CONOSCERE E USARE Node-RED 7

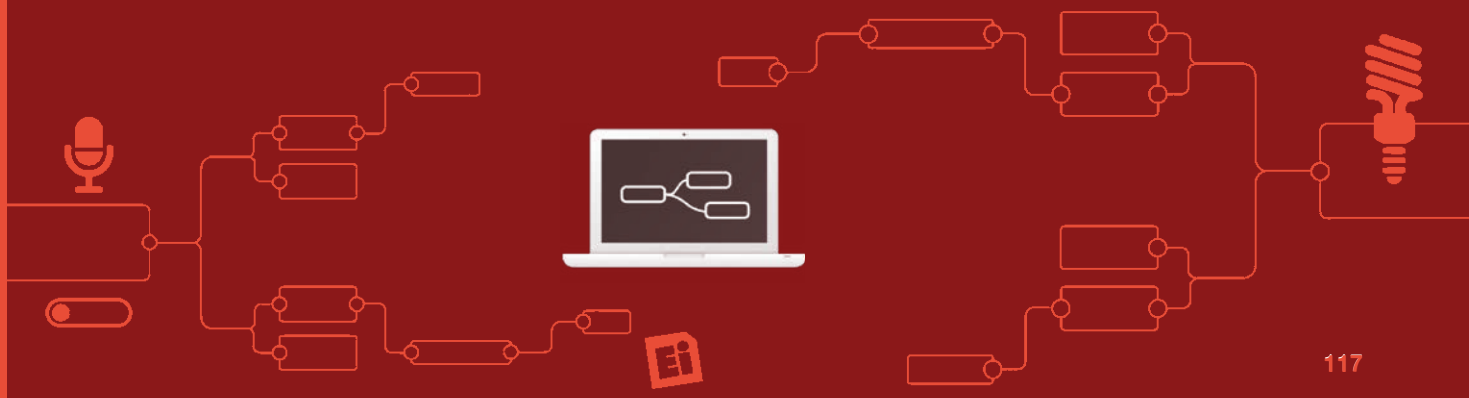
di FRANCESCO FICILI

Impariamo ad usare Node-RED, un tool di flow-based programming orientato all'IoT ed alla connettività, originariamente sviluppato dall'IBM Emerging Technology Services team e adesso parte della JS Foundation. In questa puntata parliamo dell'integrazione di funzionalità di messaggistica, sia tradizionale usando le email, che istantanea, facendo uso di applicazioni come Whatsapp e Telegram. Settima Puntata.

Abbiamo più volte posto l'accento, nelle puntate precedenti, sul fatto che Node-RED sia un ambiente con una forte predisposizione all'implementazione di applicazioni di Internet of Things e di come disponga di una serie di estensioni che ne facilitano di molto l'utilizzo in questi ambiti. Al giorno d'oggi, qualsiasi applicazione IoT che si rispetti non può più prescindere da alcune funzionalità fondamentali che ne permettano l'integrazione con delle applicazioni che siamo sempre di più abituati ad utilizzare quotidianamente sia con i nostri PC che con i nostri smartphone: le applicazioni di messaggistica. Noi siamo tipicamente abituati ad utilizzare questa tipologia di applicazioni per la comunicazione tra le persone, ma di fatto le applicazioni di messaggistica, grazie alle loro caratteristiche intrinseche di affidabilità, robustezza e semplicità d'uso, costituiscono anche un ottima interfaccia per la comunicazione uomo-macchina. Non potendo trattare, per ragioni di spazio, l'integrazione di Node-RED con tutte le applicazioni di messaggistica esistenti, tra cui si annoverano, oltre ai capostipiti WhatsApp e Telegram, anche Facebook Messenger, Viber, Signal, solo per citarne alcune, nella puntata odierna ci focalizzeremo esclusivamente su:

- Invio e ricezione di email tramite Node-RED
- Integrazione di Node-RED con WhatsApp
- Integrazione di Node-RED con Telegram

I campi di applicazione sono molteplici: la messaggistica ci consente di interagire con Node-RED usando, ad esempio Telegram o WhastApp direttamente dal nostro smartphone, permettendoci quindi di controllare attuatori o acquisire dei dati usando queste applicazioni.



INVIARE EMAIL TRAMITE NODE-RED

Uno dei metodi più comuni per inviare messaggi è quello di utilizzare le email, e prima della comparsa delle applicazioni di messaggistica istantanea era anche uno dei più diffusi, oltre ai classici SMS, oramai caduti completamente in disuso perché soppiantati proprio dall'instant messaging, più economico e pratico. Le email tuttavia continuano ad essere un sistema interessante per scambiare informazioni tra un dispositivo ed una persona. Node-RED non fornisce nativamente nodi per la gestione dei protocolli di invio e ricezione di email, ma esistono svariati nodi tramite i quali è possibile aggiungere questo supporto.

Noi proponiamo il modulo node-red-node-email, che, come abbiamo visto già diverse altre volte, può essere installato direttamente dalla palette, utilizzando il nome del nodo stesso come chiave di ricerca.

Questa libreria fornisce l'accesso a 3 nodi specifici (Fig. 1), che sono:

Email MTA: Mail Transfer Agent.

Questo nodo è un mail trasfer agent di test, che in questa puntata non tratteremo.

