

DIY: un innaffiatoio per l'Home Automation.

[Prestazioni](#)

[Caratteristiche](#)

[materiale](#)

[strumenti](#)

[software](#)

[Costruzione](#)

A)[preparazione della scheda Sonoff-basic](#)

B)[montaggio del timer](#)

C)[logica di funzionamento \(script\)](#)

[Sonoff+esp_MQTT: MQTT messaggi](#)

[Sonoff+esp_MQTT: node-red client](#)

[Sonoff+esp_MQTT: Android client](#)

Stufo di cambiare il timer dell'impianto di irrigazione del terrazzo ogni 2 anni, ho deciso di progettare un timer: ovviamente IoT e basato su MQTT.

Il risultato è questo progetto completo DIY di un timer per irrigazione, migliore di molti timer commerciali, progettato e realizzato secondo lo stato dell'arte.

(vdi https://github.com/msillano/sonoff_watering)

Questo timer contiene uno STA (client WIFI), un AP (non usato), un client MQTT ed un server (broker) MQTT, fornito di logica di funzionamento programmabile (script): in pratica un mini *mosquitto* + *node-red* on-board!

Quindi in questo progetto la logica di funzionamento è tutta interna: non è richiesto un server MQTT esterno attivo 24/7 (infatti ritengo veramente eccessivo un server 24/7 per innaffiare un terrazzo!). Il timer è totalmente autonomo, ma nel contempo sempre compatibile con altri client e broker MQTT standard esterni.



Una sonda segnala lo stato del terreno (asciutto/bagnato, con soglia regolabile) e condiziona l'innaffiamento.

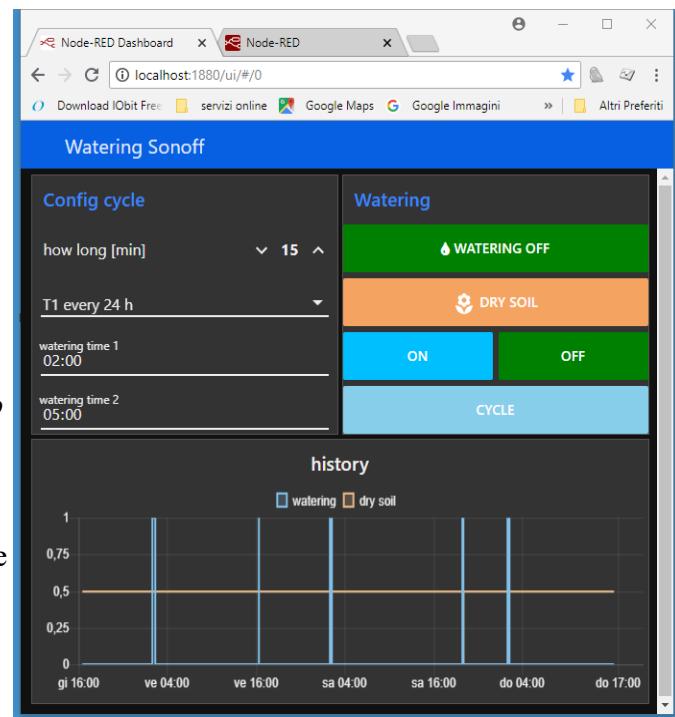
Il monitoraggio e la configurazione del timer sono effettuati con client MQTT con dashboard, sia su PC che su smartphone Android. L'accesso è locale via WIFI, ma può essere reso globale da Internet, usando un servizio gratuito di DNS dinamico (esempio **noip**: <https://www.noip.com/>).

Pur usando componenti aggiornati e di qualità, questo timer è molto economico: costo totale inferiore a 15 € (18 US\$)! Tutto questo è reso possibile dalla felice unione di due prodotti fenomenali:

- 1) [Sonoff basic](#), economica scheda WIFI con alimentatore, processore EPS8266 e relais
- 2) [esp_MQTT](#), un firmware per EPS8266 creato da Martin-ger, che trasforma *Sonoff-basic* in un carrarmato.

Prestazioni:

- Due orari T1 e T2 (hh:mm)
- Modalità di funzionamento:
 - (0) manuale (ON, OFF)
 - (1) ciclo a T1 ogni 48 ore,
 - (2) ciclo a T1 ogni 24 ore
 - (3) cicli a T1 e T2 ogni 24 ore
 - (4) start a T1, end a T2 ogni 24 ore
 - ✓ *ciclo (1, 2, 3): avviato solo se suolo asciutto (dry)*
 - ✓ *lungo (4): ON quando suolo asciutto (dry), OFF quando suolo bagnato (soaked).*
- Durata ciclo innaffiamento: 1min... 3 ore (1..180 min)
- Comandi: ON, OFF (incondizionati) CYCLE (immediato, condizionato)



Caratteristiche:

