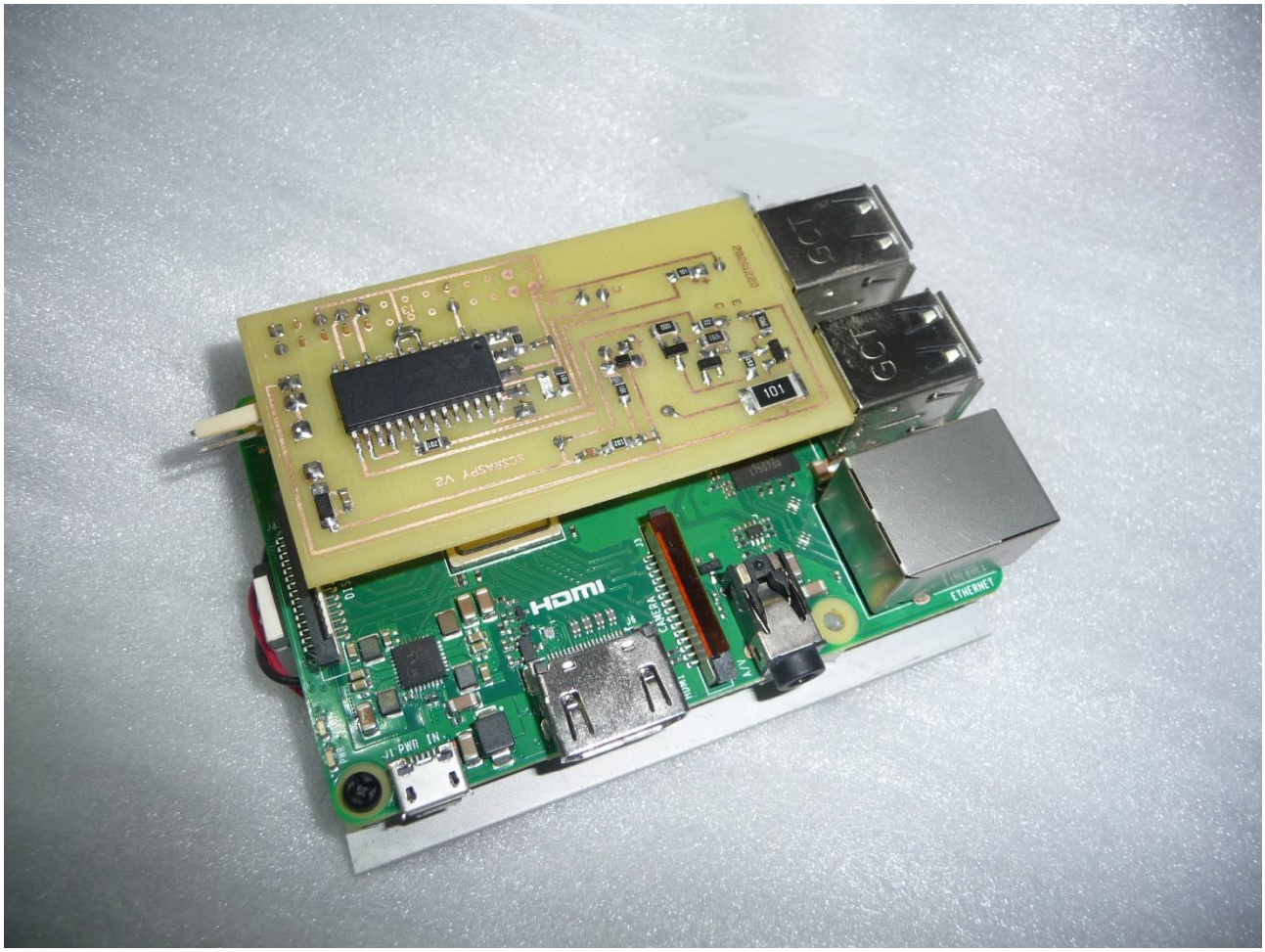
RASPY_KNXGATE V 80.5



SOMMARIO

INTRODUZIONE:

RASPY_KNXgate è un modulo di interfaccia tra il bus Konnex e RASPBERRY, disponendo di una serie di moduli software (c++) che ne consentono l'utilizzo; è un dispositivo amatoriale autocostruito e come tale privo di qualsiasi garanzia in merito al corretto funzionamento ed interfacciamento – KONNEX è un bus "proprietario" che non è lecito replicare con dispositivi commerciali senza le dovute autorizzazioni.

Esiste un isolamento galvanico tra la scheda ed il bus KNX (sono elettricamente isolati) garantito da due foto-accoppiatori con una tensione di isolamento superiore a 5KV.

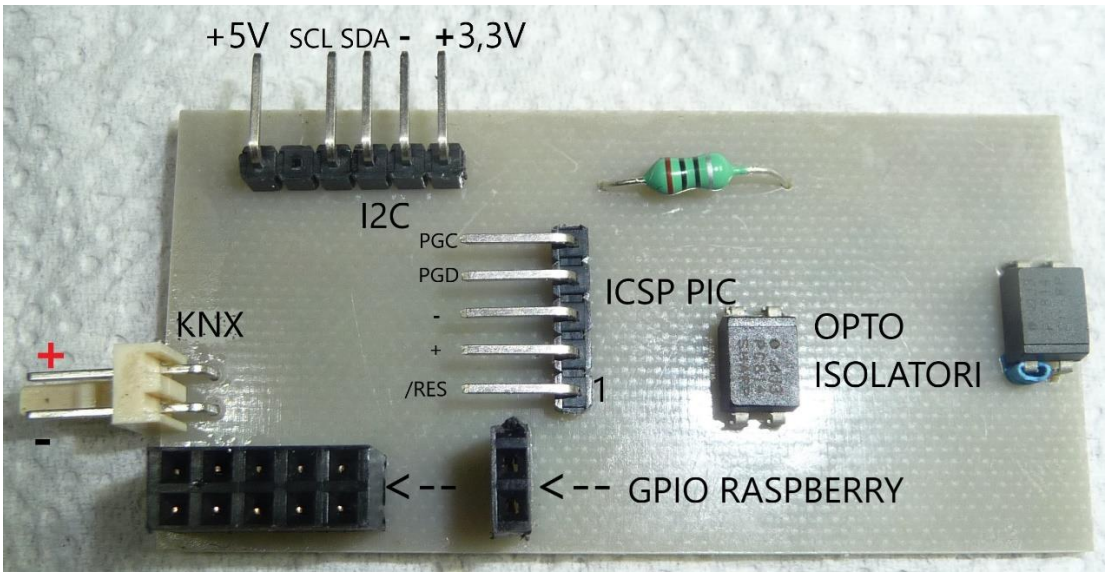
Scopo del modulo è di consentire a dispositivi locali (raspberry) o esterni (computers, Android, alexa…) di ricevere e inviare messaggi sulla rete KNX (accendere e spegnere luci, tapparelle, ecc...) - l'interfaccia si riferisce SOLO ai moduli di automazione, non alla trasmissione dati (citofonia, video, ecc...).

Il modulo è stato testato su di un bus Vimar by-me e by-me plus. E' stato testato su moduli raspberry zero W e raspberry3B+.

Il firmware è aggiornabile sia tramite i programmatori standard "Microchip" con uscita ICSP (es. pickit3) sia attraverso un software apposito direttamente da raspberry.

Questo manuale deriva da un precedente documento relativo alla scheda ESP_KNXGATE. Se trovate in qualche immagine o frase il termine "esp_knxgate" sappiate che si riferisce comunque a RASPY_KNXGATE.

INTERFACCIA RASPY_KNXGATE: CONNESSIONI



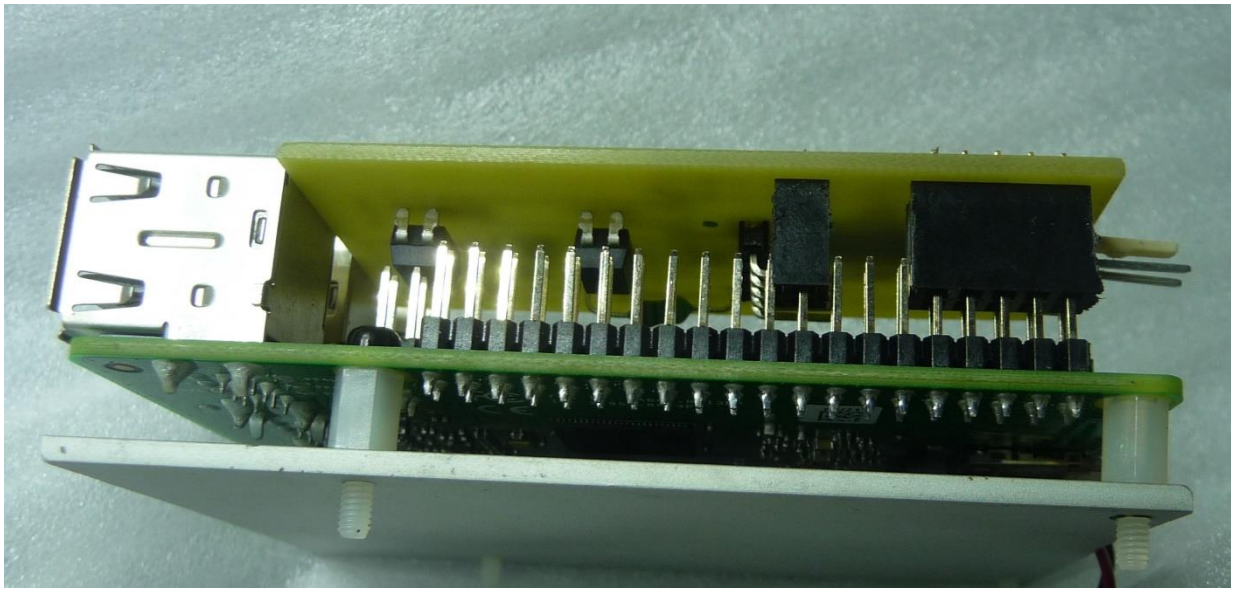
La scheda (RASPY_KNXGATE) ha 4 connettori:

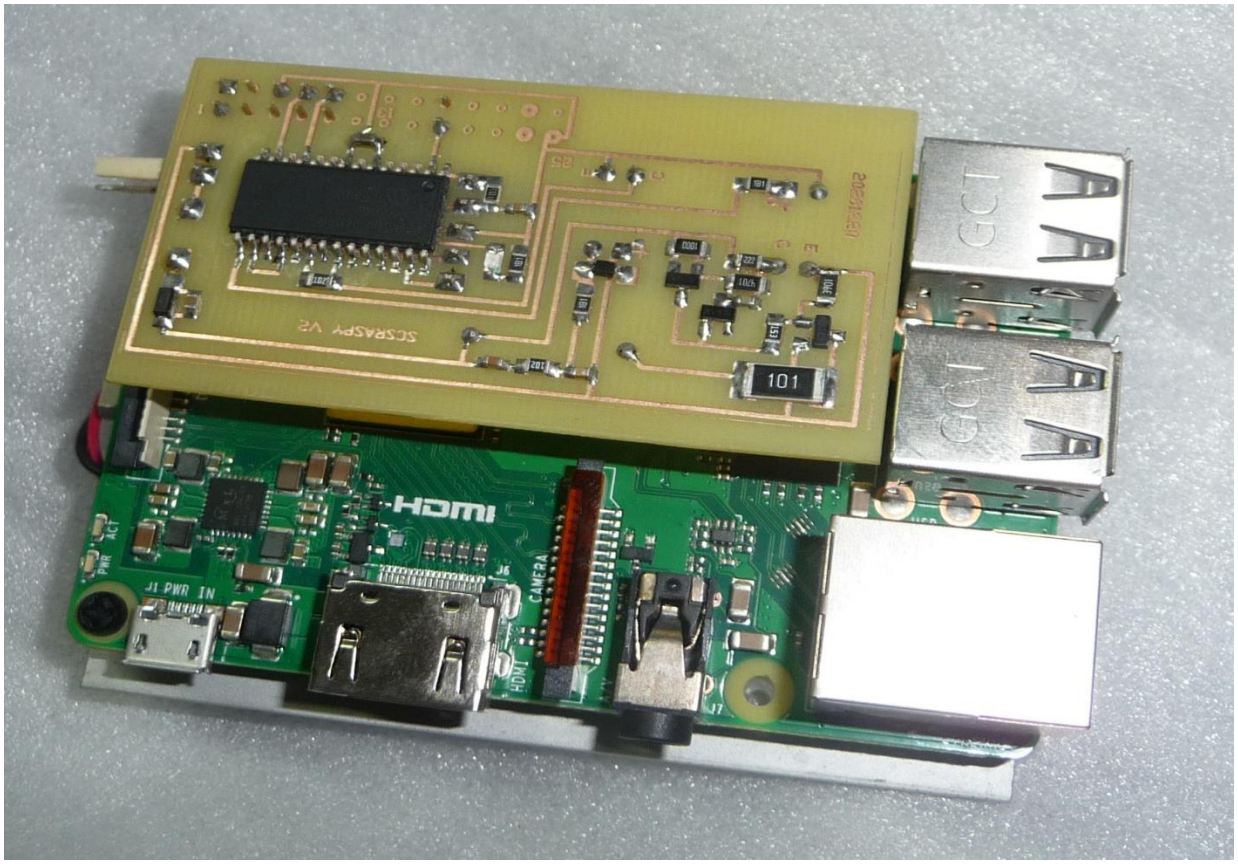
- Il connettore di sinistra va collegato al bus KNX **rispettando le polarità indicate + e –** .

Prima di collegarlo al bus controllate con un voltmetro. Il collegamento serve per i segnali di ingresso e uscita.