Email Send: questo nodo permette di inviare il contenuto del *msg.payload* come email. Il contenuto di *msg.topic* può essere utilizzato per impostare l'oggetto del messaggio.

Email Receive: questo nodo controlla periodicamente se sono state ricevute email e se ne trova di nuove invia un messaggio in uscita che contiene l'oggetto della mail su *msg.topic* e il body su *msg.payload*.

L'utilizzo di questi nodi è molto semplice, per inviare una email possiamo usare il nodo Email Send, del quale dovremo opportunamente configurare la finestra delle proprietà. Questo set di nodi non dispone di *configuration node*, quindi le configurazioni sono locali alla specifica istanza. Le opzioni da configurare sono (si veda anche Fig. 2, in cui

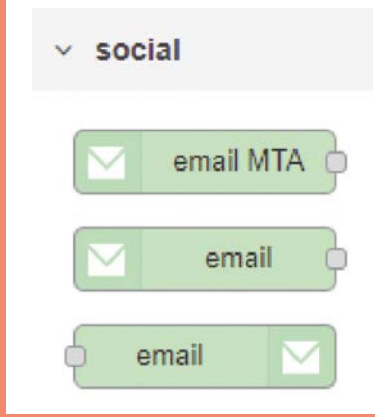


Fig. 1 - Nodi email Node-RED.

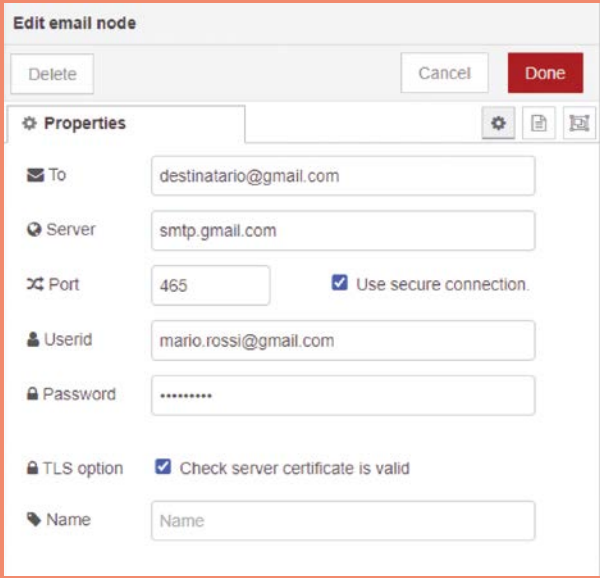


Fig. 2 - Esempio di configurazione del nodo email send.

è stato fatto un esempio con degli account non reali):

To: identifica l'email del destinatario (nell'esempio la mail d*estinatario@gmail.com*).

Server: identifica il server di mail in uscita da utilizzare (di default *smtp.gmail.com*, ma può essere modificato con qualsiasi altro server smtp).

Port: identifica la porta da utilizzare (di default la porta 465, ma può essere modificata in cui si utilizzi un server di posta diverso da quello di default).

UserId: la user ID dell'account email dal quale si desidera inviare la mail (in questo caso la mail d'esempio *mario.rossi@gmail.com*).

Password: la password dell'account.

Use secure connection e TLS option: opzioni di sicurezza, entrambe abilitate di default.

Una volta configurato il nodo, per inviare una email, sarà sufficiente inviare sulla porta di ingresso del nodo un messaggio in cui il *msg.topic* costituisce l'oggetto del messaggio stesso e il *msg.payload* il suo contenuto. Per gli utenti gmail sarà necessario abilitare l'accesso da app meno sicure dai settings dell'account di google per consentire al nodo di funzionare, oppure abilitare una application password.

Vediamo invece adesso come gestire le email in ricezione. In questo caso il nodo da utilizzare è naturalmente Email Receive e le opzioni da configurare nella sua finestra delle proprietà sono le seguenti (si veda anche Fig. 3):

Get Mail: questa opzione permette di decidere come si vuole interrogare la casella di posta elettronica. E' offerta sia la possibilità di farlo quando il nodo riceve un

messaggio in ingresso (ed in questo caso viene creata una opportuna porta), oppure automaticamente dopo un determinato intervallo di tempo (every X seconds).
Protocol: identifica il protocollo utilizzato (IMAP o PO3)
Use SSL e Start TLS: opzioni di sicurezza (di default l'SSL è attivato, mentre non viene eseguito il TLS).
Server: l'indirizzo del server da utilizzare (di default *imap.gamil.com*, ma modificabile)
Port: identifica la porta da utilizzare (di default la porta 993, ma può essere modificata nel caso in cui si utilizzi un server di posta diverso da quello di default)
UserId: la user ID dell'account email dal quale si desidera ricevere la mail (in questo caso la mail d'esempio *mario.rossi@gmail.com*).
Password: la password dell'account.
Folder: identifica il folder di posta da controllare (di default è il folder INBOX, ma potrebbe eventualmente essere modificato per controllare un folder differente).
Disposition: permette di decidere cosa fare del messaggio che viene letto (nulla, segnarlo come letto o cancellarlo).
Criteria: permette di determinare la politica con la quale i messaggi vengono inviati in uscita (ossia se inviare quelli non letti, quelli senza risposta, etc...).