- ✓ *Autonomo*: contiene un server MQTT con la logica di funzionamento (script).
- ✓ *Compatibile*: può essere usato con client MQTT e server MQTT standard, dispone di un completo set di comandi e di informazioni via MQTT
- ✓ *Autostart*: conserva status e configurazione in flash RAM: in caso di reset o blackout ripristina lo status precedente in modo autonomo.
- ✓ *NTP-client*: richiede il collegamento al WIFI domestico, con accesso ad Internet, per avere l'ora esatta. Comandi per ora estiva/invernale.
- ✓ *Console seriale*: per installazione e debug, via WIFI (telnet) o seriale (COM)
- ✓ *Aggiornamento OTA*: la logica di funzionamento (script) è aggiornabile via OTA (Over The Air), senza spostare od aprire il timer.
- ✓ *Client MQTT*: attualmente 2 opzioni per monitoraggio e configurazione:
 - Client completo su PC (*node-red*)
 - Client ridotto su Android (app *IoT MQTT Dashboard*)
- ✓ *Alimentazione*: 220 V, elettrovalvola a 220 V (***attenzione !!!***)
- ✓ *Consumo in stand-by*: non misurabile (< 1 W)
- ✓ *Consumo in ON*: 6 W
- ✓ *Custodia*: stagna e trasparente, 3 LED: **power on, irrigazione on, suolo bagnato**

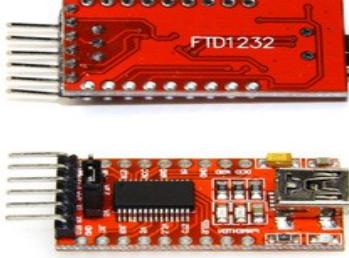
Note

1. *A seconda del tipo di impianto di innaffiamento, sono disponibili tempi brevi o tempi molto lunghi (impianti a goccia).*
2. *Il posizionamento della sonda influisce sul funzionamento: io l'ho posizionata in un punto non innaffiato ma bagnato dalla pioggia. E' anche possibile situarla in un punto innaffiato, per creare una sorta di feedback.*
3. *E' possibile comandare una piccola pompa (220 V, max 500 W) al posto dell'elettrovalvola.*

materiale

	Sonoff basic WiFi wireless switch https://www.itead.cc/smart-home/sonoff-wifi-wireless-switch.html	\$ 4.85
	IP66 Waterproof Case https://www.itead.cc/smart-home/sonoff-wifi-wireless-switch.html	\$ 3.90
	Soil Hygrometer Humidity Detection Module https://it.aliexpress.com/item/Free-shipping-soil-the-hygrometer-detection-module-robot-intelligent-car-soil-moisture-sensor-for-arduino/32281470046.html	\$ 0.49
	220V AC Electric Solenoid Valve Magnetic N/C Water https://it.aliexpress.com/item/220V-AC-Electric-Solenoid-Valve-Magnetic-N-C-Water-Air-Inlet-Flow-Switch-1-2/32691601093.html	\$ 3.97
	5 pin per c.s.	\$ 0.10
	3 Female To Female Jumper Cable Dupont	\$ 0.10
	FE-SP-HDR23-100/22 Filtro per carico induttivo (Surge protector)	\$ 4.16

strumenti

	Stampante 3D (Ho usato una 3Drag) (optionale, vedi testo)
	Pistola per colla calda
	Saldatore a punta sottile (uso PBLK 6 A1 Parkside)
	FTDI USB 3.3 V 5.5 V Modulo Adattatore Seriale TTL per Arduino (US\$ 2.41)

software

esp_MQTT	https://github.com/martin-ger/esp_mqtt
esptool	https://github.com/espressif/esptool
puTTY	https://putty.org/
e3DHW	https://www.thingiverse.com/thing:2860353
OpenSCAD	http://www.openscad.org/
Notepad++	https://notepad-plus-plus.org/
wpp_pampa	http://www.wipenpack.com/en/index.php
node-red	https://nodered.org/docs/getting-started/installation
IoT MQTT Dashboard	https://play.google.com/store/apps/details?id=com.thn.iotmqtdashboard

Costruzione



A) preparazione della scheda Sonoff-basic

1 Estrarre la scheda Sonoff -basic dal contenitore

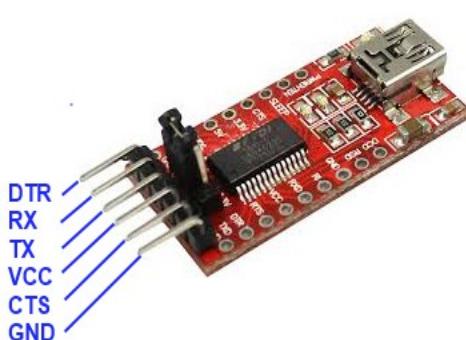
2 Saldare un connettore a 5 pin nelle piazzole di Sonoff-Basic (serve per caricare il firmware e poi per connettere il sensore di umidità)

- 1.
- 2.
- 3.
- 4.



nota: versioni precedenti di Sonoff-basic presentano solo 4 pin (manca GPIO 14). In questo caso si può usare RX (GPIO 1) al posto di GPIO 14, modificando lo script (vedi https://github.com/martin-ger/esp_mqtt/issues/28).