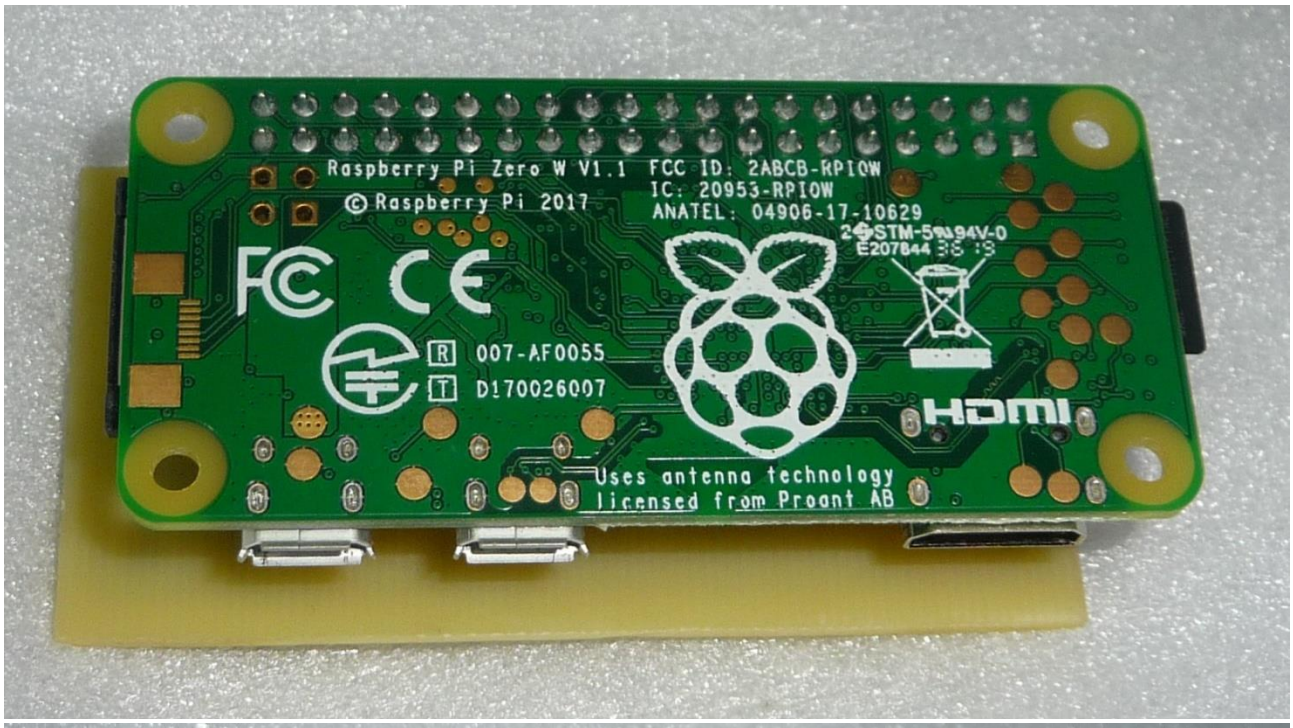
- Il connettore in basso a doppia fila è la connessione al bus gpio del raspberry da cui viene prelevata sia l'alimentazione che i segnali di comunicazione.
- Il connettore centrale a 5 pin (ICSP) serve a riprogrammare il PIC tramite un programmatore Microchip (es. pickit3): 1=/reset 2=positivo 3.3V 3=negativo 4=PGD 5=PGC
- Il connettore in alto è destinato (ad uso futuro) al collegamento con una o più schede relè e una o più schede pulsanti tramite apposite interfacce I2C.

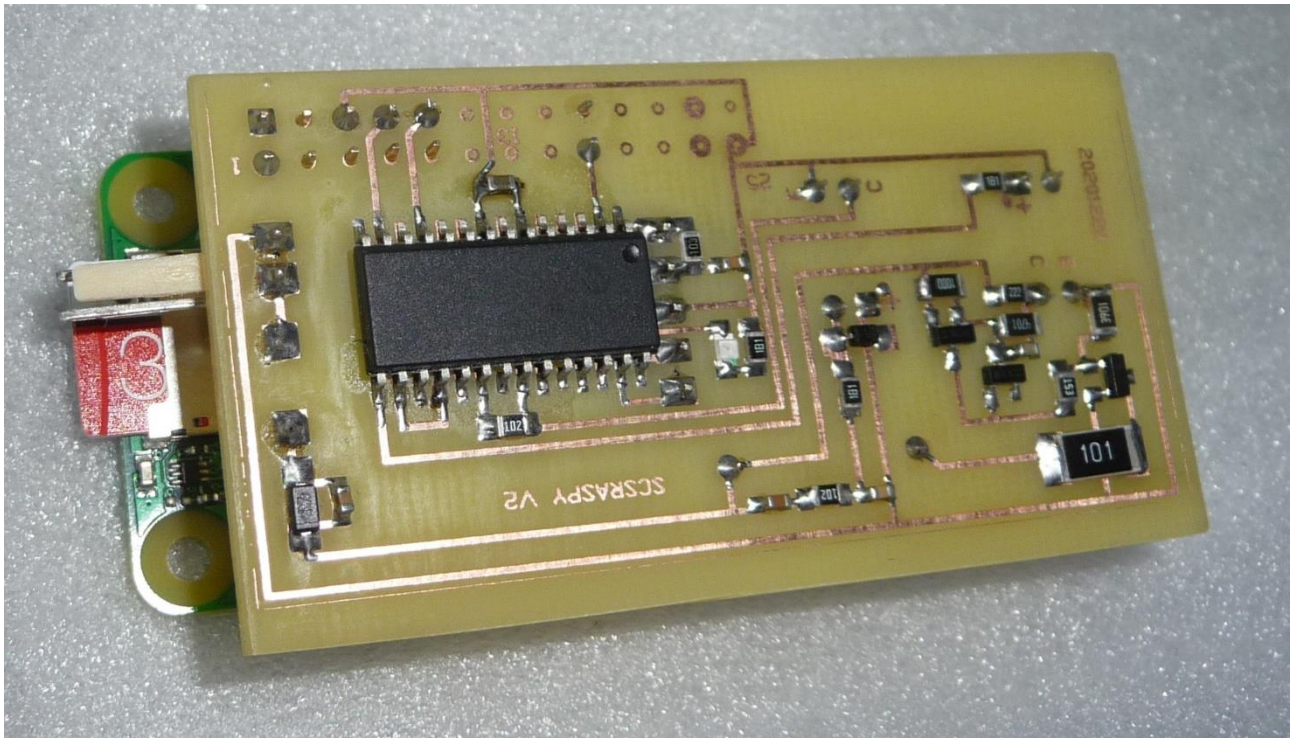
La scheda va innestata sul connettore gpio di raspberry:





Su raspberry zero W:





ACCENSIONE

Accendendo raspberry il led lampeggerà con una frequenza molto bassa (1 lampeggio ogni 10 secondi o più); con il software raspberry attivato il lampeggio si stabilizzerà a circa 1 lampeggio al secondo.

Non è indispensabile collegare il bus knx, lo potrete fare anche dopo, al momento di testare la scheda.

SOFTWARE RASPBERRY

E' RICHIESTA un minimo di competenza sulla connessione SSH con raspberry, sui comandi raspberry linux (raspbian) e (per l'aggiornamento manuale) su un software di trasferimento files ftp (es. filezilla).

Caricamento manuale:

Entrate in SSH e fate il login.

Create una directory, per esempio "raspy_knxgate_exe" e poi entrateci:

```
mkir raspy_knxgate_exe
```

```
cd raspy_knxgate_exe
```

Usando il software ftp trasferite in questa directory i files qui riportati, scaricandoli dapprima dal repository https://github.com/papergion/raspy_knxgate_exe :

- knxfirmware
- knxgate_x
- knxlog
- knxmonitor
- knxdiscover
- knxtcp
- libeasysocket.so
- libpaho-mqtt3c.so.1

tutti questi files dovranno essere dichiarati eseguibili, quindi date il comando

```
chmod +x *
```

Caricamento automatico:

Entrate in SSH e fate il login.
Verificare di avere già installato su raspberry il software GIT:

```
git –version
```

Se il software GIT non è installato:

```
sudo apt update
sudo apt install git
git --version
```

Clonate su raspberry il repository:

```
git clone https://github.com/papergion/raspy_knxgate_exe.git
```

Entrate nel repository:

```
cd raspy_knxgate_exe
```

Rendete eseguibili i files con:

```
chmod +x *
```

Per aggiornare (in futuro) il repository, entrateci e date il comando:

```
git pull
```

Dopo il caricamento:

Se doveste avere l'esigenza di cancellare il repository locale, uscite dalla directory del repository e poi date il comando:

```
rm -rf raspy_knxgate_exe
```

Ora bisogna attivare l'interfaccia seriale GPIO, usando il comando RASPI-CONFIG

```
sudo raspi-config
```

5-Interfacing options

P6-Serial

Would you like a login shell to be accessible over serial? **<NO>**

Would you like the serial port hardware to be enabled? **<YES>**

Uscendo da raspi-config vi verrà chiesto di fare il reboot di Raspberry – eseguite.

Ora potete rientrare con SSH, rifare il login e ritornare alla directory knxgate.

```
cd raspy_knxgate_exe
```

Questo l'elenco delle applicazioni ad ora disponibili ed una breve descrizione, il dettaglio a seguire.

- **knxlog** sniffer dispositivi sul bus knx
- **knxmonitor** monitor per visualizzare e inviare manualmente messaggi
- **knxdiscover** pubblica in mqtt i messaggi di discover dei dispositivi
- **knxfirmware** consente di aggiornare il software PIC
- **knxgate_x** gateway tra KNX e MQTT e/o ALEXA
- **knxtcp** gateway tra KNX e client TCP
- **gatexbatch** esempio di startup asincrono di knxgate_x
- **gatexbatchend** esempio di stop asincrono di knxgate_x

Ogni applicazione può essere lanciata singolarmente – l'esecuzione termina con <ctrl> C o con la caduta della sessione.

NON lanciate mai contemporaneamente due applicazioni da sessioni differenti – andrebbero in collisione tra loro.

Per rendere una applicazione permanente (per alcune ha senso) potete, per esempio, usare il comando linux "*nohup*" di cui potete trovare logica e sintassi su qualche manuale linux. L'applicazione partirà in maniera asincrona, lasciandovi libero il terminale SSH, e continuerà anche dopo il logout.

Un esempio lo trovate nei files "gatexbatch" (fa partire l'app knxgate_x) e "gatexbatchend" (ferma l'app knxgate_x).

Per rendere una applicazione permanente ed autostartante potete, per esempio, usare la tecnica "crontab" di cui potete prendere visione su internet.

ATTENZIONE: se lanciate una applicazione in modalità permanente RICORDATEVI di non lanciare mai contemporaneamente un'altra applicazione knx… perché avreste dei risultati fasulli.

Le applicazioni sono scritte in C++ - delle applicazioni "knxlog" e "knxmonitor" sono disponibili anche i sorgenti, come esempio di collegamento seriale con knxgate per lo sviluppo eventuale di nuove applicazioni.

La sequenza corretta di messa in funzione di raspy_knxgate è la seguente (i dettagli nei capitoli seguenti) :

- loggare i messaggi e riconoscere i vari dispositivi (knxlog / knxmonitor)
- in seguito al log generare il file "knxconfig"
- correggere manualmente il file knxconfig sistemando le descrizioni e, se necessario, aggiungendo o togliendo dispositivi
- avviare il server mqtt e l'applicativo domotico (home assistant, domoticz, …) ed eseguire "knxdiscover"
- avviare il programma/servizio knxgate_x – usate l'opzione -u
- se usate alexa fare (da alexa) il discover dei dispositivi

KNXLOG

E' la più semplice delle applicazioni: si connette a knxgate a 112500 baud – se specificato il parametro -v (verbose) scrive a video **tutti** i messaggi che passano sul bus.

Si lancia con il comando:

./knxlog

Oppure

./knxlog -v

Ogni volta che intercetta i messaggi da un nuovo dispositivo scrive una riga colorata con i dati del dispositivo riconosciuto:

verdi se non ci sono incoerenze

giallo nel caso in cui il dispositivo era precedentemente stato riconosciuto come switch (luce) ma ora in base ad un nuovo comando intercettato appare essere un dimmer.

Rosso nel caso in cui venga intercettato un dispositivo di tipo incoerente con precedenti intercettazioni.

Quindi l'uso più naturale di questo software è quello di fare un censimento dei dispositivi.

Alla chiusura (ctrl-C) l'applicazione scriverà nella directory corrente un file di nome "**discovered**" – tale file può essere rinominato in "**knxconfig**", eventualmente modificato ed utilizzato dalle altre applicazioni. Il formato di questo file è riportato nell'apposito capitolo.

ATTENZIONE: se eseguite una seconda volta **knxlog** perderete le eventuali modifiche fatte a mano sul file "discovered".

ATTENZIONE: a causa della evoluzione del sistema Vimar by-me (ora by-me plus) non risulta più possibile riconoscere AUTOMATICAMENTE i dispositivi "cover" (tapparelle) – ogni tapparella verrà riconosciuta (come fosse uno switch) con due diversi indirizzi, un indirizzo (di BASE) corrispondente al comando di STOP ed un indirizzo (BASE+1) corrispondente al comando di salita e/o al comando di discesa. L'indirizzo buono è quello BASE (di stop) – quando mettete mano al file "knxconfig" provvedete quindi a togliere la riga corrispondente all'indirizzo BASE+1. La riga con indirizzo BASE va cambiata – il tipo 1 (switch) diventa 8 (tapparella) oppure 9 (tapparella a percentuale) – in tal caso va inserito anche il parametro di posizione massima.

KNXMONITOR

L' applicazione si connette a knxgate a 112500 baud – e consente di interagire usando i comandi di base di lettura e scrittura elencati nel capitolo "il protocollo seriale di KNXGATE".