Come nel caso del nodo *send*, il *msg.topic* del messaggio di output contiene l'oggetto del messaggio, mentre il *msg.payload* ne contiene il testo. Sono presenti anche informazioni aggiuntive come la data di ricezione ed il mittente, presenti su altri campi del messaggio in uscita (Fig. 4).

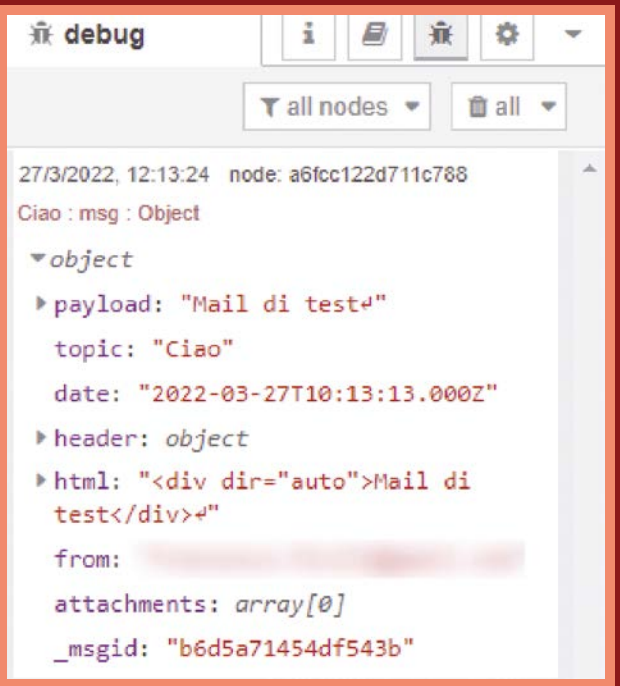


Fig. 4 - Esempio di complete msg object inviato dal nodo Email receive quando viene ricevuto un nuovo messaggio di posta.

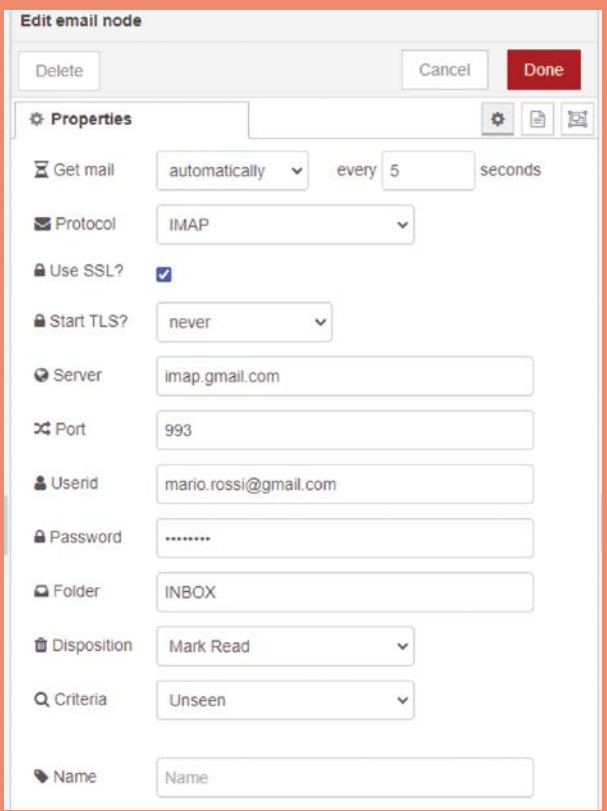


Fig. 3 - Esempio di configurazione del nodo email receive.

INTEGRAZIONE CON WHATSAPP

Dopo aver visto come gestire le email passiamo all'integrazione con la prima app di messaggistica istantanea: WhatsApp. Ci sono diverse possibilità per integrare Node-RED con WhatsApp, noi segnaliamo la libreria *node-red-contrib-whin*, che ci è sembrata la più completa tra quelle attualmente disponibili e anche quella con le migliori perfor-

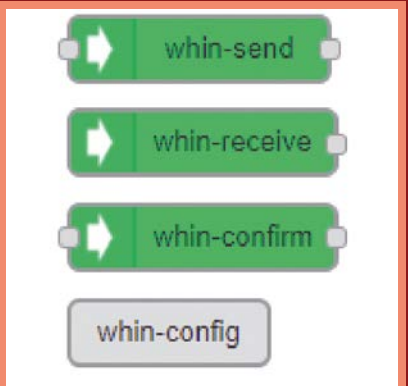


Fig. 5 - Nodi della libreria Whin.

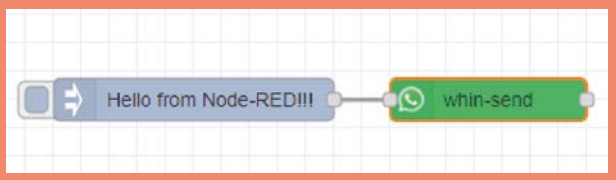


Fig. 6 - Flow di test del nodo whin-send.

mance. Per installarla, come di consueto, usiamo l'opzione "Manage Palette" del menù principale e cerchiamo la nostra libreria.
Una volta installata noterete che la librerie Whin mette a disposizione 3 nodi (si veda anche Fig. 5):
whin - send: permette di inviare un messaggio whatsApp al numero indicato nella configurazione.
whin – receive: permette di ricevere messaggi whatsApp dal numero indicato nella configurazione.
whin – confirm: permette di inviare una richiesta con risposta "chiusa" (Yes o No) al numero indicato nella configurazione.