3 Preparare il collegamento a 4 fili FTDI ↔ Sonoff:



FTDI: VCC → 3.3 (con interruttore) Sonoff

RX → TX

TX → RX

GND → GND

nota: per istruzioni complete consultare: <http://randomnerdtutorials.com/how-to-flash-a-custom-firmware-to-sonoff/>

4 Installazione di *esp_MQTT*, in Windows:

esp_MQTT non deve essere compilato: martin_ger, oltre a pubblicare i sorgenti, pubblica anche i file 'bin' pronti per essere caricati su Sonoff.

- Scaricare ed installare esptool (<https://github.com/espressif/esptool>, necessario per copiare il firmware,
- Scaricare da https://github.com/martin-ger/esp_mqtt/tree/master/firmware i 2 file 'bin' con il firmware compilato, metterli nella stessa dir di *esptool.py*
- **Disconnettere Sonoff dalla rete a 220V!**
- Collegare FTDI via USB al PC.
- Controllare il numero di porta usato (Pannello di controllo, Sistema, Gestione dispositivi, Porte) Nel mio caso: COM6.
- Configurare FTDI per 3.3 V e poi collegare Sonoff.
- Tenere premuto il pulsante su Sonoff *prima* di fornire alimentazione a Sonoff (usare l'interruttore presente sul collegamento FTDI-Sonoff). Rilasciare il pulsante su Sonoff 1 o 2 secondi dopo aver fornito tensione.
- Usare il seguente comando per copiare il firmware in Sonoff-basic (COM6 può cambiare). Io uso un piccolo file BAT (vedi *esptool/write-esp-Sonoff.bat*):

```
esptool.py --port COM6 write_flash -fs 1MB -fm dout 0x000000 0x000000.bin 0x10000  
0x10000.bin
```

nota: Il firmware originale Itead è ripristinabile con qualche complicazione (vedi <https://wiki.almeroth.com/doku.php?id=projects:sonoff>). Ma in genere non è necessario: infatti lo script "script.sonoff" di martin-ger emula perfettamente il funzionamento originale di Sonoff Basic (vedi – https://github.com/martin-ger/esp_mqtt/tree/master/scripts).

nota: Non sempre la scrittura riesce al primo colpo. Provare più volte. Il problema è spesso imputabile alla poca corrente fornita da FTDI a 3.3V. Una semplice soluzione (per me ha funzionato) è quella di inserire un condensatore elettrolitico tra VCC e GND (ho usato 100 µF). Una soluzione più drastica è usare un alimentatore separato a 3.3 V (e.g. uno step-down 5→3.3).

NON ALIMENTARE A 220V Sonoff quando FTDI è collegato!! Una svista (un cacciavite che cade sul circuito) e il vostro PC (nel miglior caso) è da buttare! Ma potreste anche rischiare la vita!!

nota: Per maggiori informazioni, per Linux etc. vedi martin-ger: "Building and Flashing": https://github.com/martin-ger/esp_mqtt

5 Primo run, configurazione

Quando i messaggi ci informano della corretta scrittura della memoria flash, spegnere e riaccendere Sonoff con l'interruttore (*senza* premere il pulsante), lasciando collegato FTDI.

La configurazione di base può essere fatta ora con la console seriale.

Usiamo *putTY* (<https://putty.org/>), con la configurazione: 'serial', COM6, 115200.

Premendo [ENTER] deve apparire il prompt '*CMD>*' di *esp_MQTT* che ci conferma il corretto funzionamento. Ora possiamo dare tutti i comandi console accettati da *esp_MQTT* (vedi https://github.com/martin-ger/esp_mqtt)