Si lancia con il comando:

./knxmonitor

L'applicazione mette la scheda knxgate in modalità ascii, attiva il log e richiama il messaggio knxgate di help @h.

Rimane poi in attesa di comandi da tastiera fino a quando riceve il comando <ctrl C>.

I comandi validi sono quelli riportati nel paragrafo “il protocollo seriale di knxgate” e consentono lettura e scrittura sul bus knx.

Attenzione, i comandi vengono letti da tastiera byte per byte e talquali rigirati verso knxgate, non sono quindi leciti comandi di movimento cursore, tab, cancellazione, ecc…

Test funzionale con ./knxmonitor:

Attivate il programma con ./knxmonitor : appariranno messaggi di questo tipo:

```
KNXMONITOR
UART_Initialization

initialized - OK

write - OK
kk
KNXgate KNX 80.557
>
Actual Vref: 22
---> modo...: A
---> filtro : 0
---> pfxname: B4
---> destin.: 0C
---> filtro byte A: NO
---> filtro byte B: NO
---> stream abbreviati: 0
---> write error: 0
---> read  error: 2
---> uart   FERR: 0
---> uart   OERR: 0
---> spikes counter: 0
---> options: FF
Gestione tapparelle %: NO

  memo EEPROM disabilitato
```

A questo punto attivate il log a video con il comando @l (L minuscolo) – il dispositivo risponde “k”. Provate ad accendere e spegnere qualche luce, a video dovreste vedere cose di questo tipo:

```
KNX[2]: B4 10 0C 0C 15 E1 00 81 29
KNX[3]: CC
KNX[4]: B4 10 0C 0C 15 E1 00 80 28
KNX[5]: CC
```

Le spiegazioni di quello che vedete le trovate sul documento di analisi del protocollo KNX (appunti). I bytes 4 e 5 rappresentano l’indirizzo del dispositivo acceso o spento (nel nostro esempio linea/settore 0B e dispositivo 15).

Provate ad accendere e spegnere il dispositivo con i comando “@w115” e “@w015” – se il dispositivo risponde siete a posto! – se non risponde può essere che il vostro indirizzo di linea non sia settato – fatelo con il comando @L0C

KNXFIRMWARE

L' applicazione si connette a knxgate a 112500 baud – e consente di aggiornare il firmware caricato nel PIC da un file locale (.bin) contenente la nuova versione.

Si lancia con il comando:

./knxfirmware -fnome del file bin test di aggiornamento (aggiornamento fittizio)

./knxfirmware -u -fnome del file bin aggiornamento reale firmware

L'applicazione visualizzerà la versione corrente del firmware caricato, poi chiederà conferma dell'aggiornamento (reale o fittizio che sia) e quindi procederà.

Per ogni blocco inviato/aggiornato visualizzerà i caratteri "." (blocco inviato) e poi "k" (blocco accettato/aggiornato).

Al termine attende la ripartenza del PIC e visualizza la versione corrente del firmware sul PIC.

KNXDISCOVER

Serve a pubblicare immediatamente in MQTT le informazioni sui dispositivi trovati e memorizzati nel file knxconfig.

E' necessario che il broker mqtt sia attivo e che sia attiva una applicazione in grado di recepire i messaggi di scoperta dei dispositivi (homeassistant, domoticz,…)

L'applicazione necessita di avere (nella directory stessa di lancio) un file denominato "**knxconfig**" contenente le definizioni di tutti i dispositivi knx riconoscibili e trattabili. Il formato di questo file è riportato nell'apposito capitolo.

Non interagisce con il bus knx.

Si lancia con il comando:

./knxdiscover ***opzioni***

Le opzioni valide sono le seguenti:

-v1/2/3 verbose a livello 1 oppure 2 oppure 3

Ogni livello rappresenta una modalità crescente di display a video, utili sostanzialmente a verificare e debuggare eventuali problemi

-Bbrokername:port

Attiva la connessione ad un broker MQTT – può essere digitato il nome o l'indirizzo ip del broker e la porta (ad esempio **-B192.168.1.150:1883**) – se il nome/indirizzo:porta non è digitato assume che sia **localhost:1883** (cioè un broker installato sullo stesso raspberry).

-Uusername

-Ppassword

Eventuali user e password per la connessione al broker MQTT

KNXGATE_X

E' l' applicazione più complessa - si connette a knxgate a 112500 baud – ed ha la possibilità di connettersi ad un broker MQTT e ad un dispositivo ECHO-DOT (alexa).

L'applicazione necessita di avere (nella directory stessa di lancio) un file denominato "**knxconfig**" contenente le definizioni di tutti i dispositivi knx riconoscibili e trattabili. Il formato di questo file è riportato nell'apposito capitolo.

Si lancia con il comando:

./knxgate_x ***opzioni***

Le opzioni valide sono le seguenti:

-v1/2/3 verbose a livello 1 oppure 2 oppure 3

Ogni livello rappresenta una modalità crescente di display a video, utili sostanzialmente a verificare e debuggare eventuali problemi

-u

Aggiorna immediatamente nel PIC l'elenco dei dispositivi ed il relativo tipo. L'opzione va usata ogni volta che nel file knxconfig si cambia l'indirizzo o il tipo o i dati di un dispositivo, oppure se ne aggiunge uno nuovo o se ne elimina uno, o se si cambia l'ordine dei dispositivi. Non è necessario se si cambiano solo le descrizioni. Non è opportuno usarlo se il file knxconfig non è cambiato.

-H

Attiva l'emulazione di un dispositivo philips HUE, indispensabile alla connessione con echo-dot ALEXA.

-Bbrokername:port

Attiva la connessione ad un broker MQTT – può essere digitato il nome o l'indirizzo ip del broker e la porta (ad esempio **-B192.168.1.150:1883**) – se il nome/indirizzo:porta non è digitato assume che sia **localhost:1883** (cioè un broker installato sullo stesso raspberry).

-Uusername

-Ppassword

Eventuali user e password per la connessione al broker MQTT

-D1/2

Connessione diretta HUE-MQTT – se specificata questa opzione i comandi hue (alexa) transitano direttamente nel server MQTT come stati (opzione -D1) o come comandi (opzione -D2) – anche in assenza della scheda KNXGATE.

KNXTCP

E' un ponte TCP che si connette a knxgate a 112500 baud – ed ha la possibilità sia di ricevere chiamate TCP dirottandone i dati verso knxgate che viceversa. Risponde anche a comandi http verso knxgate e viceversa.

L'applicazione necessita di avere (nella directory stessa di lancio) un file denominato "**knxconfig**" contenente le definizioni di tutti i dispositivi knx riconoscibili e trattabili. Il formato di questo file è riportato nell'apposito capitolo.

Si lancia con il comando:

./knxtcp ***opzioni***

Le opzioni valide sono le seguenti:

-v (verbose)

Emette una mole di display a video, utili sostanzialmente a verificare e debuggare eventuali problemi

-u

Aggiorna immediatamente nel PIC l'elenco dei dispositivi ed il relativo tipo. L'opzione va usata ogni volta che nel file knxconfig si cambia l'indirizzo o il tipo o i dati di un dispositivo, oppure se ne aggiunge uno nuovo o se ne elimina uno, o se si cambia l'ordine dei dispositivi. Non è necessario se si cambiano solo le descrizioni. Non è opportuno usarlo se il file knxconfig non è cambiato.

-pnnnnn

Indica di utilizzare per la comunicazione tcp e http la porta "nnnnn", se non indicata assume per default la porta 5045.

Il ponte TCP può essere usato per diversi scopi, i principali sono i seguenti:

- consentire una connessione tramite il (precedente) modulo vb6 scsknxgate v5.6
- consentire una connessione non-mqtt con domoticz o simili
- consentire una connessione con propri applicativi tramite chiamate tcp e/o http

i comandi accettati tramite TCP sono al paragrafo "comunicazione tcp"

informazioni ed esempi di chiamate e risposte http nel paragrafo "domoticz - http"

KNXCONFIG – IL FILE DI CONFIGURAZIONE

Come già detto il file può essere generato automaticamente con “knxlog” e poi sistemato manualmente. Esempio di file:

{"coverpct":"false","devclear":"true"}
{"device":"0C21","type":"8","maxp":"","descr":"tapparella sala"}
{"device":"0C22","type":"18"}
{"device":"0C23","type":"9","maxp":"200","descr":" tapparella cucina"}
{"device":"0C25","type":"19"}
{"device":"0C26","type":"1","descr":"salotto"}
{"device":"0C27","type":"4","descr":"scala"}
{"device":"0C28","type":"24"}
{"device":"0C29","type":"1","descr":"bagnetto"}
{"device":"0C01","type":"15","descr":"temp. zona notte”}

Non modificate la prima riga, è necessaria quando il file viene caricato nel PIC per cancellare la situazione precedente.

Ogni riga successiva rappresenta un dispositivo presente sul bus: il valore successivo a **“device”** indica l’indirizzo knx (linea/settore e dispositivo) assegnato a quel dispositivo al momento della configurazione dell’impianto. Il valore indicato da **“type”** invece indica la tipologia a cui appartiene – i tipi trattati da knxgate sono i seguenti:

“1” – switch o luce (attuatore on/off)
“4” – indirizzo base dimmer (luce regolabile) che può essere spenta/accesa/alzata/abbassata
“24” – indirizzo up/down dimmer
“8” – indirizzo base tapparella, può essere alzata/abbassata/fermata
“18” – indirizzo up/down tapparella
“9” – indirizzo base tapparella a percentuale, può essere alzata/abbassata/fermata/impostata ad un certo valore
“19” – indirizzo up/down tapparella
“11” – dispositivo generico da cui possono arrivare telegrammi
“12” – dispositivo generico a cui possono essere inviati telegrammi

I dispositivi di tipo 1 / 4 / 8 / 9 possono interagire tramite mqtt e tramite alexa. I dispositivi di tipo 11 / 12 solo con mqtt e solo in lettura. I dispositivi 24/18/19 rappresentano solo indirizzi per ottenere determinate funzioni – non vengono gestiti direttamente in mqtt/alexa.

Il parametro indicato da **“maxp”** è applicabile solo ai dispositivi di tipo 9 (tapparelle a percentuale) ed indica il tempo di salita o discesa in unità da 0,2 secondi.

Il parametro indicato da **“descr”** indica la descrizione mnemonica del dispositivo. Per i tipi 1-4-8-9 è il nome che viene poi censito da Alexa nella fase di “scoperta” e che verrà poi riconosciuto nei comandi vocali.

NOTA IMPORTANTE SUGLI INDIRIZZI

Gli indirizzi principali dei dispositivi Vimar sono normalmente dispari, con valore teorico da 01 a FF. Alcuni dispositivi come dimmer e tapparelle hanno associato anche un indirizzo pari insieme ad dispari (es: 01 e 02 oppure 59 e 5A) – l’indirizzo dispari (59) agisce sull’attuatore on-off, il pari (5A) sull’attuatore di direzione (up-down).

In alcuni impianti il settaggio è rovesciato (es: 02 e 03 oppure 5A e 5B): i dispositivi hanno indirizzo pari, in dimmer e tapparelle l'indirizzo pari (5A) agisce sull'attuatore on-off, il dispari (5B) sull'attuatore di direzione (up-down).

Per uniformità chiamiamo "**BASE**" l'indirizzo di on-off o di stop, "**BASE+1**" l'indirizzo di up-down.

Per scoprirlo è sufficiente loggare su di una tapparella i comandi di "up" e di "stop" – l'indirizzo BASE è quello loggato con lo stop.

L'ultima generazione di dispositivi (Vimar by-me plus) NON ha più vincoli di indirizzi pari/dispari anche se viene mantenuta la logica dell'indirizzo BASE e BASE+1 (per tapparelle e dimmer).

Questo complica (rende impossibile) la "scoperta" automatica dei dispositivi, che vanno perciò individuati e censiti manualmente, con il programma ./knxlog oppure ./knxmonitor.

Esempio:

```
KNX[1]: B4 10 01 0B 0C E1 00 81 3D
KNX[2]: B4 10 0C 0B 0B E1 00 80 36
KNX[3]: B4 10 01 0B 31 E1 00 80 3C
KNX[4]: B4 10 0C 0B 47 E1 00 80 38
KNX[5]: B4 10 01 0C 24 E1 00 80 3F
```

In giallo l'indirizzo del dispositivo ricevente. Possono apparire anche dei messaggi con unico dato "CC" (sono gli acknowledge dei dispositivi)

Se accendete una luce potete vedere uno o due messaggi: il primo è il comando di accensione diretto all'attuatore, il secondo la risposta della centrale che comanda ai led dei pulsanti di accendersi. A noi serve individuare gli indirizzi degli attuatori, perciò prendete nota di indirizzi e nomi da abbinare.

Supponiamo che abbiate "scoperto" l'indirizzo 0B31 relativo alla luce della cucina; potete provare ad accenderla digitando @w131 (non digitiamo 0B perché normalmente è un dato comune a tutti i dispositivi, quello che appare nell'help (@h) sotto la voce "destin:"; @w031 la spegnerà.

Azionando una tapparella scoprirete l'indirizzo di "base" (quello che la ferma) e l'indirizzo di base+1 (quello che la alza o abbassa) – prendete nota.

Azionando un dimmer scoprirete l'indirizzo di "base" (quello che lo accende o spegne) e l'indirizzo di base+1 (quello che la alza o abbassa) – prendete nota.

Modo alternativo – se avete a disposizione un gateway Vimar 01410/01411 e l'applicazione "View Pro": aprite l'applicazione e collegatevi al gateway – entrate nel menu "applicazioni" – per ogni applicazione cliccate prima sul simbolo della "penna" e poi a sinistra su "indirizzi di gruppo". L'indirizzo dell'attuatore (luce) è quello della riga "**DPTx_onoff**". Per le tapparelle l'indirizzo **base** è quello della riga **DPTx_StopStepUpDown,** l'indirizzo base+1 è quello della riga **DPTx_UpDown.**

CONNESSIONE MQTT

NON inserite l'IP di un broker se non lo userete, appesantireste inutilmente il lavoro dell'applicazione in operazioni inutili.

