire delle label specifiche per ogni porta.

La Fig. 5 rappresenta le tre sezioni Properties, Description ed Appearence della finestra delle proprietà per un nodo Inject.

Configuration Nodes

Non sempre i nodi sono visibili sul workspace, infatti esiste un tipo particolare di nodo, chiamato configuration node, che non ha una parte grafica (icona e porte di connessione), ma costituisce un insieme di configurazioni che possono essere riutilizzate da più nodi in un flow. I nodi che necessitano un configuration node per funzionare ne permettono la creazione direttamente dalla loro finestra delle proprietà. Ad esempio i nodi serial in e serial out, che necessitano di un configuration node serial port (che contiene le opzioni di configurazione della porta seriale, che possono essere utilizzate da diverse istanze dei nodi di comunicazione seriale), ne permettono la creazione (o, se il nodo è già esistente, l'associazione) direttamente dalla loro finestra delle proprietà, come visibile in Fig. 6 e Fig. 7. Non avendo parte grafica, la finestra delle proprietà dei configuration nodes è costituita dalla sola parte Properties. Una lista completa dei configuration nodes presenti può essere ricavata dalla sezione configuration nodes della sidebar.

Messaggi

Durante il funzionamento di un flow, i vari nodi che lo compongono si scambiano messaggi, e questa è una ca-

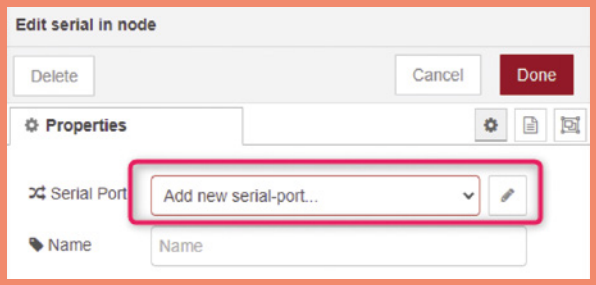


Fig. 6 - Creazione del configuration node serial port dalla finestra di proprità del nodo serial in.

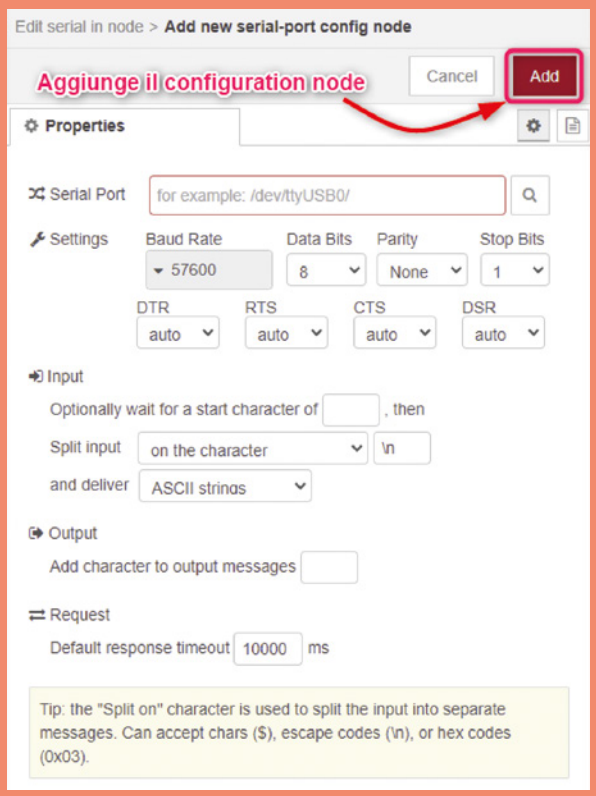


Fig. 7 - Finestra delle proprietà del configuration node serial port.