Oltre a questi tre nodi è presente anche il classico *configuration node whin-config* per le configurazioni generali.
Il set di nodi della libreria Whin usa uno shared gateway per la comunicazione, ed è quindi necessario registrarsi al servizio inviando uno specifico messaggio ad un numero, che successivamente rilascerà una API key da utilizzarsi per accedere alla chat.
I dettagli per la registrazione sono indicati nella sezione "Set-up and Usage" della seguente pagina web: *https://flows.nodered.org/node/node-red-contrib-whin.*
Il numero utilizzato per la registrazione è anche quello da cui riceveremo e a cui invieremo i messaggi (potete registrarlo in rubrica con il nome che preferite).
Una volta ottenuto il token possiamo posizionare un nodo **whin – send** sul workspace e procedere al setup del configuration node. Facciamo quindi doppio click sul nodo **whin – send** e poi clicchiamo sull'icona a forma di matita per accedere alla finestra delle proprietà del *configuration node*. Qui inseriamo un nome per la configurazione (nell'esempio Whin – Test) e poi il numero di telefono dal quale abbiamo fatto la registrazione al servizio e il Token ottenuto in precedenza. E' importante ricordare che il numero di telefono va inserito con il relativo country code, ma senza il carattere '+' all'inizio e che il numero di telefono va inserito senza simboli '-' o spazi all'interno.
Una volta configurato correttamente il config node possiamo salvare ed eseguire il deploy.
Per testare il nodo possiamo usare un semplice nodo inject contenente una stringa qualsiasi (ad esempio "Hello from Node-RED!!!") collegata alla porta di ingresso del nodo **whin – send**, come illustrato nel flow di Fig. 6.
Una volta cliccato sul pulsante del nodo inject il messaggio verrà inviato sulla chat whatsapp che abbiamo con il numero tramite il quale abbiamo registrato il servizio.

INTEGRAZIONE CON TELEGRAM

Passiamo ora all'integrazione di node-RED con un'altra diffusissima applicazione di messaggistica istantanea: Telegram.
Anche in questo caso, esistono diverse librerie in Node-RED che permettono l'interfacciamento con Telegram; quella che noi proponiamo in questa puntata è la *node-red-contrib-telegrambot*, che fa uso di un bot telegram.
La libreria fornisce i seguenti nodi, riportati anche in Fig. 7:
Telegram Receiver: questo nodo invia un messaggio di output quando dei messaggi sono ricevuti nella chat. I dettagli del messaggio sono contenuti nei seguenti campi di msg.payload:

- Content: il contenuto del messaggio
- Type: la tipologia di dati
- messageId: il numero che rappresenta l'ID del messaggio
- chatId: il numero che rappresenta l'ID della chat

Telegram Command: questo nodo invia un messaggio quando uno specifico comando è ricevuto dal bot sulla chat. Esempi di comando sono **/On** o **/Temperature**.
Telegram Event: questo nodo invia un messaggio quando uno specifico evento è ricevuto dal bot sulla chat
Telegram Sender: questo nono invia un messaggio nella chat quando riceve un messaggio internamente. Il contenuto di *msg.payload.content* viene scritto sulla chat. Deve inoltre essere trasmesso la corretta chat ID, contenuta nel campo *msg.payload.chatId*.
Telegram Reply: questo nodo invia un messaggio quando nella chat viene inviata una risposta ad un messaggio specifico.

Fig. 7 - Nodi della libreria Telegrambot.

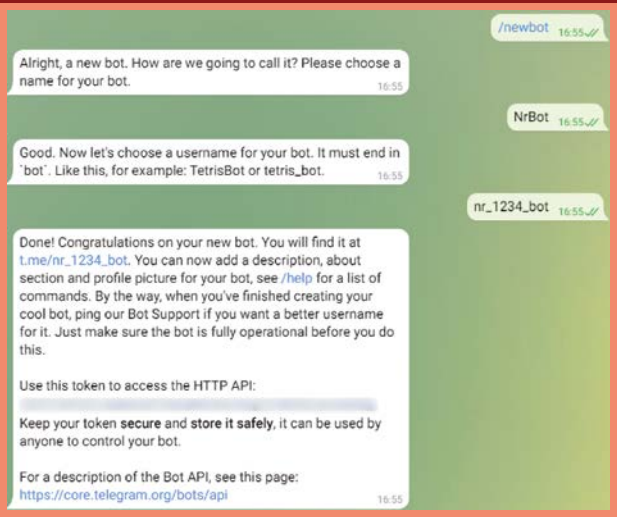


Fig. 8 - Creazione di un nuovo bot.