La connessione con il broker avviene all'inizio e se andata a buon fine viene mantenuta – se poi la connessione cade l'applicazione ritenta la connessione ogni minuto.

La connessione MQTT è relativa a luci, dimmer, tapparelle, termostati. e sottoscrive i seguenti topics:

knx/+/set/+ per i comandi on/off/stop dei dispositivi

luci o dimmer	knx/switch/set/<knx *address*>	payload ON o OFF
cover (tapparelle)	knx /cover/set< knx *address*>	payload ON o OFF o STOP

knx /+/setlevel/+

luci o dimmer	knx /switch/setlevel/< knx address>	payload da 10 a 90

knx /+/setposition/+

cover (tapparelle a %)	knx /cover/setposition< knx *address*>	payload da 1 a 100 (%)

PUBBLICAZIONE STATO DISPOSITIVI:

La connessione MQTT pubblica, (ad ogni notifica di stato che transita sul bus), i seguenti topics di aggiornamento stato dispositivi:

knx /switch/state/< knx *address*> payload "ON" o "OFF" (per luci/attuatori/dimmer)

knx /switch/value/< knx *address*> payload 00-01 oppure 10-255 (per dimmer)

knx /cover/state/< knx *address*> payload "open" o "closed" o "STOP" (per tapparelle)

knx /cover/value/< knx *address*> payload 0-100 (per tapparelle gestite a percentuale)

Se è stato settato l'uso per HOME-ASSISTANT gli stati pubblicati per le tapparelle invece saranno "open" e "closed".

CAMBIO DI STATO DISPOSITIVI:

La connessione MQTT accetta i seguenti topics di comando sui dispositivi, li trasforma in telegrammi KNX e li invia sul bus tramite knxgate:

knx/switch/set/< knx *address*> payload "ON" o "OFF" (per luci/attuatori/dimmer)

knx/switch/setlevel/< knx *address*> payload 10-90 oppure 10-255 (per dimmer)

knx /cover/set/< knx *address*> payload "OPEN" o "CLOSE" o "STOP" (per tapparelle)

knx /cover/setposition/< knx *address*> payload 0-100 (per tapparelle gestite a percentuale)

DISPOSITIVI "GENERICI":

Alcuni telegrammi che transitano sul bus KNX possono non riguardare i classici dispositivi elencati (switch, dimmer, tapparelle) ma dispositivi differenti di cui non mi è noto il significato del messaggio; per esempio videocitofoni, impianti di allarme, ecc… Questi dispositivi possono essere censiti come "GENERICI" (tipo 11 e 12).

Ricordo che la struttura dei messaggi KNX è:

<PREFISSO> <provenienza> <destinazione> <ctr - pdu> <valore> <checkbyte>

Alla ricezione dal bus di un messaggio viene analizzata la <destinazione> e se l'indirizzo risulta censito come dispositivo ti tipo 11 viene pubblicato un topic:

knx/generic/to/<*destinazione*> payload: <provenienza> <valore>

Viene analizzata la <provenienza> e se l'indirizzo risulta censito come dispositivo di tipo 12 viene pubblicato un topic:

knx/generic/from/<provenienza> payload: <destinazione> <valore>

Viene invece generato un messaggio sul bus se in MQTT viene ricevuto un topic:

knx/generic/set/<*destinazione*> payload: <provenienza> <valore>

ALEXA

Sono del parere che questo tipo di interfaccia sia il futuro della domotica. Anche se ancora limitata e pomposamente chiamata “intelligenza artificiale” è comunque un ottimo primo passo (insieme ad altre realtà) nel mondo dell’interazione vocale.

Con il termine “Alexa” si definisce un processo complesso, che “gira” su dei server remoti e che interagisce con dei terminali locali “echo dot” o altro – ovviamente se non siete connessi ad internet nulla funziona.

La prima possibilità di integrazione tra raspy_knxgate ed Alexa passa attraverso HOME-ASSISTANT. Su Amazon-Alexa potete installare gratuitamente la skill di integrazione. Purtroppo questa skill utilizza il “cloud” come passaggio di integrazione, e purtroppo Amazon richiede un pedaggio mensile di qualche euro.

La seconda possibilità di integrazione tra raspy_knxgate ed Alexa passa ancora attraverso HOME-ASSISTANT. Se avete un po’ di dimestichezza con i sistemi di sviluppo software, seguendo questa guida (ce ne sono diverse) potrete scrivere (clonare) la vostra skill ed ottenere l’integrazione senza pagare canoni a nessuno: https://indomus.it/guide/integrare-gratuitamente-amazon-echo-alexa-con-home-assistant-via-haaska-e-aws/

La terza possibilità è la più semplice (e limitata) e consiste nel connettere direttamente raspy_knxgate con Alexa – il firmware di raspy_knxgate (knxgate_x) lo consente, utilizzando un software che emula un dispositivo “philips hue” (lampada intelligente), da cui derivano queste possibilità e limitazioni:

- Le luci potranno essere comandate direttamente da Alexa con i comandi vocali “*accendi*” e “*spegni*”
- Le luci dimmerabili potranno essere comandate direttamente da Alexa con i comandi vocali “*accendi*”, “*spegni*”, “*alza*”, “*abbassa*”, “*imposta al xx%*”
- Le tapparelle , se gestite senza %, potranno essere comandate direttamente da Alexa con i comandi vocali “*alza*” , “*abbassa*” , “*ferma*” – il comando “ferma” arresterà il movimento della tapparella ma l’interfaccia vocale risponderà che qualcosa non ha funzionato
- Le tapparelle, se gestite con %, potranno essere comandate direttamente da Alexa con i comandi vocali “*alza*”, “*abbassa*”, “*ferma*”, “*imposta al xx%*” – i comandi “alza” e “abbassa” muoveranno la tapparella in su o in giù circa del 25% a botta (per alzare o abbassare totalmente una tapparella occorrerà usare i comandi “imposta al 100%” e “imposta al 0%” – inoltre il comando “ferma” arresterà il movimento della tapparella ma l’interfaccia vocale risponderà che qualcosa non ha funzionato

I passi necessari ad utilizzare l’interfaccia diretta Alexa-raspy_knxgate sono i seguenti:

- Preliminarmente dovete aver individuato i dispositivi knx e aver compilato il file knxconfig.
- Il processo di Alexa lavora sulla porta 80 – se tale porta fosse già in uso da parte di altri processi nel raspberry riceverete un errore. Dovete liberarla.
- Attivate l’applicazione knxgate_x con il parametro -H (***sudo ./knxgate_x -H***)
- Se avete più di un echo-dot lasciate spenti tutti gli echo-dot secondari, va lasciato acceso solo quello principale (solitamente il primo che si è installato)
- A questo punto andate sull’APP (o sulla pagina https://alexa.amazon.it) di ALEXA – cliccate sull’opzione “Casa intelligente” e poi su “Dispositivi”. In fondo alla pagina cliccate su “Trova” e aspettate che Alexa termini il lavoro di scoperta di nuovi dispositivi
- Se nessun dispositivo viene scoperto, **spegnete e riaccendete echo-dot**, aspettate circa 30 secondi e riprovate
- A volte è poi necessario refreshare la pagina. A volte capita che uno dei dispositivi appaia censito due volte – non importa.

Se avete un numero cospicuo di dispositivi (più di 30) alcuni potrebbero mancare – in tal caso ripetete l'operazione di "Trova" più volte, ogni volta se ne aggiungeranno circa 30.

Avete fatto tutto – provate a dire "Alexa, accendi <nome dispositivo>

Attenzione: se correggete il nome di un dispositivo, cancellatelo dai dispositivi scoperti su Alexa e poi rifate il "Trova".

Attenzione: se aggiungete o togliete dei dispositivi dalla lista "knxconfig" dovete farli "dimenticare" TUTTI ad alexa e quindi rifare il "Trova" perché il codice interno di alexa cambia

Come ho precisato, questa è una BETA-VERSION per quanto riguarda l'accesso diretto ad Alexa, può essere che qualcosa ancora non funzioni per il verso giusto… comunicatemi il vostro problema e cercherò di risolverlo

Attenzione ai nomi che date ai dispositivi, alexa può avere difficoltà a riconoscere certi nomi o fare confusione quando più dispositivi hanno nomi simili. Usando le "routine" potreste superare il problema.

In una prossima versione il software consentirà anche di utilizzare come dipositivi i pin gpio di raspberry, in ON/OFF.

COMUNICAZIONE TCP

La connessione TCP può essere aperta solo con raspy_knxgate connesso come client di un router, sulla porta 5045.

I pacchetti che arrivano sulla porta TCP vengono dapprima analizzati nel contenuto: se contengono un comando valido secondo la lista che segue esso viene eseguito. Diversamente, se è stato "aperto" il canale di comunicazione tcp-uart essi vengono inviati tal quali a KNXGATE che li interpreta esattamente come il protocollo UDP.

I dati che provengono da KNXGATE (se è stato "aperto" il canale di comunicazione tcp-uart) vengono impacchettati ed inviati tal quali all'indirizzo IP ed alla porta che hanno effettuato l'ultimo invio a RASPY_KNXGATE.

I comandi validi sono:

#setup {"uart":"tcp"} apre il canale di comunicazione TCP verso KNXGATE come sopra descritto. Il canale rimane aperto sino a che non viene ricevuto un pacchetto UDP. Dopo questo comando ogni altro messaggio che non inizi con # verrà automaticamente girato a knxgate che risponderà sempre in TCP al chiamante. Le chiamate lecite e le relative risposte sono elencate nel paragrafo "il protocollo seriale di knxgate"

#setup {"uart":"no"} chiude il canale di comunicazione TCP verso KNXGATE come sopra descritto.

#request {"device":"dddd", "command":"cc"} invia il comando KNX "cc" al dispositivo con indirizzo "dd". Comando e indirizzo vanno espressi con caratteri ascii esadecimali (come nel protocollo UDP in modalità ascii).

IL PROTOCOLLO SERIALE DI KNXGATE

Vale sia per la comunicazione TCP che UDP. I comandi che arrivano nel pacchetto wifi vengono trasposti tal quali al PIC che li interpreta e li esegue.

Lo scambio di dati viene sempre sollecitato da un comando inviato al dispositivo – ogni comando ha il seguente formato (i simboli < e > vengono utilizzati come separatori):

<@><comando><valore><dati opzionali>

Il modulo risponde sempre ma in vario modo a seconda del comando.

La connessione può essere effettuata in UDP sulla porta specificata all'atto della configurazione WiFi (default 52056), oppure in TCP sulla porta 5045 – in quest'ultimo caso riferirsi innanzitutto al paragrafo "Comunicazione TCP"

@M<valore>

Imposta la modalità di comunicazione sulla porta seriale; i valori possono essere i seguenti:

X : esadecimale (default): i dati vengono scambiati in esadecimale puro (bytes da 0x00 a 0xFF)

A : ascii: i dati vengono scambiati con caratteri ascii, per esempio l'esadecimale 0x2A va inviato con 2 caratteri, il carattere '2' seguito dal carattere 'A'. In modalità ascii la lunghezza effettiva dei dati inviati risulta quindi doppia rispetto alle lunghezze specificate che sono quindi lunghezze logiche. La modalità ascii è utile per fare test diretti con programmi standard di interfaccia seriale (tipo "putty").

Il valore impostato viene memorizzato nella eeprom del PIC e rimane valido anche spegnendo e riaccendendo il dispositivo.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@F<valore>

Permette di applicare un filtro sui messaggi KNX per ignorare quelli che non interessano; i valori espressi sempre in ascii possono essere i seguenti:

0 : nessun filtro

1 : i messaggi doppi o tripli vengono trasmessi in seriale una volta sola.

2 : i messaggi di ACK (0xCC) vengono ignorati.

3 : comprende sia il filtro "**1**" che il filtro "**2**".

4 : i messaggi di stato vengono ignorati.

Il valore impostato viene memorizzato nella eeprom del PIC e rimane valido anche spegnendo e riaccendendo il dispositivo.

Sono accettati anche i valori 5-6-7: il filtro applicato consiste nella somma dei filtri 1-2-4 così come il valore è sommato (7=4+2+1)

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@A<tipo><numero><valore>

Permette di applicare un filtro su di un byte dei messaggi KNX per ignorare quelli che non interessano; i valori espressi **sempre in ascii** possono essere i seguenti:

tipo: può valere "**i**" (includi) oppure "**e**" (escludi) oppure "**0**" o altro (rimuovi il filtro)

numero: da "**1**" a "**9**" indica il numero del byte da osservare in ogni telegramma

valore: da "**00**" a "**FF**" indica il valore che deve essere trovato su tale byte per escludere o includere il telegramma tra quelli ricevuti

Il filtro impostato viene memorizzato nella eeprom del PIC e rimane valido anche spegnendo e riaccendendo il dispositivo.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@B<tipo><numero><valore>

Permette di applicare un filtro su di un byte dei messaggi KNX per ignorare quelli che non interessano; i valori espressi **sempre in ascii** possono essere i seguenti:

tipo: può valere "**i**" (includi) oppure "**e**" (escludi) oppure "**0**" o altro (rimuovi il filtro)

numero: da "**1**" a "**9**" indica il numero del byte da osservare in ogni telegramma

valore: da "**00**" a "**FF**" indica il valore che deve essere trovato su tale byte per escludere o includere il telegramma tra quelli ricevuti

Il filtro impostato viene memorizzato nella eeprom del PIC e rimane valido anche spegnendo e riaccendendo il dispositivo.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

Nota sui filtri: utilizzando più di un filtro (@F - @A - @B) questi vengono applicati sequenzialmente e i telegrammi in uscita devono superare TUTTI i filtri; se un solo filtro nega l'uscita il telegramma viene scartato.