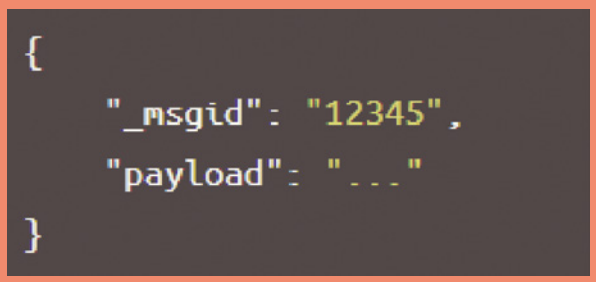


Fig. 8 - Esempio di messaggio Node-RED.

ratteristica peculiare di Node-RED e della FBP. I messaggi che i nodi si cambiano sono semplici oggetti JavaScript che possono avere qualunque set di proprietà.
Tipicamente una proprietà denominata "payload" è sempre presente, ed è la proprietà con cui più frequentemente i nodi lavorano, ma possono essercene molte altre.
Node-RED inoltre aggiunge ad ogni messaggio una proprietà "_msgid", che sostanzialmente identifica in maniera univoca il messaggio all'interno di un flow (Fig. 8).
Il valore di una proprietà di un messaggio può essere un qualsiasi tipo JavaScript, ad esempio:

- Booleano – true/false
- Numerico – 1, 123.4
- Stringa – "Hello World"
- Array – [1,2,3,4,5,6,7,8,9]
- Object – {"a" : 1, "b" : 2}
- NULL

CORE NODES

Come abbiamo visto in precedenza, l'elemento principale di un flow Node-RED sono i nodi. Node-RED contiene una ricca dotazione di nodi predefiniti e moltissimi altri nodi possono essere aggiunti alla palette.
Per ragioni di spazio è impossibile descrivere nel dettaglio tutti i nodi a disposizione nell'ecosistema di Node-RED, nonché anche poco utile in pratica, dato che si tratta di un ambiente in continua evoluzione.
Esiste tuttavia un set di nodi standard, a cui ci si riferisce spesso con il termine "Core Nodes" che è importante conoscere (Fig. 9). Questi nodi sono:

- Inject;
- Debug;
- Function;
- Change;
- Switch;
- Template.

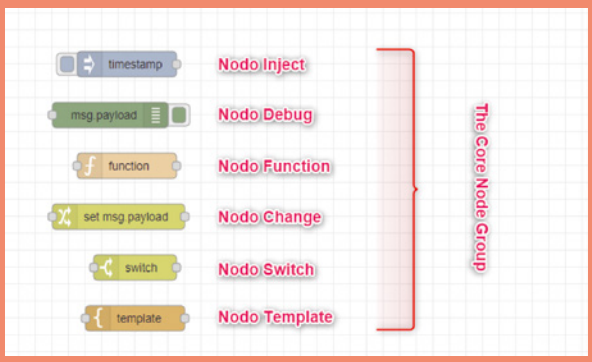


Fig. 9 - I Core Nodes.

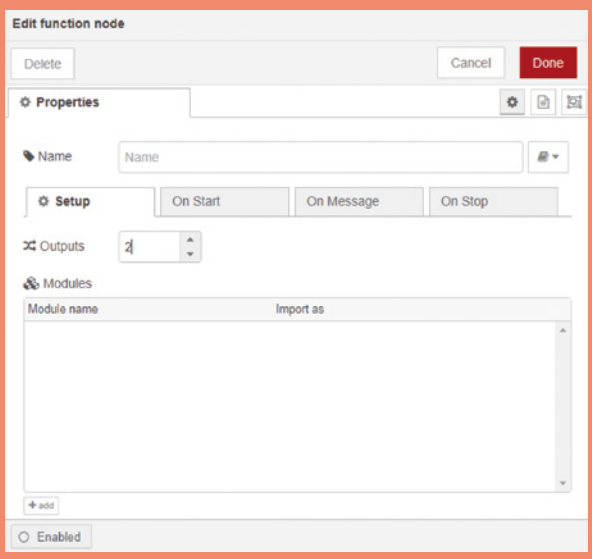


Fig. 10 - Finestra delle proprietà del nodo Function.

Qui di seguito li descriviamo uno per uno.

Inject

Il nodo Inject permette di inviare messaggi all'interno di un flow, sia manualmente (interagendo con il pulsante integrato nell'icona del nodo), che automaticamente. Il messaggio inviato può avere qualsiasi proprietà si desideri e le proprietà possono essere settate in vari modi. Ad esempio una proprietà di un messaggio del nodo inject può essere:

- uno dei tipi base supportati da JavaScript (ad esempio un boolean, un valore numerico, una stringa, un array...);
- un timestamp;
- una stringa JSON;
- una espressione JSONata.