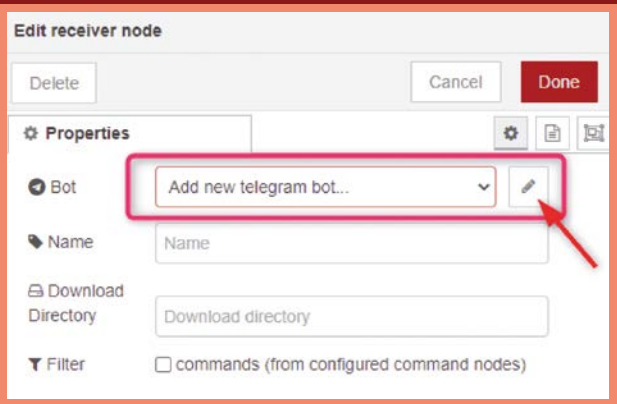


Fig. 9 - Accesso alla finestra delle proprietà del configuration node Telegram Bot.

Oltre a questi nodi è presente anche un *configuration node* chiamato **Telegram Bot** che permette di impostare alcune proprietà del bot con cui si desidera comunicare (come ad esempio la API Key) ed altri dettagli della comunicazione.
Prima di utilizzare la libreria è necessario creare un bot telegram, in modo da recuperare la API Key necessaria per collegarsi al bot.
Per creare un bot su Telegram è necesario richiederne la creazione ad uno speciale bot, denominato BotFather (può essere trovato cercando *@BotFather* sulla barra di ricerca di telegram). BotFather ha uno specifico comando, */newbot*, che può essere utilizzato per creare un nuovo bot. Una volta inviato questo comando bisogna seguire le istruzioni del BotFather e dare al bot un nome confidenziale (può essere qualsiasi cosa, nel nostro esempio è NrBot) e poi uno username, che invece deve essere univoco e terminare con suffisso "bot". Qui potranno essere necessari alcuni tentativi e il nome finale potrebbe essere stravagante, in quanto esistono molti bot Telegram e quindi ci sono molti username.
BotFather vi indicherà pazientemente se lo username che avete selezionato è già in uso, fino a quando non ne troverete uno. Trovato il nome giusto BotFather vi fornirà l'API key necessaria per comunicare con il bot.
Un esempio di creazione di un bot è riportato in **Fig. 8**.
A questo punto va recuperata la chatId; per farlo possiamo semplicemente posizionare sul workspace un nuovo Telegram receiver ed impostare le proprietà del configuration node Telegram bot. Clicchiamo quindi 2 volte sul nodo Telegram receiver che abbiamo posizionato sul workspace e poi sull'icona a forma di matita presente a fianco del campo Bot, come illustrato in **Fig. 9**. A questo punto possiamo configurare le proprietà del nostro Telegram Bot, inserendo le due stringhe salvate in precedenza, ossia lo username del bot e la API key, come illustrato in **Fig. 10**.
In questa finestra delle proprietà è anche possibile impostare degli altri parametri relativi alla gestione del bot, ma per questo primo semplice esempio possiamo mantenere i valori di default.
Possiamo ora cercare il nostro bot su Telegram (sempre tramite lo username), avviarlo ed inviare un semplice messaggio al nostro nodo *Telegram Receiver* su Node-RED.

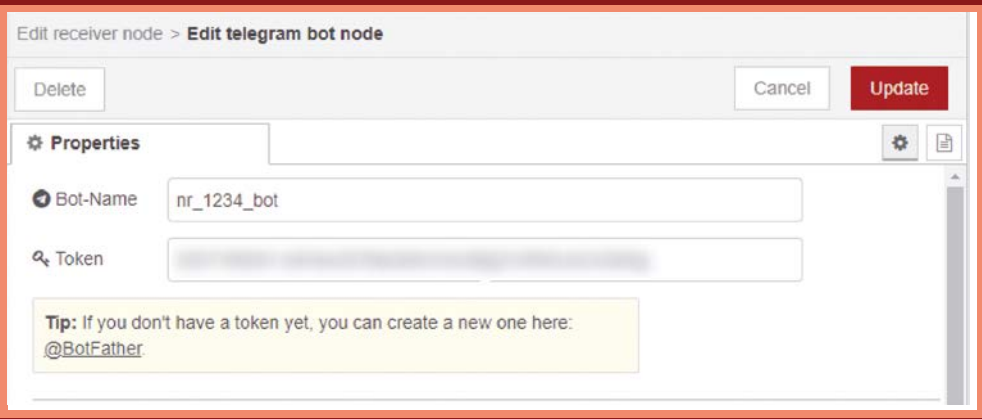


Fig. 10 - Inserimento dello username e della API key nella finestra delle proprietà del configuration Node Telegram Bot.

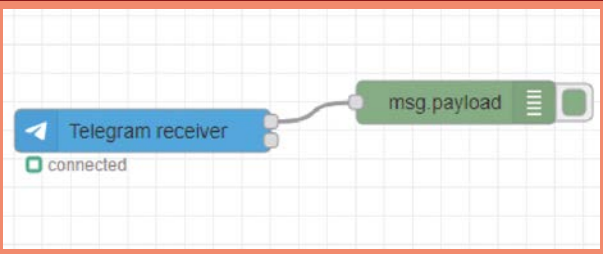


Fig. 11 - Ricezione di un messaggio per ottenere la ChatId.

Se intercettiamo l'output del nodo con un nodo debug (come evidenziato in Fig. 11), otterremo un output simile a quello di Fig. 12, dal quale possiamo ricavare la chatID, che ci servirà per la parte di trasmissione.