@L<valore>

Permette di impostare il valore del byte di "linea e settore" del dispositivo di destinazione nei messaggi – inizialmente questo valore è impostato a 0B.

Il valore impostato viene memorizzato nella eeprom del PIC e rimane valido anche spegnendo e riaccendendo il dispositivo. Viene utilizzato per definire la linea e settore usata poi nei comandi "brevi" (@w)

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@r

Permette di leggere dal dispositivo il successivo telegramma ricevuto e bufferizzato; la lettura lo rimuove dal buffer. Il dispositivo ha un buffer in grado di contenere fino a 10 telegrammi; la mancata lettura dei messaggi provoca un overflow del buffer con conseguente perdita dei telegrammi più vecchi.

Risposta:

<lunghezza> : sempre e solo un carattere (ascii o esadecimale puro) da '**0**' a '**F**' che indica la lunghezza logica dei dati rimanenti (se vale '0' significa che non ci sono telegrammi in attesa di scodamento)

<dati> : i dati del telegramma in esadecimale puro oppure in ascii a seconda del modo operativo

@R

Permette di leggere dal dispositivo il successivo telegramma ricevuto e bufferizzato; la lettura lo rimuove dal buffer. Il dispositivo ha un buffer in grado di contenere fino a 10 telegrammi; la mancata lettura dei messaggi provoca un overflow del buffer con conseguente perdita dei telegrammi più vecchi. Se il buffer non contiene messaggi la risposta viene differita: la risposta avrà luogo solo quando il buffer conterrà un telegramma.

Risposta:

<lunghezza> : sempre e solo un carattere (ascii o esadecimale puro) da '**0**' a '**F**' che indica la lunghezza logica dei dati rimanenti (non può essere 0)

<dati> : i dati del telegramma in esadecimale puro oppure in ascii a seconda del modo operativo

@c

Permette di uscire dallo stato di attesa innescato dal comando @R.

Risposta: <**k**>

@W<valore><dati>

Permette di trasmettere al dispositivo un telegramma da inviare in rete

<valore> : sempre e solo un carattere (ascii o esadecimale puro) da '**0**' a '**F**' che indica la lunghezza logica dei dati rimanenti

<dati> : i dati del telegramma in esadecimale puro oppure in ascii a seconda del modo operativo

Risposte possibili: <k> : tutto ok <E> : errore

@w<valore><dati>

Modalità rapida per mandare un telegramma da inviare in rete. Attenzione: questa modalità si basa su di una serie di valori di default che vanno sicuramente bene per reti Vimar by-me ma che potrebbero non funzionare su altre reti KONNEX. Si consiglia quindi di usare per tali scopi il comando @W.

<valore> : sempre e solo un carattere (ascii o esadecimale puro) da '**0**' a '**F**' che rappresenta il comando da inviare al dispositivo ('1' accendi, '0' spegni)

<dati> : indirizzo dell'attuatore in esadecimale puro (1 byte) oppure in ascii (2 bytes) a seconda del modo operativo

Per poter utilizzare correttamente questa modalità è indispensabile che il valore di linea e settore del dispositivo ricevente sia stato precedentemente memorizzato nella eeprom del pic (vedi comando @D).

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@Y0

Chiusura modalità abbreviata dei telegrammi. Dopo questo settaggio i telegrammi ricevuti con il comando @r oppure @R oppure @l verranno ritornati in modalità normale.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@Y1

Settaggio modalità abbreviata dei telegrammi. Dopo questo settaggio i telegrammi ricevuti con il comando @r oppure @R oppure @l verranno ritornati in modalità abbreviata, corrispondenti ai bytes 3-4-5-8 del telegramma ricevuto (indirizzo mittente, linea/settore destinatario, indirizzo destinatario, comando).

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@Y3

Settaggio modalità abbreviata dei telegrammi. Dopo questo settaggio i telegrammi ricevuti con il comando @r oppure @R oppure @l e i telegrammi scritti con il comando @W verranno trattati in modalità abbreviata, corrispondenti ai bytes 3-4-5-8 del telegramma (indirizzo mittente, linea/settore destinatario, indirizzo destinatario, comando).

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@y<dati brevi>

Modalità abbreviata per mandare un telegramma di comando da inviare in rete – indipendente dal settaggio @Y

<dati brevi> : 4 caratteri in formato esadecimale puro – corrispondenti ai bytes 3-4-5-8 del telegramma da inviare (indirizzo mittente, linea/settore destinatario, indirizzo destinatario, comando).

Risposte possibili: <**k**> : tutto ok <**E**> : errore

I valore impostati con i comandi @M, @F, @D , @Y1, @Y3 vengono memorizzati nella eeprom del PIC e rimangono impostati anche spegnendo e riaccendendo il dispositivo.

@<0x15>

Dopo questo comando i settaggi successivi NON verranno più memorizzati in eeprom ma impostati in maniera temporanea, fino a che il dispositivo non viene resettato, oppure fino alla ricezione del comando **@<0x16>**. Inoltre in questa modalità il dispositivo eviterà anche la classica risposta di acknowledge “k”

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@b

Pulisce tutti i buffers di ricezione.

Risposta : <**k**> : tutto ok

@h

Espone un menu di help – solo in modalità ASCII.

@o

Re-imposta la modalità standard (estesa) di conversazione togliendo eventuali filtri e abbreviazioni.

Risposta : <**k**>

@q

Query version – solo in modalità ASCII.

Risposta : <**k**> seguito dalla VERSIONE : tutto ok

@l (L minuscolo)

Attiva un log a video dei messaggi ricevuti e trasmessi

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@c

Pulisce lo stato di attesa - solo in modalita ASCII.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@d

Dump di tutti i buffers di ricezione – solo in modalita ASCII.

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@D<indirizzo>

Permette di conoscere gli indirizzi dei dispositivi scoperti sul bus – viene utilizzato da ESP8266 per sincronizzare l'elenco dei dispositivi scoperti sul bus. Ne sconsiglio l'uso manuale.

<indice di partenza scansione> : carattere esadecimale puro oppure due caratteri ascii che indicano da quale indice partire con la scansione. L'indirizzo reale si ottiene dal calcolo (<indice> * 2) – 1. (l'indirizzo di settore/linea viene ignorato, si presuppone uguale per tutto l'impianto e viene memorizzato all'indice 0)

Risposta: D**<indice><tipo>**

<indice > : carattere esadecimale puro oppure due caratteri ascii che indicano quale indice valido è stato trovato. L'indirizzo reale si ottiene dal calcolo (<indice> * 2) – 1.

<tipo > : carattere esadecimale puro oppure due caratteri ascii che indicano che tipo di dispositivo è stato trovato (1=switch 3=dimmer 8=tapparella 0xFF=nessuno)

@z

Elenca in formato testuale gli indirizzi dei dispositivi scoperti sul bus ed il relativo "tipo" (1=switch, 4=dimmer, 8=tapparella, 9=tapparella gestita a %)

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@s

Sbilancia l'indice dei buffers di ricezione simulando una ricezione di messaggio – solo in modalita ASCII-

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@Ux

Comandi di settaggio e lista delle tapparelle in modalità *percentuale*. Il suffisso x può assumere i seguenti valori:

1. La modalità gestione a percentuale viene disattivata
2. Modalità di attesa setup automatico. Dopo questo comando ogni tapparella va chiusa, stoppata, aperta completamente, stoppata – per calcolare i tempi di salita.
3. Conclusione del setup automatico, i tempi calcolati vengono memorizzati nella eeprom del PIC e si entra in modalità 3.
4. La modalità a percentuale è attiva. Se il log @l è attivo, oltre ai telegrammi verranno notificate anche le posizioni intermedie raggiunte; in formato HEX rappresentate in 3 bytes nella forma "u<device><posizione>", in formato ASCII con righe con CRLF in testa seguito sempre da "u<device><posizione>" dove device e posizione sono espressi in esadecimale su 2 bytes ciascuno.
5. La modalità percentuale è attiva ma le posizioni intermedie non vengono notificate sul log.
6. Questo comando elenca in formato ascii lo stato delle tapparelle a percentuale, con l'indirizzo del dispositivo, la posizione massima, la posizione attuale, la direzione di movimento attuale, la posizione richiesta, il timeout attuale.
7. Elenco in formato HEX dello stato di una tapparella, il cui indirizzo va specificato subito dopo il "6". In formato ASCII invece questo comando può essere usato per impostare manualmente una nuova tapparella o cambiare il tempo di posizionamento. Dopo questo comando il PIC chiede l'indirizzo della tapparella, da digitare in 2 caratteri (00-7F) e quindi espone e richiede il valore (in decimi di secondo) per il tempo di salita. Dopo il tempo va digitato <invio> o <spazio>. Se non si digita nulla rimane impostato il tempo precedente.
8. Imposta il flag di pubblicazione immediata della posizione di tutte le tapparelle

9. Solo per uso interno
10. Azzera e ripulisce dalla eeprom la tabella tapparelle a percentuale

@u<indirizzo><percentuale>

Richiesta posizionamento tapparella gestita a percentuale

<indirizzo> : indirizzo knx (lineasettore+device) della tapparella da comandare in esadecimale puro (2 byte) oppure in ascii (4 bytes) a seconda del modo operativo.

<percentuale> : percentuale di apertura richiesta da 0 a 100 (hex 00-64) in esadecimale puro (1 byte) oppure in ascii (2 bytes) a seconda del modo operativo

Risposte possibili: <**k**> : tutto ok <**E**> : errore

@m<indirizzo><percentuale>

Richiesta settaggio dimmer

<indirizzo> : indirizzo knx (lineasettore+device) del dimmer da comandare in esadecimale puro (2 byte) oppure in ascii (4 bytes) a seconda del modo operativo.

<percentuale> : percentuale di luce richiesta da 0 a 254 (hex 00-FE) in esadecimale puro (1 byte) oppure in ascii (2 bytes) a seconda del modo operativo

Risposte possibili: <**k**> : tutto ok <**E**> : errore

Comandi di settaggio

Il ricevitore dei segnali sul bus ha un valore di "sensibilità" che è preimpostato ad un valore base che dovrebbe essere sempre adeguato. Tuttavia in casi particolari può essere utili ritararlo, per abbassare la sensibilità qualora vengano loggati dei messaggi spuri, generati da disturbi sulla linea, o (più raramente) aumentata qualora non riceva i telegrammi. Il valore (Vref) può variare da 1 a 31; il valore corrente viene esposto dal comando @h dato in modalità ascii.

@i

Questo comando può essere dato quando sul bus non circolano assolutamente telegrammi. Il PIC legge la tensione sul bus ed imposta una sensibilità medio-alta adatta allo scopo. Il comando può essere dato solo in modalità ascii.

@I+ (i maiuscolo)

Questo aumenta la sensibilità di 1 punto. Il comando può essere dato solo in modalità ascii.

@I- (i maiuscolo)

Questo diminuisce la sensibilità di 1 punto. Il comando può essere dato solo in modalità ascii.

GESTIONE TAPPARELLE A PERCENTUALE

Va chiarito che la gestione della percentuale di apertura è gestita a tempo; significa che per esempio una tapparella è aperta al 50% quando è trascorsa la metà del tempo necessario all'apertura completa (ma probabilmente a questo punto la tapparella sarà a meno della metà di apertura a causa del tempo impiegato ad aprire le righe). Inoltre, probabilmente la discesa sarà leggermente più veloce e quindi la percentuale di apertura potrà differire. Ad ogni chiusura completa comunque la percentuale va a zero ed il calcolo riparte da capo.

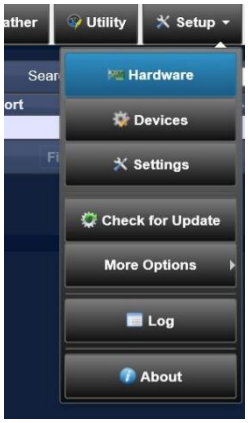
Per conoscere e gestire la percentuale di apertura sono quindi necessarie delle operazioni preliminari. Tali operazioni, pur essendo descritte nel paragrafo "MQTT" al capitolo "Pubblicazione censimento dispositivi", possono essere eseguite anche senza una connessione mqtt, limitandosi a memorizzare i dati delle tapparelle a percentuale.

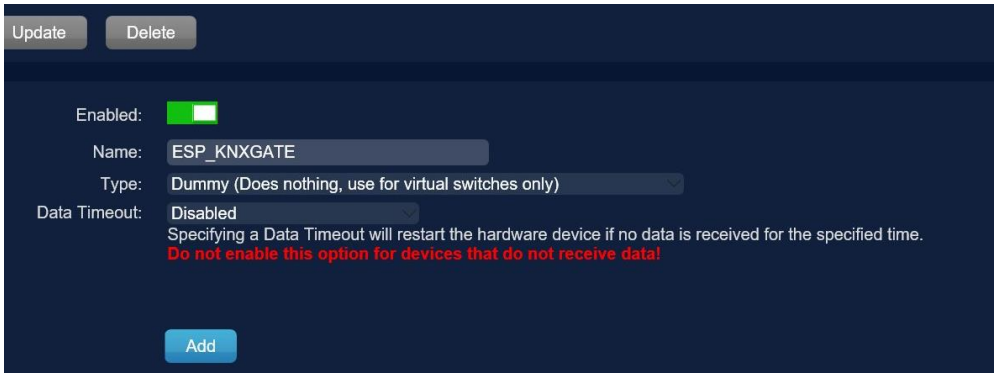
DOMOTICZ - HTTP

Sappiate che non sono affatto un esperto di domoticz – la mini guida che segue serva a mostrarvi i passi che ho fatto e come l'ho integrato.