- Per i nostri scopi, vogliamo che **Sonoff+esp_MQTT** si connetta come STA al WIFI router casalingo:

```
CMD>set ssid <your_home_router's_SSID>  
CMD>set password <your_home_router's_password>  
CMD>set ap 0
```

```

CMD>set speed 160
CMD>set npt_server 1.it.pool.ntp.org          (Italia)
CMD>set npt_timezone 2      (Italia, 2: ora estiva, 1: ora invernale)
○ settaggio del server MQTT (per semplicità tralasciamo ogni configurazione di sicurezza):
  CMD>set mqtt_host <hidro_name>
  CMD>save
  CMD>reset

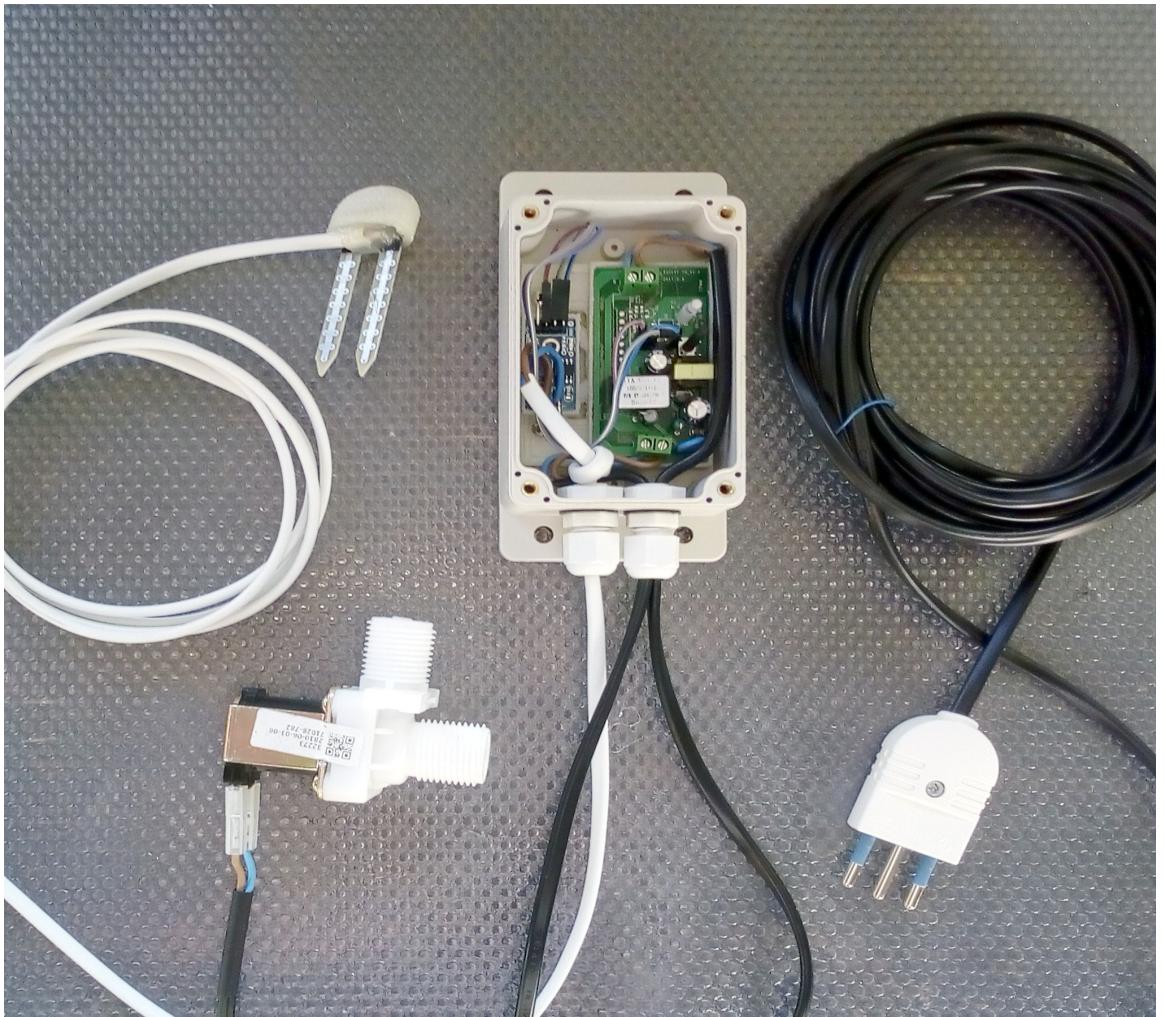
```

Dopo il reset **Sonoff+esp_MQTT** si conserverà al router indicato nella configurazione.

- Controlliamo sul router la connessione e l'indirizzo di **Sonoff+esp_MQTT**: sul router rendiamo fisso l'indirizzo di **Sonoff+esp_MQTT** (esempio: 192.168.0.53).
- D'ora in poi la console di **Sonoff+esp_MQTT** è *attivabile in remoto da ogni nodo della rete all'indirizzo fisso, con un client telnet* (e.g. puTTY) via tcp port 7777 per ricevere comandi e per test.
- Il server MQTT è raggiungibile allo stesso indirizzo fisso, port 1883 (default).
- nota: *dopo alcuni comandi (e.g. 'reset') occorre chiudere e riaprire la sessione in puTTY.*

Possiamo scollegare definitivamente FDTI, e controllare il funzionamento corretto di **Sonoff+esp_MQTT** collegato a 220V con la console remota (**attenzione!**, meglio rimettere Sonoff nella sua custodia). La scheda Sonoff è pronta: lo script con la logica di funzionamento sarà caricato via OTA dopo il montaggio.