In questo modo possono essere iniettati nel nostro flow messaggi contenenti qualsiasi dato.
Di conseguenza il nodo inject è molto utilizzato per il debug e per testare il funzionamento di altri nodi, pur trovando anche altri interessanti utilizzi (ad esempio per l'inject periodico di valori all'interno del flow, come abbiamo visto nell'esempio pratico della scorsa puntata).

Debug

Il nodo Debug permette di visualizzare i messaggi che riceve dalla sua porta in ingresso nella visuale di debug della sidebar.
La finestra di debug della sidebar permette di visualizzare in maniera strutturata le informazioni contenute nel messaggio, rendendo quindi il nodo di debug molto utile per il debugging dei flow (come era anche facilmente intuibile dal suo nome).

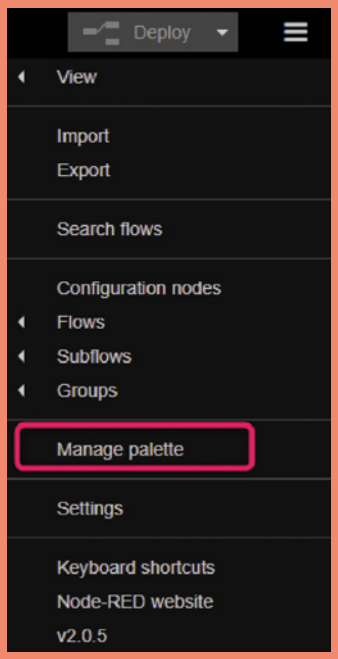


Fig. 11 - Opzione Manage Palette tra le opzioni di Node-RED.

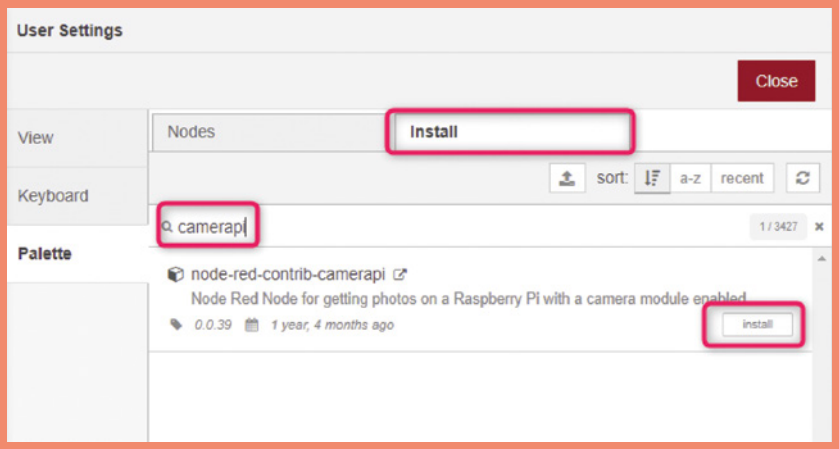


Fig. 12 - Installazione del nodo "node-red-contrib-camerapi".

Function

Il nodo Function (Fig. 10) permette di scrivere del codice JavaScript che agisce direttamente sul messaggio passato al nodo stesso.
Il messaggio è passato in un oggetto chiamato msg, e possono essere creati quanti output si vuole.
Dalle versioni più recenti di Node-RED è possibile importare moduli JavaScript da utilizzare all'interno del blocco ed è inoltre possibile invocare codice JavaScript all'avvio ed allo stop del nodo (ad esempio allo scopo di inizializzare delle variabili di stato e via di seguito).

Change

Il nodo Change è utilizzato per modificare una proprietà di un messaggio senza dover ricorrere al nodo function.
Ogni nodo può essere configurato per eseguire una serie di operazioni in ordine.

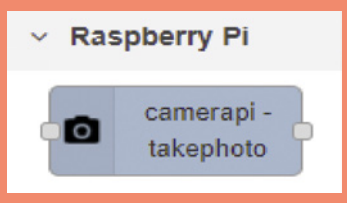


Fig. 13 - Nodo camerapi installato.