ESEMPIO PRATICO: CONTROLLARE UN LED TRAMITE TELEGRAM

A questo punto abbiamo acquisito sufficienti informazioni per implementare un semplice progetto pratico in cui controlleremo un LED attraverso Telegram, facendo in modo che ad un opportuno set di comandi scritto sulla chat con il nostro bot il LED venga accesso e spento. Faremo anche in modo che il nostro flow Node-RED restituisca anche indietro una stringa che indica lo stato del LED.

Per implementare questo semplice flow ci occorrono:

- 2 nodi Telegram Command
- 2 nodi Change
- 1 nodo Rpi Gpio out
- 1 nodo Function
- 1 nodo Telegram sender

Fig. 12 - Log del messaggio ricevuto tramite nodo Telegram receiver.

Iniziamo con la parte del flow dedicata alla ricezione, posizionando i due nodi Telegram Command e i due nodi change ed il nodo rpi gpio out come illustrato in Fig. 13. Il nodo rpi gpio out deve controllare il Pin che collegheremo al LED. Se usiamo nuovamente lo shield Rpi I/O (codice FT1060) di Futura Elettronica, possiamo connettere questo nodo al GPIO 12 che è, a sua volta, collegato al primo LED.

A questo punto configuriamo la finestra delle proprietà dei due nodi Telegram Command, impostando come Bot il bot creato in precedenza e come comando in un caso /ledon (per accendere il LED) e nell'altro */ledoff* (per spegnerlo).

Il resto delle opzioni di questi due nodi può essere lasciato al valore di default. In Fig. 14 è rappresentata la finestra delle proprietà del nodo che riceve il comando di accensione del LED. Passiamo adesso ai nodi change, che ci occorrono per modificare il valore del campo payload per adattarlo a quello necessario ad accendere e spegnere il LED tramite il nodo rpi gpio out.

I due nodi devono avere ciascuno una semplice regola che setta il *msg.payload* a true in corrispondenza del comando */ledon* ed a false in corrispondenza di */ledoff*.

In Fig. 15 è riportato l'esempio sempre nel caso di accensione del LED.

A questo punto la parte di ricezione ed attuazione del comando è completa e possiamo già eseguire un primo test inviando sulla chat del nostro bot i comandi */ledon* e */ledoff* per accendere e spegnere il LED.

Tuttavia ci manda ancora la ricezione del feedback sulla

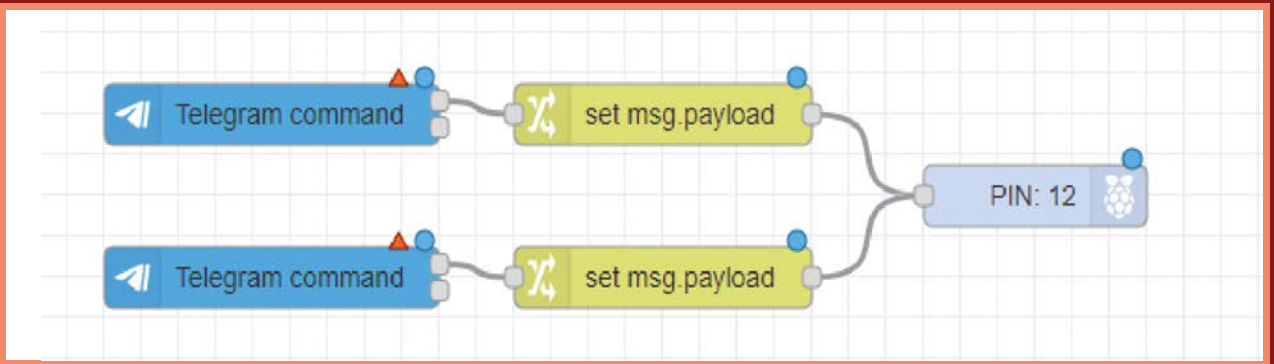
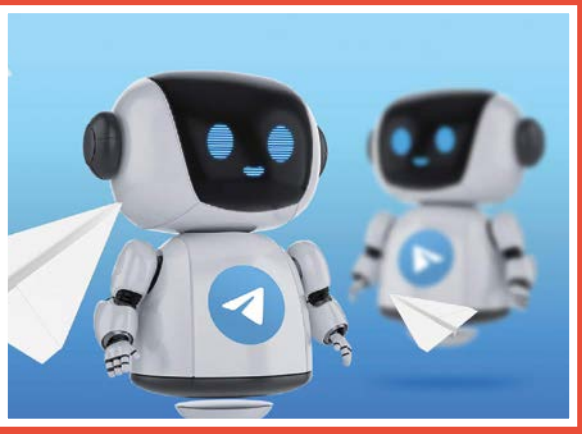


Fig. 13 - Posizionamento dei nodi per la ricezione ed attuazione del comando.