B) montaggio del timer

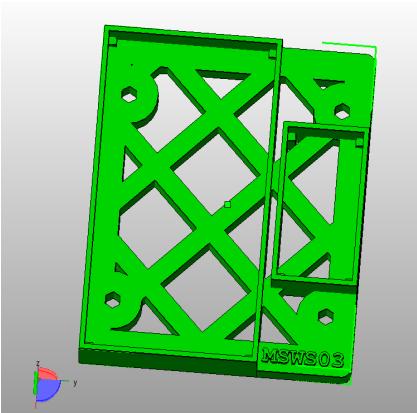


Questo montaggio è effettuato seguendo la metodologia e3DHW, anzi questo progetto è stato proprio utilizzato come esempio in Thingiverse.



Obiettivo di e3DHW è quello di fornire metodologie professionali per la realizzazione dell'hardware di apparecchiature elettroniche DIY, basate da su stampa 3D, inclusa la semplice integrazione di boards e schede come in questo caso.

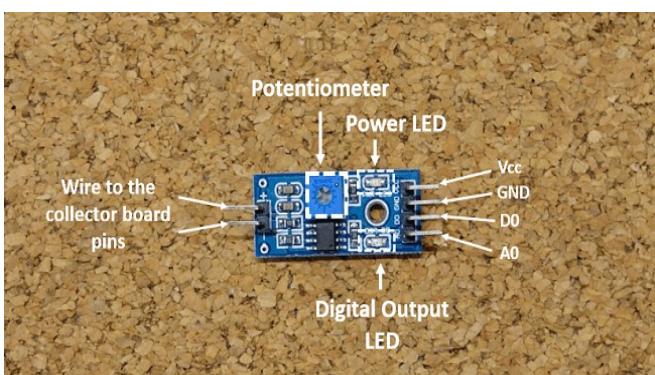
<https://www.thingiverse.com/thing:2860353>



La scheda Sonoff e la scheda del rilevatore di umidità sono posizionate su di un supporto ad hoc ottenuto con la stampa 3D, fissato poi al contenitore stagno con due viti autofilettanti.

Il file 3d/e3DHW_base_lib.1.1.scad contiene proprio questo supporto come esempio. E' anche disponibile il file 3d/e3DHW_base_lib.stl. (Il file stl è stampabile, il file scad è il progetto, modificabile con [OpenSCAD](#))

Note: *senza accesso ad una stampante 3D questa soluzione può essere simulata tagliando della plastica rigida secondo le misure del modello: dimensioni 67 x 56 mm, distanze tra i fori: 43 e 44 mm. Le due schede possono essere fissate alla plastica con colla calda. Ma ogni serio hobbista elettronico dovrebbe avere una stampante 3D !!*



Posizionate ed eventualmente fissate con colla calda i circuiti, si passa alla cablatura, consistente in 3 cavetti interni con connettori femmina-femmina, e tre cavi bipolari esterni:

- 1 – da sonda-Vcc a Sonoff-3.3V
- 2 – da sonda-GND a Sonoff-GND
- 3 – da sonda-DO a Sonoff-GPIO 14

Inoltre:

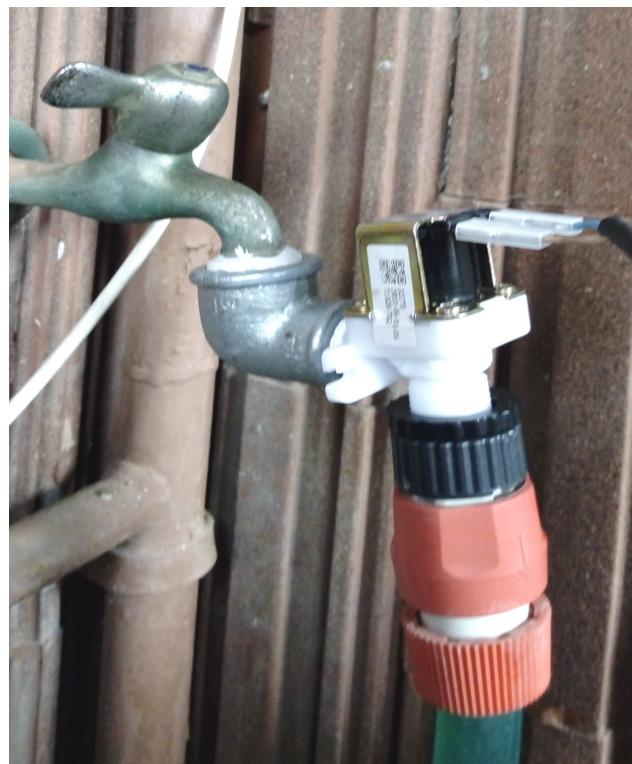
4 - il cavo che collega la sonda esterna (bianco nelle foto) è saldato direttamente ai 2 pin.

Anche la sonda è saldata, poi inserita in una protezione superiore e fissata con colla calda.
(La protezione del sensore è stampata 3D.
Vedi file 3d/sensor.scad e 3d/sensor.stl).