Le operazioni disponibili sono:

- **Set**: settare una proprietà ad un certo valore. Il valore può essere una varietà di tipi differenti, o può essere preso ad un'altra proprietà.
- **Change**: cercare e rimpiazzare parte di una proprietà di un messaggio.
- **Move**: muovere o rinominare una proprietà.
- **Delete**: cancellare una proprietà

Switch

Il nodo Switch permette di indirizzare messaggi verso differenti porte in uscita, in base alla valutazione di regole interne al nodo. A tale riguardo va precisato che sono previsti i quattro tipi di regola descritti qui di seguito.

- **Valore**: Il valore di una specifica proprietà può essere controllato con espressioni comparative o booleane (ad esempio maggiore di, minore di, uguale, vero falso, etc..).
- **Sequenza**: la sequenza di arrivo dei messaggi può essere controllata, estraendo solo alcuni elementi della sequenza stessa, dalla testa, dalla coda o da un numero arbitrario di elementi intermedi.
  Vengono triggerate le porte di uscita per cui si verifica un match rispetto al controllo di sequenza.
- **JSONata**: il messaggio viene controllato per mezzo di una espressione JSONata (viene fornito un apposito editor per impostare e testare la query) e viene triggerato lo specifico output solo se l'espressione ritorna true.
- **Otherwise**: questo è il classico caso di default delle espressioni switch dei linguaggi classici, viene triggerato se nessuna delle precedenti regole viene soddisfatta.

Template

Il nodo Template, infine, è un nodo che è in grado di genera-

Javascript Object Notation (JSON)

Il formato JSON (JavaScript Object Notation) è un formato di interscambio molto utilizzato in applicazioni client/server. È basato sul linguaggio JavaScript Standard ECMA-262 3ª edizione (dicembre 1999), ma ne è indipendente, trattandosi di un formato completamente indipendente dallo specifico linguaggio di programmazione utilizzato, sebbene le convenzioni che utilizza siano familiari alla famiglia dei linguaggi basati sul C.
La semplicità di JSON ne ha decretato un rapido utilizzo specialmente nella programmazione in AJAX.
Il suo uso tramite JavaScript è particolarmente semplice, infatti l'interprete JavaScript è in grado di eseguirne il parsing tramite la funzione JSON.parse().
Questo lo ha reso velocemente molto popolare a causa della diffusione della programmazione in JavaScript nel mondo del Web.
La maggior parte dei linguaggi di programmazione possiede un typesystem molto simile a quello definito da JSON per cui sono nati molti progetti che permettono l'utilizzo di JSON con altri linguaggi quali, per esempio: ActionScript, C, C#, Adobe ColdFusion, Common LISP, Delphi, E, Erlang, Java, JavaScript, Lua, ML, Objective Caml, Perl, PHP, Python, REBOL e Ruby.

re un output testuale utilizzando le proprietà dei messaggi per riempire un template usando la sintassi del Mustache template language (maggiori informazioni sul Mustache template language possono essere reperite a questo idirizzo web: *https://mustache.github.io/mustache.5.html*).
Questo nodo è molto utile per generare testi formattati in modi differenti partendo da tags contenuti su delle message properties.
Ad esempio, un nodo template con all'interno questa dicitura:

```
Il messaggio di testo contenuto nel payload è: {{payload}}
```

A fronte della ricezione di un messaggio contenente nella property payload la stringa "Ciao Mondo!!!", genererebbe il seguente output testuale:

```
Il messaggio di testo contenuto nel payload è: Ciao Mondo!!!
```

che andrà pertanto a sostituire al tag `{{payload}}` il contenuto di msg.payload.