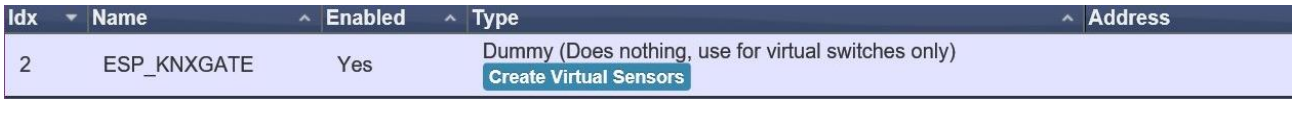
Prima ancora di cimentarvi con domoticz è necessario "sniffare" l'impianto, magari usando come già detto il mio programmino VB6, per scoprire tutti gli indirizzi e i comandi associati ai vostri dispositivi KNX.

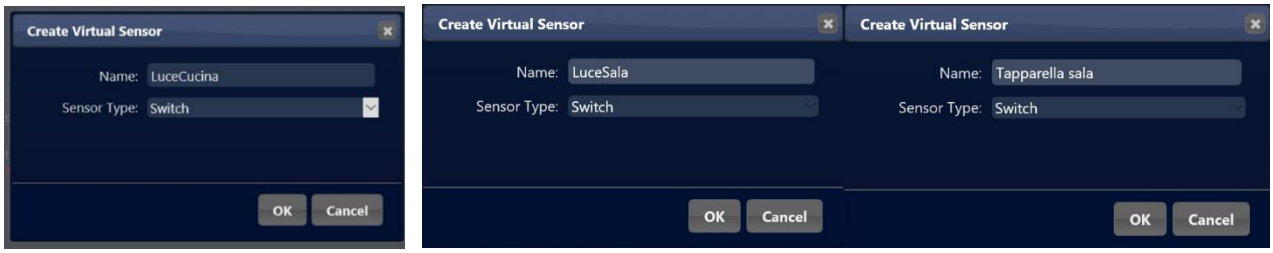
Primo passo: in domoticz ho definito un hardware virtuale (di tipo Dummy) e l'ho chiamato ESP_KNXGATE





Associati a questo hardware ho definito dei "sensori virtuali" – uno per ciascun attuatore KNX che voglio controllare



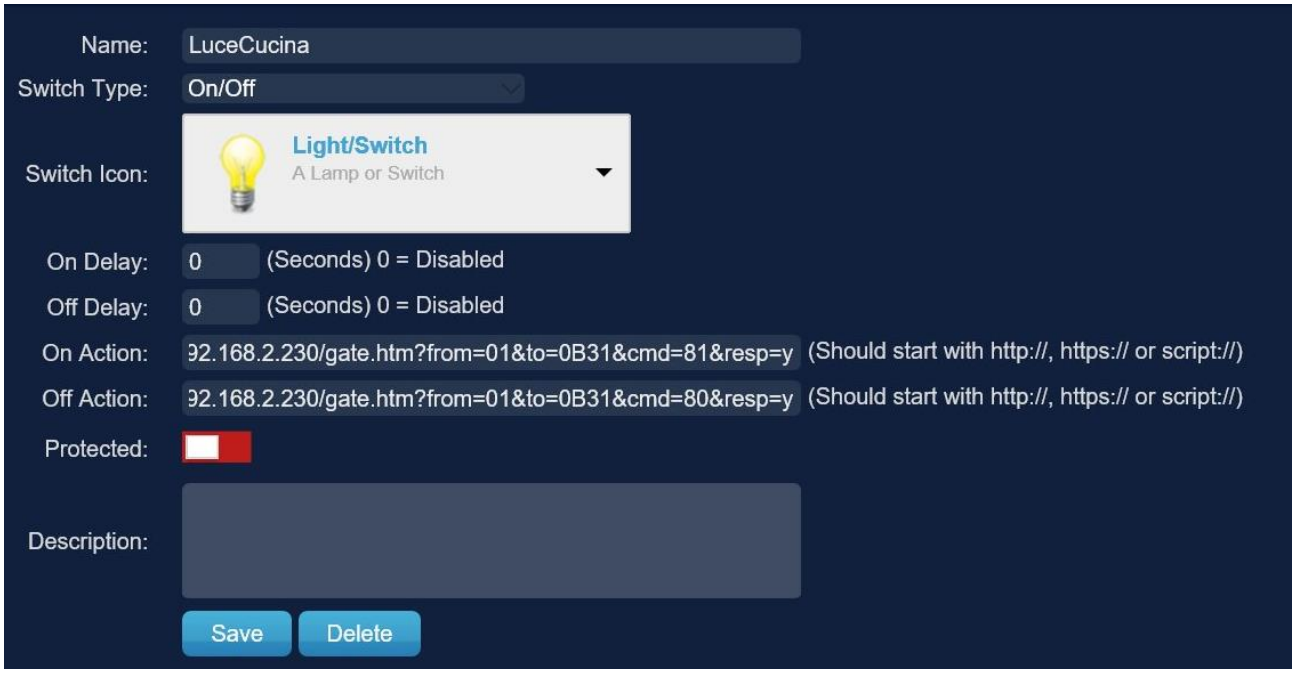


Li ho definiti di tipo "Switch". Quindi la lista dei dispositivi (Setup – devices) era questa:

Idx	Hardware	ID	Unit	Name	Type	SubType	Data			Last Seen
3	ESP_KNXGATE	00014053	1	Tapparella sala	Light/Switch	Switch	Closed	-	-	2019-01-30 19:37:18
2	ESP_KNXGATE	00014052	1	LuceSala	Light/Switch	Switch	Off	-	-	2019-01-30 19:30:03
1	ESP_KNXGATE	00014051	1	LuceCucina	Light/Switch	Switch	Off	-	-	2019-01-30 19:29:35

Ogni device è contraddistinto da un "idx" univoco. Cliccando sul bottone "switches" appare poi così:

Cliccando sul bottone “Edit” si impostano le proprietà dello switch:



Sulla casella “On action” (accendi) ho scritto il comando TCP che richiama il mio ESP_KNXGATE che è all’indirizzo 192.168.2.230, con i bytes di comando che ho scoperto sniffando

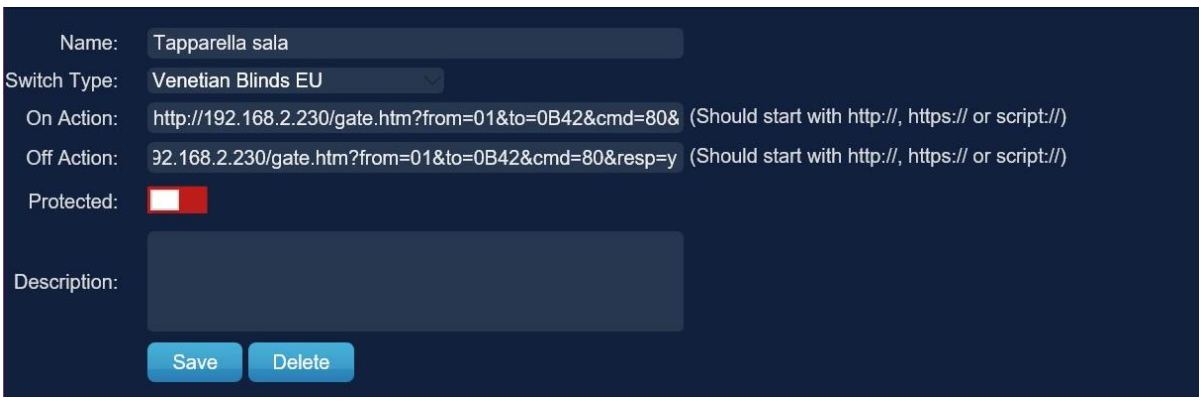
http://192.168.2.230/gate.htm?from=01&to=0B31&cmd=81&resp=y

stessa cosa per “Off action” (spegni)

http://192.168.2.230/gate.htm?from=01&to=0B31&cmd=80&resp=y

i “selector switches” consentono ulteriori comandi personalizzabili e associabili. Limite attuale: ad ogni bottone è possibile associare un solo telegramma.

Ho salvato ed ho fatto la stessa cosa per gli altri switches. Per la tapparella, su “switch type” ho scelto “Venetian blinds EU” – ciò permette di avere un’icona a tre bottoni (apri / stop / chiudi). Peccato che domoticz permetta di inserire solo le chiamate ON e OFF (manca lo STOP)…



Sulla videata degli “switches” cliccate sulla stellina di ciascuno così che diventi gialla: questi switches verranno così presentati sulla videata principale (la dashboard).

Sulla dashboard, cliccando sulla lampadina dello switch, essa si accenderà/spegnerà e contestualmente domoticz lancerà la richiesta TCP verso esp_knxgate.

Questa era la parte facile. Un po' più complicato è il viceversa, cioè far accendere o spegnere la lampadina sulla dashboard quando la lampadina viene accesa da un interruttore fisico. Qui entra in ballo la procedura di callback che viene richiesta dal parametro di chiamata &resp=y. Se ne deduce che perché venga richiamata è indispensabile fare almeno una chiamata di switch (ne basta una sola, su di uno switch qualunque).

Prima di tutto bisogna configurare esp_knxgate.

La richiesta *…./callback* serve a impostare la pagina che verrà richiamata dal gate per il callback, quando intercetta telegrammi che viaggiano sul bus. Sul browser richiamatela così (il mio esp_knxgate si trova all'indirizzo 192.168.2.230)

http://192.168.2.230/callback

Nella stringa da digitare non va impostato l'IP poiché si assume che sarà il medesimo che ha effettuato la chiamata "gate" di on/off. La stringa DEVE iniziare con il carattere ":" seguito dal numero di porta e dagli altri parametri previsti da domoticz:

:8080/json.htm?type=command&param=udevices&script=knxgate_json.lua

Al momento della chiamata di callback (di tipo GET) verranno aggiunti 4 parametri che riportato i dati del dispositivo che ha cambiato stato: "&from=xx&to=xxxx&cmd=xx

Come avete visto, viene richiamato lo script knxgate_json.lua che va personalizzato secondo le vostre esigenze e messo nella directory di domoticz denominata "**scripts/lua parsers**" . Quello che vi mostro è un esempio, quello che ho usato io nei test. Serve a convertire l'indirizzo KNX nel numero di device di domoticz (idx), e a convertire il byte di comando in una azione da effettuarsi sul device virtuale.

Ecco lo script "demo" che io ho usato e che va completato / personalizzato secondo le vostre esigenze. Come vedete manca l'interpretazione del comando "stop" per le tapparelle, non essendo previsto nel device era inutile prevederlo…. Contiene anche alcune istruzioni "print" di per se inutili che possono servire in fase di debug, poiché vengono mostrate in domoticz con il bottone setup – log.

```
-- Example of JSON parser handling GET data with the following structure

--
http://192.168.1.17:8080/json.htm?param=udevices&script=knxgate_json.lua&from=xx
&to=xxxx&cmd=xx

-- retrieve the GET params
local from = uri['from'];
local device = uri['to'];
local cmd  = uri['cmd'];
```

```
    print (" from="..from..", to="..device..", cmd="..cmd.." ")


-- UpdateDevice']='idx|nValue|sValue'
--                 idx= id device
--                 nValue=
--                 sValue=

-- 20
-- domoticz_updateDevice(1,0,'Off')

local idx;
local nCmd;
local sCmd;

if     (device == '0B71') then
    idx = 1
elseif (device == '0B73') then
    idx = 2
elseif (device == '0B77') then
    idx = 3
else
    idx = 0
end

if (cmd == '81') then
      nCmd = 1
      sCmd = "On"
elseif (cmd == '80') then
      nCmd = 0
      sCmd = "Off"
else
      nCmd = 9
      sCmd = 9
end

 print ("idx="..idx..", nCmd="..nCmd..", sCmd="..sCmd.." ")

 if ((idx > 0) and (nCmd < 9)) then
      domoticz_updateDevice(idx,nCmd,sCmd)
 end

-- Retrieve the request content
--s = request['content'];

-- Update some devices (index are here for this example)
--local id = domoticz_applyJsonPath(s,'.id')
--local s = domoticz_applyJsonPath(s,'.temperature')
--domoticz_updateDevice(id,'',s)
```

Se avete qualche monitor wifi (io uso wireshark) potete verificare i colloqui. Ho potuto constatare che esp_knxgate è abbastanza reattivo e che invece domoticz non è particolarmente veloce (forse perche l'ho installato su windows).

Su domoticz, di più non so…

DOMOTICZ - MQTT

MQTT si sta sempre più diffondendo come standard di comunicazione tra dispositivi domotici e relativi sistemi di controllo. E' basato su di un server centrale, detto "broker" a cui vengono inviati ("pubblicati") i messaggi di controllo e di stato, distinti per argomento ("topic") e che si occupa di distribuirli ai "client" che hanno "sottoscritto" i corrispondenti argomenti.

Nel nostro caso il sistema di controllo testato è stato ancora una volta DOMOTICZ, che a sua volta è stato definito "client" del broker MQTT attraverso un apposito PLUGIN
(https://github.com/emontnemery/domoticz_mqtt_discovery)

Tale plugin funzionava bene con le luci e male con dimmer e tapparelle per cui ho provveduto a correggerlo in casa, cercherò di proporre le mie correzioni all'autore. La versione da me implementata la potete trovare qui (è nella cartella esp_scsgate ma va bene anche per esp_knxgate e raspy_xxxgate):
http://guidopic.altervista.org/esp_scsgate/mqtt_discovery-development.zip

Come broker MQTT ho usato MOSQUITTO perché semplice, leggero e disponibile anche in windows (test effettuati solo in windows).