5 – Il cavo che porta la 220 (nero) è collegato ai due morsetti IN di Sonoff

6 – Il soppressore di disturbi e il cavo che collega l'elettrovalvola (nero) sono collegati in parallelo ai due morsetti OUT di Sonoff.

Per quanto riguarda l'elettrovalvola, nella foto vedete la mia soluzione, tolta la copertura esterna, ricavata da un contenitore in plastica.



C) logica di funzionamento

Il file “ws0103.eub” è lo script usato per il nostro Timer.

Il modo più semplice per caricare OTA uno script in **Sonoff+esp_MQTT** è quello di usare un server WEB. In fase di sviluppo è comodo avere il server locale, nel PC principale. Nel mio caso ho usato il server portatile “*wpp_pampa*” <http://www.winpenpack.com/en/index.php> contenuto in una chiavetta USB “*winPenPack*”. Altrimenti si può usare un qualunque server *WAMP/LAMP* oppure anche un server remoto su Internet, ad esempio scaricando direttamente da Gitub.

Per creare lo script ho usato *Notepad++* (<https://notepad-plus-plus.org/>), ottimo text editor per programmatori. Nota: *il file script deve essere codificato ANSI*.

Usando la console remota (*puTTY*) dare il comando 'CMD>script <url_to_script>' Per comodità il comando da usare sulla console è scritto nella seconda riga del file: usare copia-incolla.

Per rimanere nei limiti di memoria (4'000 byte), nello script ho:

- 1) eliminato l'indentazione
- 2) ridotti al minimo i commenti
- 3) limitatati i 'println' di debug
- 4) ridotti i comandi 'config' iniziali
- 5) usati nomi di 2-3 lettere per le variabili.
- 6) riusato la variabile \$tmp più volte.
- 7) eliminati i controlli sui valori di configurazione ricevuti

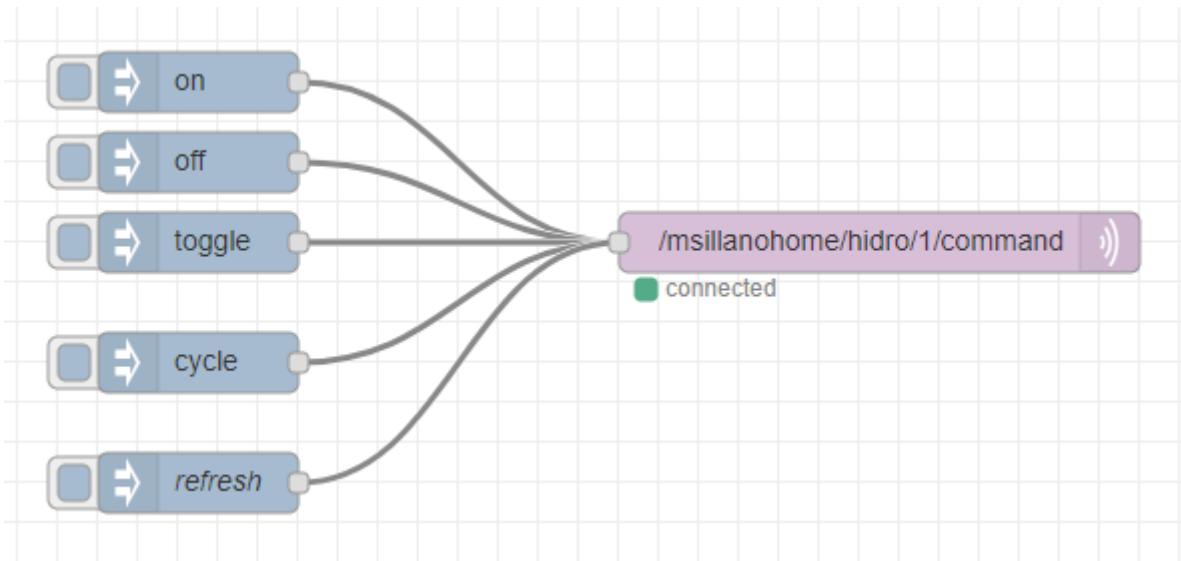
Con queste avvertenze lo script è circa 3780 byte (per una versione con tutti i commenti vedi file ws0103-source.pdf).