AGGIUNGERE NODI ALLA PALETTE

Oltre a questi nodi di base, Node-RED contiene diversi altri nodi pre-installati di default, aventi interessanti funzioni aggiuntive, ma che non possono essere descritti uno per uno per ovvie ragioni di spazio.
Ma non è tutto, infatti la natura fortemente open dell'ambiente e la sua vasta community hanno portato alla produzione di una quantità di nodi aggiuntivi, che possono essere installati. Sarebbe assolutamente impossibile descriverli tutti, trattandosi di diverse migliaia di nodi, mentre può essere utile descrivere brevemente la procedura per aggiungere dei nodi alla palette che contiene quelli installati per impostazione predefinita.
L'aggiunta di un nuovo nodo può essere fatta selezionando l'opzione "Manage Palette" presente nelle opzioni di Node-RED (Fig. 11).
Una volta cliccato sull'opzione si aprirà una finestra di dialogo, ed in questa finestra bisogna selezionare il tab "Install" e poi cercare il nodo desiderato usando l'apposito motore di ricerca integrato.
Supponiamo, ad esempio, di voler installare un nodo per la gestione della camera pi (*https://www.raspberrypi.org/products/camera-module-v2*).
Per farlo selezioniamo l'opzione "Manage Palette", e nella finestra di dialogo che ci viene presentata selezioniamo il tab install e scriviamo "camerapi" nell'apposita textbox per la ricerca, come illustrato in Fig. 12.
Selezionando il pulsante "Install" di uno dei risultati della ricerca, Node-RED procederà all'installazione del nodo. Nel caso dell'esempio appena visto, ossia l'installazione del nodo camerapi, al termine dell'operazione troveremo il nodo di Fig. 13 tra i nodi della palette.

COSA SONO I CONTEXT

Passiamo adesso all'ultimo argomento di questa puntata, ossia i Context: Node-RED prevede un sistema per lo stoccaggio di informazioni in memoria che possono essere utilizzate da diversi nodi oppure da differenti istanze di esecuzione di un nodo stesso: si tratta dei context.
Ebbene, esistono tre diversi tipi di context in Node-RED, cui sono associate tre differenti visibilità (o scope):

- Node: visibile solo da uno specifico nodo;
- Flow: visibile all'interno di uno specifico flow;
- Global: visibile globalmente da parte di tutti i nodi di Node-RED.

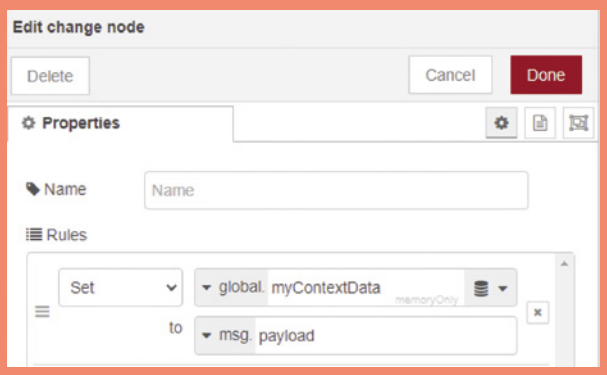


Fig. 14 - Esempio di utilizzo di un context con Change node.

Il tipo specifico di scope da utilizzare dipende da quello che si desidera fare con il context: se, a titolo di esempio, il valore di un determinato dato deve essere visibile solo all'interno di un function node, allora è sufficiente il Node Context. Se invece il dato deve essere visibile a più nodi, allora sarà necessario utilizzare un Flow Context oppure un Global Context. I context vengono salvati in RAM di default, quindi si comportano in maniera simile a delle variabili globali, ma tuttavia non sono persistenti ad un riavvio di Node-RED. Esiste comunque la possibilità di cambiare lo store type dei context per far si che il loro valore venga salvato sul file system anziché in RAM. I context possono essere manipolati in diversi modi. Il modo più semplice è quello di utilizzare un change node, ad esempio la regola in Fig. 14 salva il contenuto presente in msg.payload sul context globale myContextData.
Esistono inoltre delle API che permettono di manipolare (leggere e scrivere) i context da un function node. Abbiamo 3 tipologie di accesso per ognuno dei 3 tipi di context, ed ognuna di queste può accedere a metodi di set e get:

- Context: node context;
- Flow: flow context;
- Global: global context.

Ad esempio, per leggere il valore myContexData con node scope da un function node possiamo scrivere la seguente riga di codice all'interno di un nodo function:

```
var myData = context.get("myContextData");
```

Invece per scrivere il valore della nostra variabile locale, una volta che è stato aggiornato, sul context, possiamo utilizzare la seguente istruzione:

```
context.set("myContextData", myData);
```

I due esempi appena proposti utilizzano il node scope (context), ma sono validi anche per il flow e global context, basterebbe cambiare la keyword context appunto con flow o global.