I bot di Telegram



In questa puntata abbiamo introdotto l'integrazione di Node-RED con applicazioni di messaggistica istantanea, tra cui Telegram, e di conseguenza di siamo imbattuti in una delle caratteristiche peculiari di questa applicazione: i 'bot'. Ma cosa sono precisamente i bot di Telegram? I bot, come il nome suggerisce, non sono altro che degli utenti artificiali di Telegram (dei programmi quindi) che possono interagire sia nelle chat singole che nelle chat di gruppo. Un bot è facilmente riconoscibile in telegram, in quanto il suo username termina sempre con il suffisso 'bot'. Questi programmi interagiscono con gli utenti essenzialmente rispondendo a dei comandi, che possono essere impartiti tramite il carattere speciale '/'. Tipicamente ogni bot può listare la sua lista di comandi rispondendo al comando /help. Nel corso degli anni sono stati sviluppati molteplici bot per Telegram, una lista completa e divisa per categorie che può essere consultata è lo StoreBot (*https://storebot.me/bots/*). Alcuni bot sono fondamentali per il funzionamento dell'ecosistema, come il BotFather (*@botfather*), che permette di registrare nuovi bot, o il TelegramStoreBot (*@store_bot*), che funge da motore di ricerca per i bot e permette di visualizzare, direttamente nella chat di Telegram, l'elenco completo dei bot presenti sulla piattaforma. Altri bot sono invece simili a programmi di utilità, che possono quindi semplificarci alcuni aspetti della nostra vita quotidiana, come ad esempio TrackBot (*@TrackBot*), un bot che permette di tenere sotto controllo le nostre spedizioni, offrendo un servizio di monitoraggio dei corrieri italiani e stranieri, o TranslateBot (*@Translate_Bot*), che permette di tradurre istantaneamente delle parole scritte all'interno della chat. Infine esistono anche bot pensati per l'intrattenimento, come EnigmiBot (*@enigmibot*), un bot che sottopone degli enigmi, oppure 'culturali' come DottorBot (*@dottor_bot*), un bot dedicato alla cultura tecnico-scientifica.

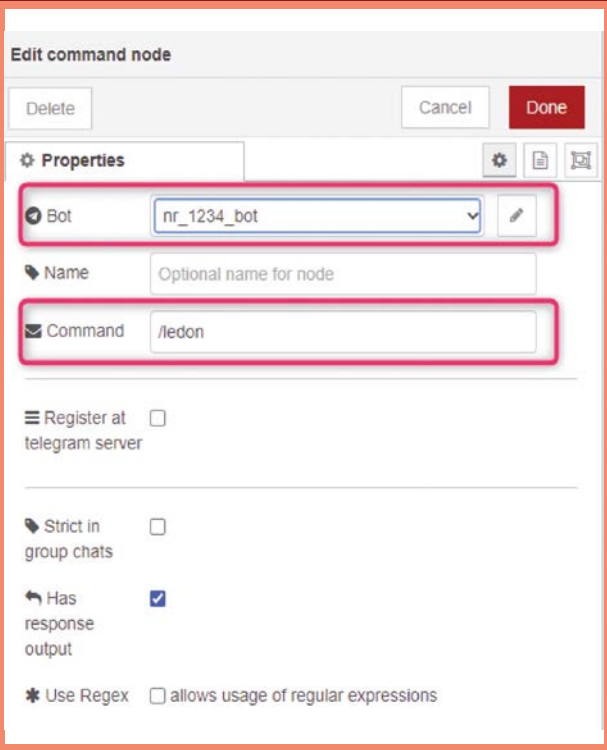


Fig. 14 - Finestra delle proprietà del nodo Telegram command per l'implementazione del comando di accensione.

chat, che ci indica che il comando è stato ricevuto ed eseguito. Aggiungiamo quindi il nodo function ed il nodo Telegram Sender al flow e colleghiamoli come illustrato in Fig. 16. Per fornire correttamente la risposta dobbiamo prima di tutto configurare il nodo Telegram Sender, referenziando il solito configuration node che contiene la configurazione del nostro bot selezionandola dal campo bot della finestra delle proprietà del nodo. Poi dobbiamo inserire un piccolo snippet di codice javascript nel function node, in quanto il nodo sender si aspetta di ricevere un messaggio con un oggetto javascript con una specifica formattazione, contenente, all'interno del campo msg.payload:

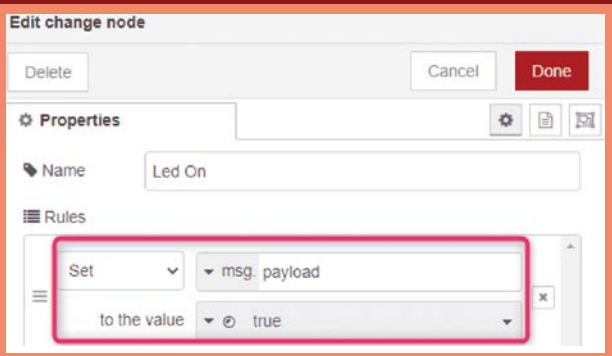


Fig. 15 - Finestra delle proprietà del nodo change per la modifica del valore di msg.payload.