Personalizzazioni necessarie:

```
2      % telnet: script http://192.168.178.23:85/www/sonoff/ws0103.eub
4      config @6 /msillanohome/hidro/1
```

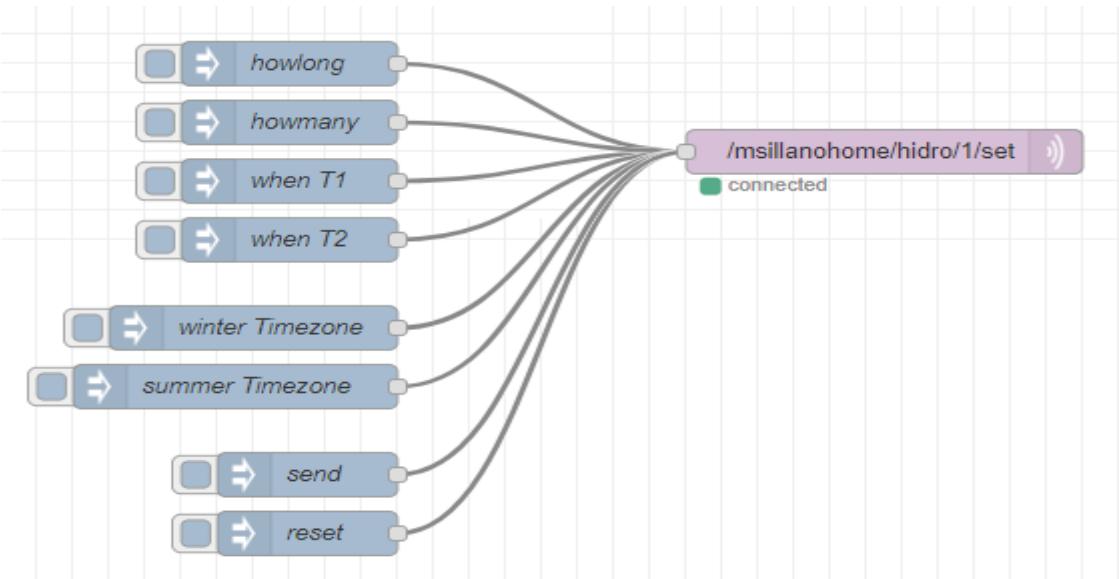
- adattare all'URL usata
- modificare topic ed ID

Sonoff+esp_MQTT: MQTT messaggi



Subscribe Comandi: /msillanohome/hidro/<num>/command

- on : innaffiamento ON non condizionato
- off : innaffiamento OFF non condizionato
- toggle : inverte lo stato di innaffiamento, non condizionato
- cycle : ciclo immediato di innaffiamento (condizionato)
- send : (refresh) ri-invia gli ultimi *satus/watering* e *status/moisure* memorizzati (retained)



Subscribe Configurazione: /msillanohome/hidro/<num>/set

- {“data” : “long”, “value” : <n>} : durata ciclo innaffiamento, in minuti (1..180)
- {“data” : “mode”, “value” : <n>} : (howmany) modo di funzionamento (0..4)
- {“data” : “when1”, “value” : “<HH:MM>” } : orario T1
- {“data” : “when2”, “value” : “<HH:MM>” } : orario T2
- {“data” : “timezone”, “value” : <h>} : set timezone, per orario estivo/invernale
- {“data” : “send”, “value” : none} : comando di ri-invio info status-configurazione
- {“data” : “reset”, “value” : none} : comando di reset immediato

nota: *I valori inviati sono cablati nelle proprietà dei nodi quindi questo flow serve solo da test.*

Publish Status-configurazione: /msillanohome/hidro/<num>/status/config
(dopo ogni modifica, dopo 'config/send')

```
{"run" : <num_run>, "long" : <min>, "tx" : ["<HH:MM>", "<HH:MM>"], "mode" : <n>}
```

Publish Status-watering: /msillanohome/hidro/<num>/status/watering
(dopo ogni comando, dopo 'command/send')

```
{"Time" : "<timestamp>", "watering" : <0|1>} : 0 = off, 1 = on
```

Publish Status-moisure: /msillanohome/hidro/<num>/status/moisure
(dopo ogni cambiamento, dopo 'command/send')

```
{"Time" : "<timestamp>", "moisure" : <0|1>} : 0 = bagnato, 1 = secco
```

note:

/msillanohome/hidro/ definisce il tipo di dispositivo (in “ws0103.eub”, si può cambiare).

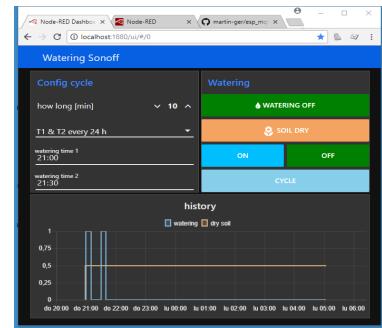
<num> numero del dispositivo (in “ws0103.eub”, si può cambiare).