ESEMPIO PRATICO

Passiamo adesso all'esempio pratico di questa puntata. Ciò che vogliamo fare è nuovamente far lampeggiare un LED, ma stavolta applicando alcuni dei nuovi concetti esposti in questa puntata.
Utilizzeremo ancora una volta un nodo inject per creare la temporizzazione ed il medesimo nodo rpi-gpio-out per controllare la linea di uscita, ma stavolta, al posto del nodo trigger, faremo uso di un function node, all'interno del quale scriveremo un po' di codice javascript per generare la logica necessaria al lampeggio ed utilizzeremo un context per immagazzinare il valore booleano corrispondente allo stato del LED tra un'attivazione del nodo function e la successiva.
Iniziamo inserendo sul workspace i nodi necessari al nostro progetto, ossia:

- un nodo inject;
- un nodo function;
- un nodo rpi-gpio-out.

e collegandoli poi tra di loro logicamente come proposto nella Fig. 15.
I nodi inject e rpi-gpio-out devono essere configurati esattamente come nell'esempio pratico che vi abbiamo proposto nella puntata precedente del corso, così come rimane

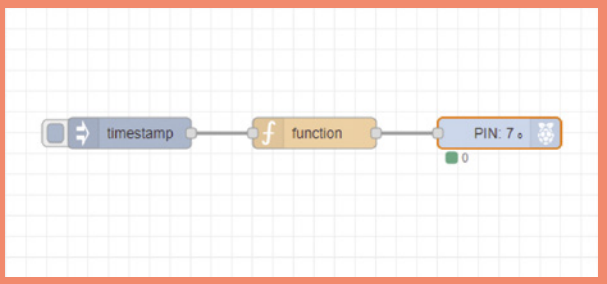


Fig. 15 - Flow per Led Blink con nodo function.

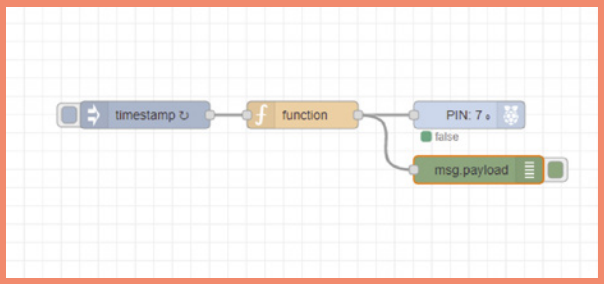


Fig. 16 - Inserimento del nodo Debug.

Listato 1 - Codice javascript per il LED blink

```
// Create the variable Led an load it with the context
var Led = context.get("LED");
// Invert the variable value
Led = !Led;
// Store back the inverted value into the context
context.set("LED", Led);
// Associate the payload with the varible value (Led status)
msg.payload = Led;

return msg;
```