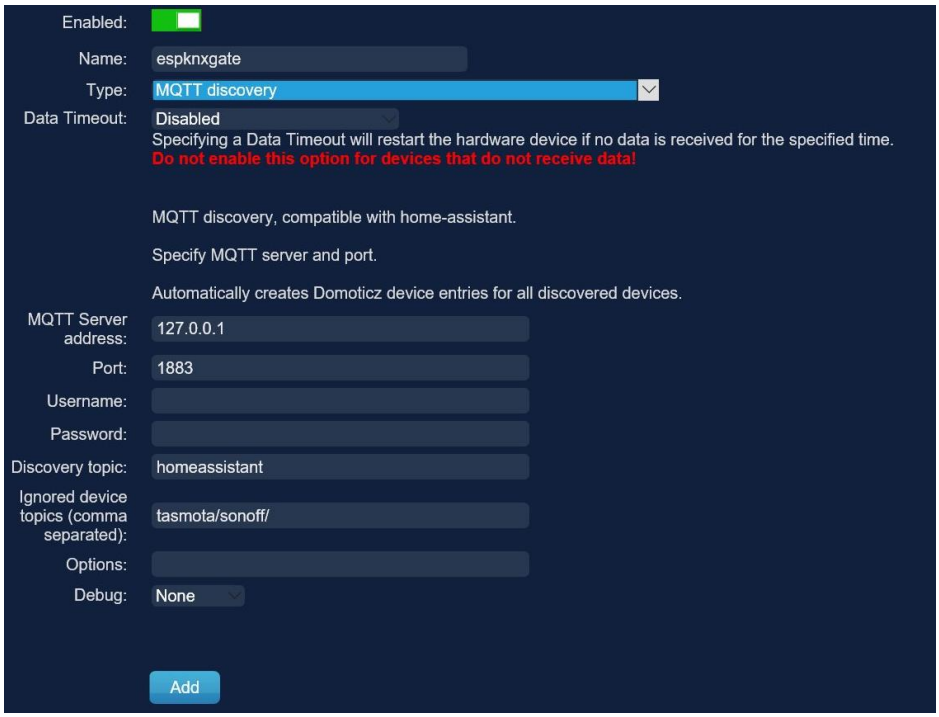
Anche ESP_KNXGATE va configurato per la connessione al medesimo broker – accertatevi sul monitor/log del broker che la connessione venga correttamente accettata.

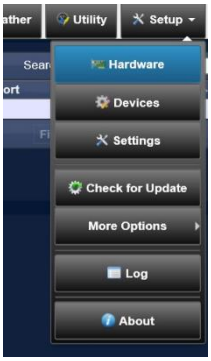
Come già detto non sono affatto un esperto di domoticz – la mini guida che segue serva a mostrarvi i passi che ho fatto e come l'ho integrato.

Primo passo: localizzate la directory di installazione di domoticz (in windows è C:\Program Files (x86)\Domoticz) – qui dovrebbe esserci una cartella "plugins" – se non c'è createla. Dentro la cartella plugins inserite la cartella "mqtt_discovery-development" contenente il plugin "plugin.py". E' indispensabile avere sul computer anche "python" – io ho dovuto usate la v.3.5.2 perché con la più recente 3.7 in windows non funzionava neppure domoticz.

Dopo aver copiato il plugin, domoticz va spento e riacceso – controllate sul log di non avere errori.

Se tutto è corretto, andando a censire nuovo hardware dovreste trovare anche l'hardware "MQTT Discovery"





Va digitato il nome (io ho usato espknxgate) e indirizzo e porta del broker. Cliccate su ADD.

I dispositivi (SWITCHES) non possono essere aggiunti manualmente, è indispensabile procedere al censimento automatico:

Connettete esp_knxgate al bus e verificate che la connessione con il broker sia avvenuta correttamente.

Richiamate la pagina di preparazione configurazione:

http://192.168.2.230/mqttdevices?request=prepare

Ora bisogna che tutti i dispositivi KNX dell'impianto lancino sul bus un messaggio di stato. Provvedete quindi ad accendere o spegnere le luci, variare la luminosità dei dimmer, azionare le tapparelle (dispositivi diversi non verranno riconosciuti). Richiamate quindi la pagina di censimento automatico:

http://192.168.2.230/mqttdevices?request=start

Per verificare se il processo è finito potete usare:

http://192.168.2.230/mqttdevices?request=query

Per fermare il processo potete usare:

http://192.168.2.230/mqttdevices?request=stop

I dispositivi vengono memorizzati anche in esp8266; per ripetere il processo senza rifare tutti i passi potete usare:

http://192.168.2.230/mqttdevices?request=resend

A questo punto refreshate la pagina di domoticz “SWITCHES” e ci troverete i vostri dispositivi, censiti con dei nomi standard corrispondenti agli indirizzi knx:

Cliccando sulla stellina essi appariranno anche sulla DASHBOARD di domoticz. Potete provarli cliccando sulla lampadina, variano lo slide dei dimmer, premendo le icone delle tapparelle.

Dopo aver così individuato la corrispondenza con i vostri dispositivi, per ciascuno potete entrare in EDIT e cambiare opportunamente il nome.

Rifacendo da capo il censimento i nuovi nomi verranno mantenuti e non ci saranno duplicazioni.

Lo stato generale di esp_knxgate può essere letto con la richiesta;

http://192.168.2.230/status

Riepilogo delle richieste http

/status visualizza la situazione attuale del dispositivo

/reset?device=esp oppure =pic oppure=mqtt (resetta il processore di esp8266 o di knxgate)

/request mappa di richiesta esecuzione comandi knx

/gate ?from=xx&to=xxxx&cmd=xx&resp=y|n

Lancia su knxgate il relativo telegramma – il parametro resp=y richiede anche tutte le successive chiamate di callback al cambio di stato dei dispositivi.

/callback configurazione callback http

/mqttconfig configurazione MQTT

/mqttdevices?request=prepare | start | stop | query

inizia/termina/interroga il censimento automatico knx

HOME ASSISTANT - MQTT

Ho installato home-assistant inizialmente su windows 10 per comodità di test, consapevole del fatto che in una installazione windows non tutte le funzionalità vengono garantite, successivamente l'ho installato su raspberry.

Preciso che la versione homeassistant era 0.85.1 – lo dico perché ho riscontrato nei fatti che homeassistant è un bellissimo progetto, ma non spicca per stabilità. Consiglio di non cambiare senza motivo la versione che avete installato perché potreste avere delle sorprese. Recentemente sono passato alla versione 0.90.1 e poi alla 0.114.0: tutto continua a funzionare.

Come broker MQTT ho usato MOSQUITTO perché semplice, leggero e disponibile anche in windows.

Primo passo: localizzate la directory contenente il file di configurazione "configuration.yaml" – editate tale file.

Aggiungete le definizioni per mqtt:

```
mqtt:
  broker: 127.0.0.1
  port: 1883
  discovery: true
  discovery_prefix: homeassistant
```

Ovviamente usate ip address e port del vostro broket mqtt.

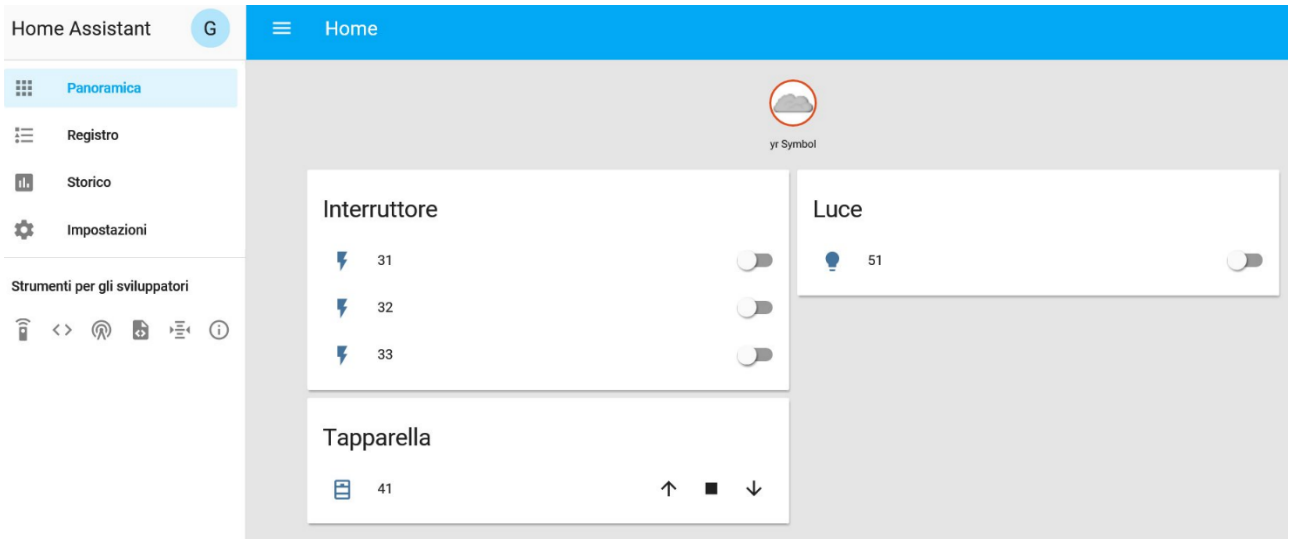
Dopo aver modificato il file, home assistant va spento e riacceso – controllate sul log di non avere errori.

Se tutto è corretto, andando nella pagina "Impostazoni -> integrazioni" dovreste vedere MQTT come configurato:

Connettete raspy_knxgate al bus.

Seguendo le istruzioni iniziali, preparate il file "knxconfig" e poi lanciate il programma "knxdiscovery"

A questo punto refreshate la pagina iniziale (“panoramica”) di home assistant e ci troverete i vostri dispositivi, censiti con dei nomi standard, corrispondenti agli indirizzi KNX:



Cliccando sulle icone potrete verificarne la funzionalità.

Secondo le indicazioni dell’help, mqtt discovery avrebbe dovuto aggiungere automaticamente le medesime definizioni anche sul file “configuration.yaml”. Non è stato così: forse non ho fatto qualche settaggio, forse la versione windows non funziona bene, o forse ho capito male io.

Invece ciò non avviene e quindi spegnendo e riaccendendo homeassistant le definizioni scompaiono: per renderle permanenti bisogna invece modificare manualmente configuration.yaml, approfittandone per mettere nomi effettivi:

```
mqtt:
  broker: 127.0.0.1     (se sulla stessa macchina, scrivere  localhost )
  port: 1883
  discovery: true
  discovery_prefix: homeassistant

switch:
  - platform: mqtt
    name: Luce cucina
    command_topic: "knx/switch/set/0B31"
    state_topic: "knx/switch/state/0B31"

  - platform: mqtt
    name: Luce sala
    command_topic: "knx/switch/set/0B32"
    state_topic: "knx/switch/state/0B32"

  - platform: mqtt
    name: Luce bagno
    command_topic: "knx/switch/set/0B33"
    state_topic: "knx/switch/state/0B33"

cover:
```

(tapparella gestita up/down/stop)

```
  - platform: mqtt
    name: Sala grande
    command_topic: "knx/cover/set/0B41"
```

```
    state_topic: "knx/cover/state/0B41"
```

(tapparella gestita a percentuale)

```
  - platform: mqtt
    name: Sala piccola
    command_topic: "knx/cover/set/0B42"
    state_topic: "knx/cover/state/0B42"
    set_position_topic: "knx/cover/setposition/0B42"
    position_topic: "knx/cover/value/0B42"

light:
  - platform: mqtt
    name: Dimmer Sala
    command_topic: "knx/switch/set/0B51"
    state_topic: "knx/switch/state/0B51"
    brightness_command_topic: "knx/switch/setlevel/0B51"
    brightness_state_topic: "knx/switch/value/0B51"
```

SE conoscete già gli indirizzi KNX dei vostri dispositivi potete tranquillamente evitare tutto il giro di “discover” e censirli direttamente nel file “configuration.yaml”

ATTENZIONE.

Può capitare che home-assistant, tra discover e manipolazioni varie, possa ad un certo punto segnalarvi su alcuni dispositivi l’errore: “Questa entità () non ha un ID univoco, pertanto le sue impostazioni non possono essere gestite dall’interfaccia utente…”

Se vi capita, semplicemente aggiungete sul dispositivo, in configuration.yaml, la direttiva “unique_id” riportando del parametro la medesima scritta del parametro “name” con gli spazi sostituiti da underscore (_); esempio:

```
  - platform: mqtt
    Unique_id: Sala_grande
    name: Sala grande
    command_topic: "knx/cover/set/0B41"
    state_topic: "knx/cover/state/0B41"
```

HOME ASSISTANT (HASSIO) - MQTT

L'installazione su home-assistant basato su Hassio presenta qualche attenzione in più riguardante soprattutto il broker Mosquitto che se non installato correttamente può pregiudicare il corretto funzionamento.

Per la corretta installazione (o reinstallazione) di Mosquitto in ambiente Hassio fate riferimento a questa guida: https://indomus.it/guide/configurare-correttamente-mqtt-su-hassio-versione-addon-dalla-v3-in-poi/

È molto particolareggiata e seguendola attentamente riuscirete ad installarlo correttamente.

A questo punto vale tutto quanto detto per l'installazione su home-assistant "raspbian".

Dovrete censirli manualmente in configuration.yaml (attenzione, SOLO i dispositivi, NON il server mqtt che viene invece gestito da hassio).

Altra attenzione: usando questa procedura il broker mosquitto verrà definito accessibile solo con user e password, che andranno quindi usati coma parametri di knxdiscover e knxgate_x.