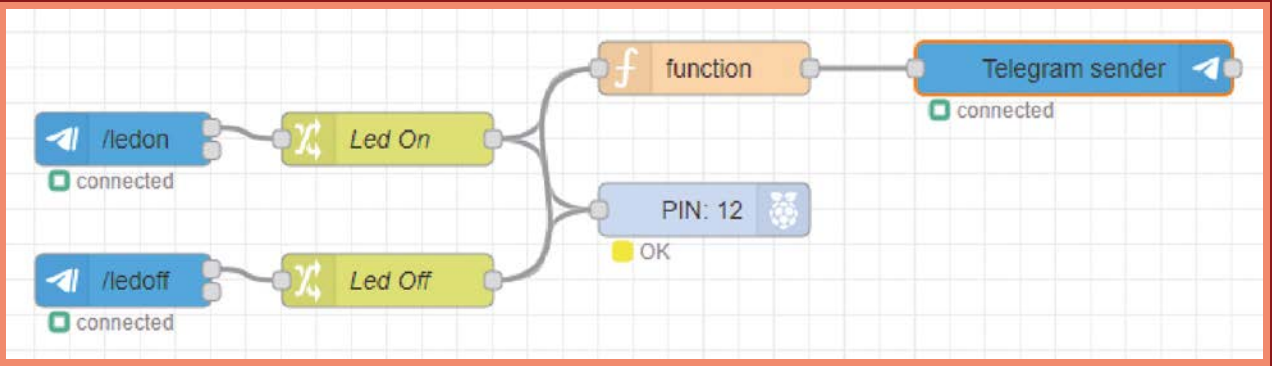


Fig. 16 - Inserimento della risposta.

Listato 1 – Snippet di codice Javascript da inserire nel nodo Function

```
if (msg.payload == true)
{
    msg.payload = {
        content : "The Led is ON",
        type : "message",
        chatId : 738459487
    };
}
else
{
    msg.payload = {
        content : "The Led is OFF",
        type : "message",
        chatId : 738459487
    };
}

return msg;
```

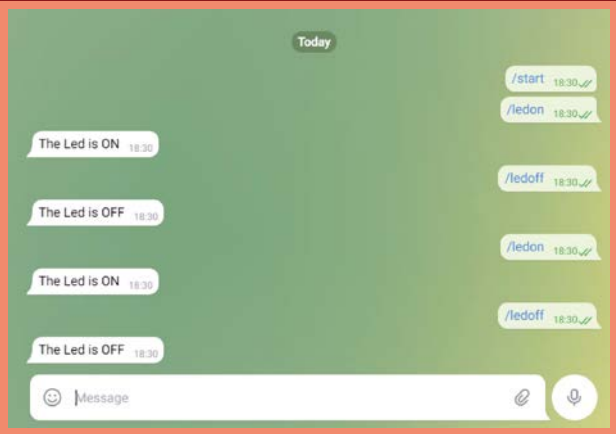


Fig. 17 - Test finale del Flow.

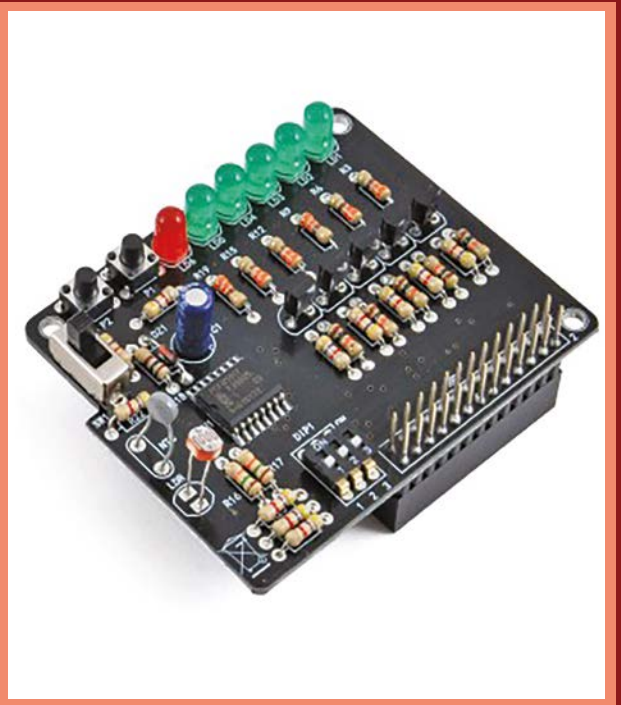


Fig. 18 - Shield Raspberry Pi I/O di Futura Elettronica.

- un campo content (string), che rappresenta il contenuto del messaggio;
- un campo type (string), che rappresenta la tipologia di messaggio, nel nostro caso useremo la stringa "message" che identifica un semplice messaggio di testo;
- un campo chatId (number), che contiene la chatId che abbiamo recuperato in precedenza.

Lo snippet da inserire nel nodo function è riportato nel Listato 1. A questo punto il nostro flow è completo e possiamo ritestarlo. Noterete, a questo punto, che in corrispondenza dei comandi */ledon* e */ledoff*, oltre all'attuazione del LED riceveremo anche la risposta, come illustrato in Fig. 17.

CONCLUSIONI

Siamo arrivati alla conclusione di questa puntata in cui abbiamo visto come integrare Node-RED con varie applicazioni di messaggistica, partendo dalle semplici email, fino ad arrivare all'integrazione di applicazioni più complesse e di recente introduzione come Whatsapp e Telegram. Nella prossima puntata ci occuperemo dell'integrazione di Node-RED con un'altra tipologia di dispositivi che sta iniziando sempre di più a diffondersi nelle nostre case: gli assistenti vocali.