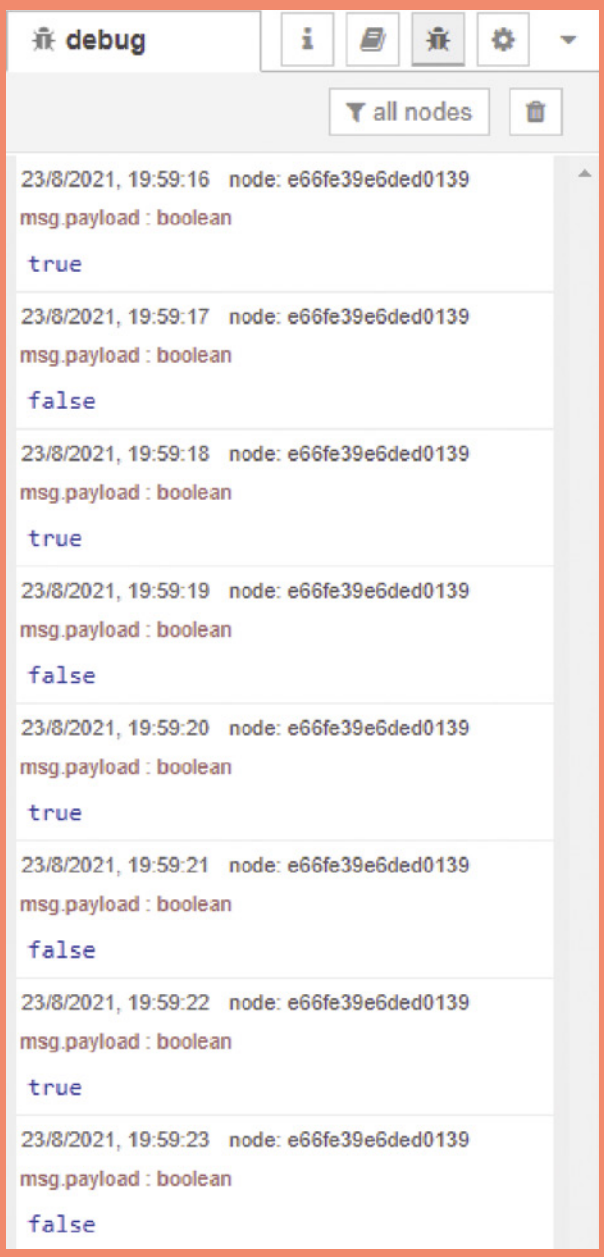


Fig. 17 - Debug Log.

la stessa la configurazione hardware da utilizzare (il pin al quale è collegato il LED è il GPIO4).
Il nodo inject ha un'attivazione periodica e genera un messaggio ogni secondo; il contenuto di tale messaggio è irrilevante, in quanto non viene utilizzato dal nodo function, ma serve soltanto ad attivarlo.
Ad ogni attivazione, il nodo function deve generare in uscita un messaggio contenente sulla property payload lo stato del pin (e quindi del LED); questo viene realizzato tramite il codice javascript riportato nel Listato 1.
Come è possibile vedere da tale listato, viene creata una variabile Led, che viene caricata con il contenuto del context LED mediante l'istruzione `context.get(LED)`. Il context ha uno scope di tipo node, dal momento che non deve essere utilizzato da nessun altro nodo del flow al di fuori del nodo function stesso.
Di fatto questo context funziona come una variabile globale che immagazzina lo stato del LED tra una invocazione del nodo function e la successiva.
Nella riga successiva lo stato della variabile Led (appena caricato) viene invertito, allo scopo di generare il lampeggio desiderato.
A questo punto il nuovo stato viene salvato sul context mediante l'istruzione `context.set("LED", Led);` fatto questo, non resta altro da fare che copiare il valore della variabile Led su msg.payload e "ritornare" il messaggio in uscita.
Se lanciamo il flow in esecuzione mediante il pulsante deploy, utilizzando lo stesso setup dell'esempio pratico che vi abbiamo proposto nella prima scorsa puntata, vedremo il medesimo risultato, ossia il LED collegato al GPIO4 lampeggiare alla frequenza di 1Hz.
Se colleghiamo un nodo debug all'uscita del nodo function (Fig. 16) vedremo anche apparire un log che illustra il risultato pratico del codice che abbiamo inserito nel nodo, ossia una successione di messaggi con alternanza dello stato booleano tra true e false (che va a pilotare la linea digitale 4 della scheda Rapsberry Pi), come visibile in Fig. 17.

CONCLUSIONI

In questa seconda puntata del corso abbiamo approfondito la nostra conoscenza di Node-RED, soffermandoci maggiormente sui suoi concetti fondamentali, come i flow, i nodi e i messaggi. Abbiamo inoltre analizzato alcuni nodi fondamentali installati per impostazione predefinita con l'ambiente (si tratta dei cosiddetti Core Nodes), visto come aggiungere nodi alla palette ed introdotto i context.
Infine abbiamo concluso proponendo il solito esempio pratico, allo scopo di farvi "prendere la mano" con i concetti esposti e farveli fissare più facilmente nella mente .
Nella prossima puntata inizieremo a concentrare la nostra attenzione sull'interazione con il mondo esterno, quindi vedremo come fare per controllare alcune periferiche della scheda Raspberry PI utilizzando le funzionalità offerte da Node-RED.