Sonoff+esp_MQTT: node-red client

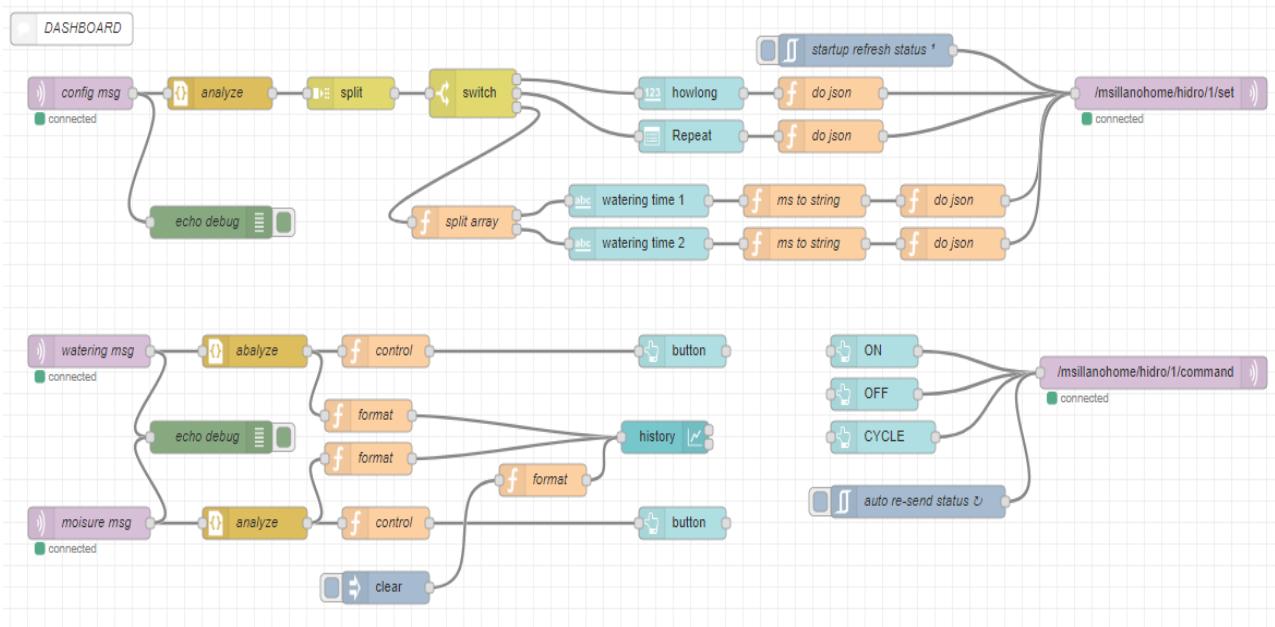
Questo client permette di monitorare lo stato del timer e la sua completa configurazione.

E' anche disponibile un grafico storico (se si lascia acceso il PC con *node-red*) degli ultimi 3 giorni.

Non è stata implementata la memorizzazione dei dati su DB, ma, se desiderata, è semplice da aggiungere (purtroppo richiede un server on 24/7).



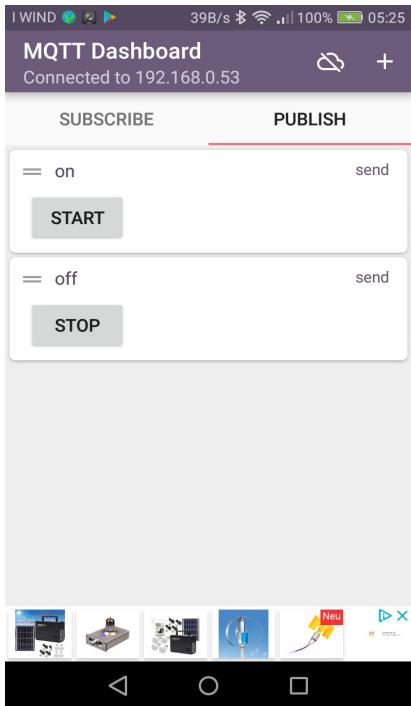
Flow completo con *node-red* (per installazione di *node-red*, vedi <https://nodered.org/docs/getting-started/installation>).



Node-red può essere installato su ogni PC (Win/Linux) ed anche su *Raspberry pi 3* se si desidera un computer dedicato a basso costo:

- Se non installata di default, importare anche la library 'node-red-dashboard'.
- Copiare nel clipboard il contenuto del file "node-red/ws103-client-red.txt" ed importarlo in una nuova scheda del dashboard di *node-red*.
- Configurare l'accesso a **Sonoff+esp_MQTT**.

Sonoff+esp_MQTT: Android client



Mancando sul timer ogni comando immediato, questo semplice client permette di controllare l'irrigazione da qualunque punto della casa usando uno smartphone.

Per questo client ho scelto “*IoT MQTT Dashboard*”

<https://play.google.com/store/apps/details?id=com.thn.iotmqtdashboard>

free, ma ci sono molte altre alternative. Cercare in *Google Play*.

Basta configurare come broker **Sonoff+esp_MQTT**, ed aggiungere (publish) almeno i due comandi ON e OFF.

Esempio:

Topic: /msillanohome/hidro/1/command

Value to publish: on

Button text: START

Nota: *il tempo di risposta è lento, si possono perdere dei messaggi di aggiornamento dei dati (subscribe), ma in questa applicazione non è critico.*

Nota: Ritengo si trovino app con funzione di MQTT+dashboard anche per S.O. diversi da Android.