OPENHAB

Non conosco openHAB, gli esempi che seguono mi sono stati mandati da Omar, che ringrazio:

file “broker.things”:

```
mqtt:broker:RpiBroker " Comandi bticino" [ host=" IP_SERVER_MQTT ",
secure=false, username="USER", password="PASSOWORD" ]
```

file “interruttori.items”

```
Switch          GF_LivingDining_Light         "Luce Sala"
<light>           (GF_LivingDining, gLight)         ["Lighting",
"Switchable"]   {channel="mqtt:topic:RpiBroker:interruttori:lamp1",
alexa="PowerController.powerState"}

Rollershutter   GF_LivingDining_Shutter       "Tapparella Sala"
<rollershutter>   (GF_LivingDining, gShutter)       ["Rollershutter"]
{channel="mqtt:topic:RpiBroker:tapparelle:roller1", alexa="Blind" [
actionMappings="Close=ON,Open=OFF,Lower=ON,Raise=OFF",
stateMappings="Closed=100,Open=0"]}
```

file “mqtt.things”

```
Bridge mqtt:broker:RpiBroker "Comandi bticino" [ host="IP_SERVER_MQTT",
secure=false, user="USER", password="PASSWORD" ]
{
    Thing topic interruttori "Interruttori bticino " {
    Channels:
    Type switch : lamp1 "Luce Sala" [ stateTopic="knx/switch/state/0B25",
commandTopic="knx/switch/set/0B25", on="ON", off="OFF" ]
    ........ Altri interruttori
    }
    Thing topic tapparelle "Tapparelle bticino " {
    Channels:
        Type rollershutter : roller1 "Tapparella Sala" [
stateTopic="knx/cover/value/14",
commandTopic="knx/cover/setposition/0B14",
transformationPattern="JS:invertpercent.js",
transformationPatternOut="JS:invertpercent.js" ]
     .......... Altre tapparelle
    }
}
```

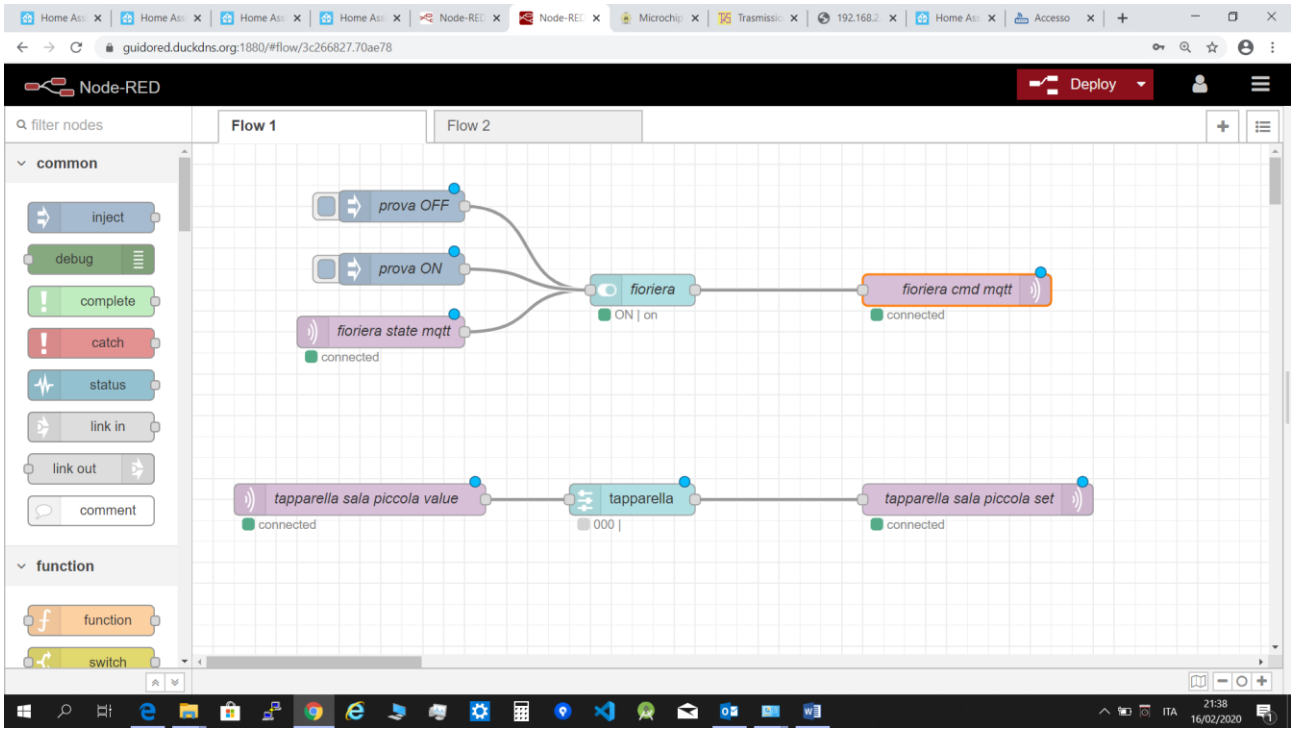
File “invertpercent.js”: è una funzione indispensabile per “rovesciare” la percentuale di apertura delle tapparelle, che in openhab viene gestita al contrario (percentuale di chiusura).

```
(function(i)  {
   var percent_shelly = parseInt(i,10);
   var percent_oh = (100.0- percent_shelly);
   return percent_oh.toFixed(0);
})(input)
```

NODE-RED

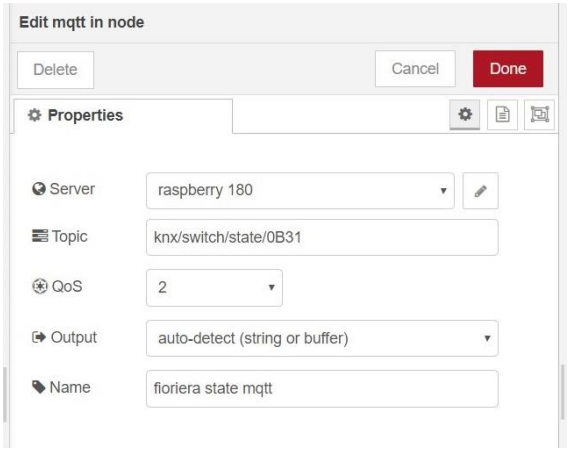
Node-red è un altro sistema domotico diffuso e apprezzato che consente una facile interazione attraverso un broker mqtt.

Ecco un esempio di definizione di uno switch luce e di una tapparella gestita a percentuale (ovviamente i modi di definizione possono essere molteplici):

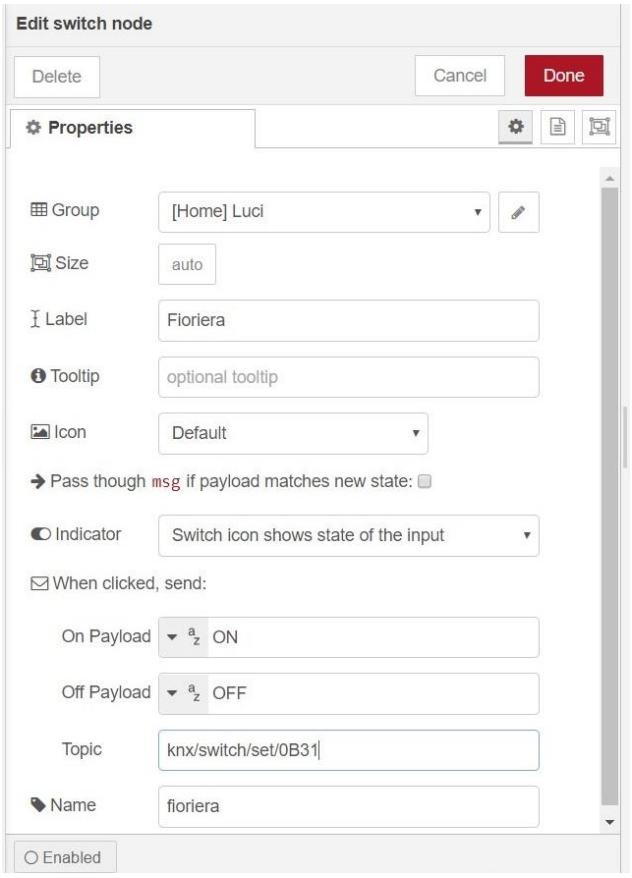


I blocchi “inject” servono solo come test.

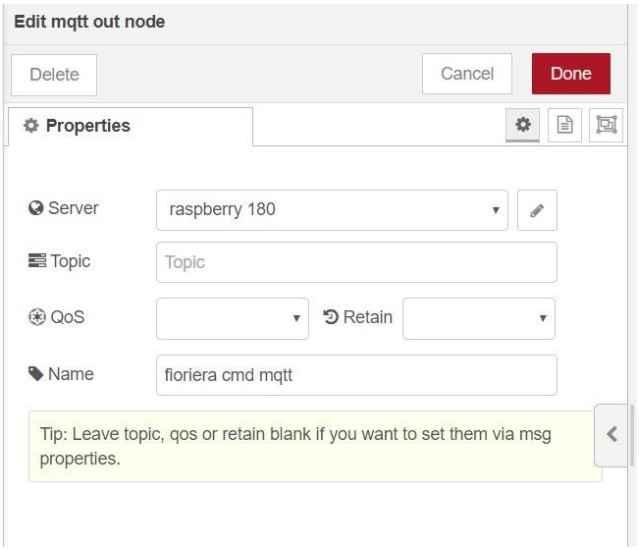
Il blocco che legge lo stato della luce (fioriera state mqtt) è un blocco “mqtt-in”:



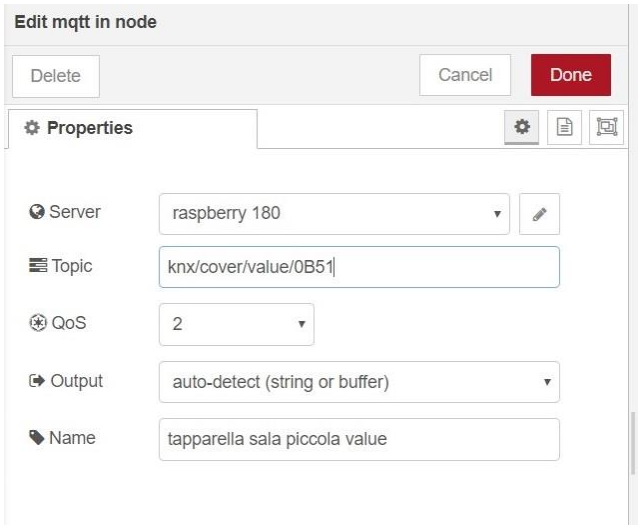
Il blocco luce (fioriera) è un blocco “switch”:



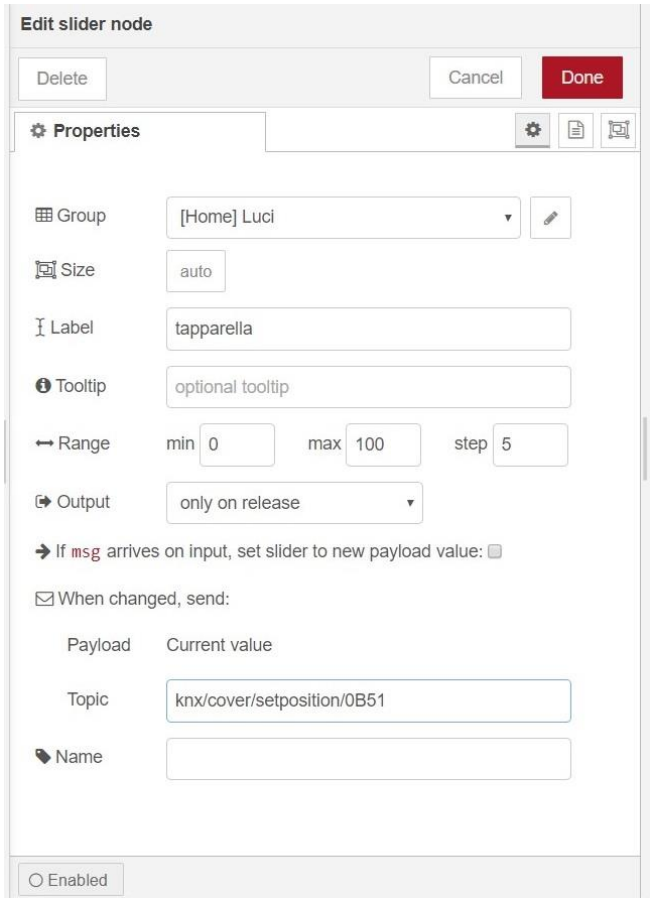
Il blocco di comando è un “mqtt-out”:



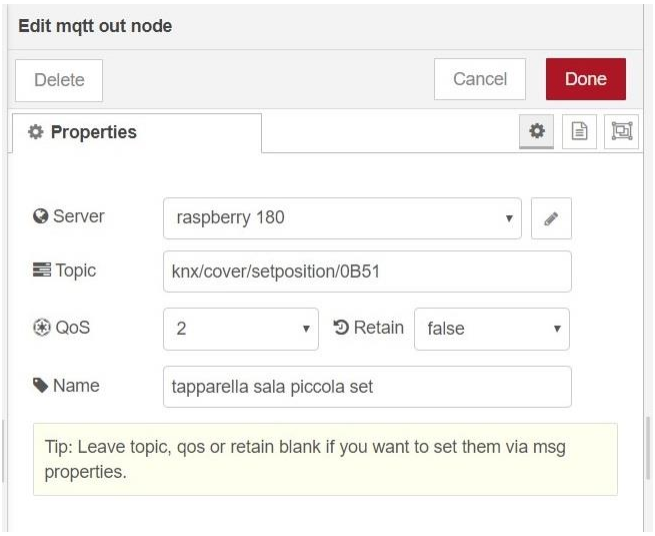
Il blocco che legge lo stato della tapparella è un blocco “mqtt-in”:



Il blocco tapparella è un blocco “slider”:



Il blocco di comando è un “mqtt-out”:



DISCLAIMER

Non posso garantire che l'integrazione con Domoticz, con Home-assistant e con Alexa sia perfetta in quanto il test si è limitato alle prove suddette, né posso garantire che le prossime versioni di Domoticz e Home assistant e Alexa (e Philips hue) mantengano inalterata questo tipo di interfaccia.

Non mi ritengo responsabile di danni che potreste causare al vostro impianto domotico per imperizia o negligenza o per non aver adottato le indispensabili precauzioni di sicurezza.