DIY: a watering can for Home Automation.

Tired of changing the terrace irrigation timer every 2 years, I decided to design a timer: obviously IoT and based on MQTT.

The result is this complete DIY project of an irrigation timer, better than many commercial timers, designed and manufactured according to the state of the art. https://github.com/msillano/sonoff_watering

This timer contains a STA (WIFI client), an AP (not used), a MQTT client and a MQTT server (broker), provided with programmable logic (script): in practice a mini *mosquitto* + *node-red* on-board !

So in this project the operating logic is all internal: an external MQTT server 24/7 is not required (in fact I think a server 24/7 is too much to water a terrace!). The timer is totally autonomous, but at the same time always compatible with external standard MQTT clients and brokers.

A probe monitors the state of the soil (dry/soaked, with adjustable threshold) and conditions the watering.

Monitoring and configuration of the timer are performed with MQTT clients with dashboards, both on a PC and on a smartphone. Access is local via WIFI, but can be made Internet global using a free dynamic DNS service (example *noip*: https://www.noip.com/).

While using updated and quality components, this timer is very cheap: total cost less than € 15 (US$ 18)! All this is made possible by the happy union of two phenomenal products:

1) Sonoff basic, economic WIFI card with power supply, microprocessor EPS8266 and relay
2) esp_MQTT, a firmware for EPS8266 created by Martin-ger, which transforms the *Sonoff-basic* into a tank.

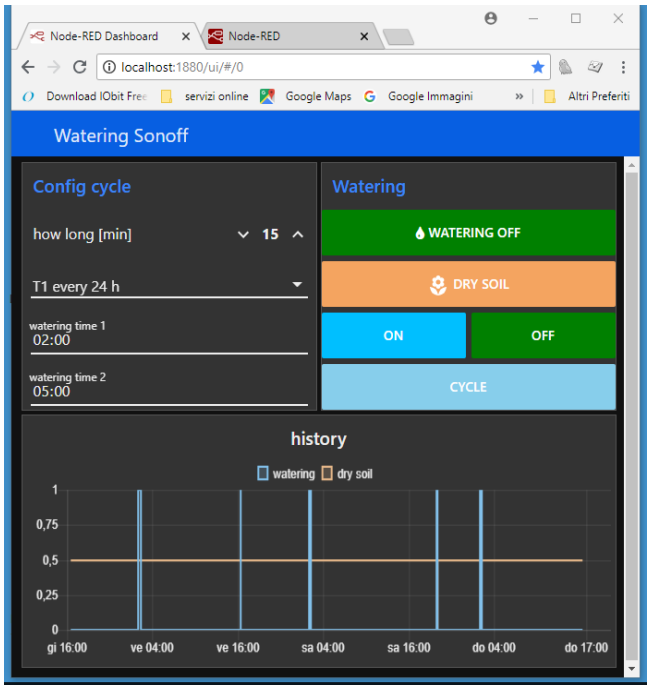
Performances:

- Two times: T1 and T2 (hh:mm)

- Operating mode:
  (0) manual (ON, OFF)
  (1) cycle at T1 every 48 hours,
  (2) cycle at T1 every 24 hours
  (3) cycles at T1 and T2 every 24 hours
  (4) start at T1, end at T2 every 24 hours
  - ✔ *cycle (1, 2, 3) start only if dry soil*
  - ✔ *interval (4): ON when dry soil OFF when soaked soil*

- Duration of the watering cycle:
  1 min...3 hours (1...180 min)

- Commands: ON, OFF (unconditional)
  CYCLE (immediate, conditioned)



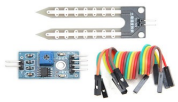
Features

- ✔ *Autonomous*: contains an MQTT server with the operating logic (*script*)..
- ✔ *Compatible*: can be used with standard MQTT client and MQTT server, has a complete set of commands and information via MQTT
- ✔ *Autostart*: preserves status and configuration in flash ROM: in case of reset or blackout restores the previous status autonomously.
- ✔ *NTP-client*: requires connection to the domestic WIFI, with access to the Internet, to have the correct time. Commands for summer/winter time.
- ✔ *Serial console*: for installation and debugging, via WIFI (telnet) or COM
- ✔ *OTA update*: the operating logic (*script*) can be updated via OTA (Over The Air), without moving or opening the timer.
- ✔ *MQTT client*: currently 2 options for monitoring and configuration:
  - Full client on PC (*node-red*)
  - Reduced client on Android (*IOT MQTT Dashboard* app)
- ✔ *Power supply*: (90-250) 220V AC, 220 V solenoid valve (***attention !!!***)
- ✔ *Stand-by consumption*: not measurable (<1 W)
- ✔ *Consumption ON*: 6 W (220 V)
- ✔ *Case*: waterproof and transparent, 3 LEDs: **power on, irrigation on, wet ground**

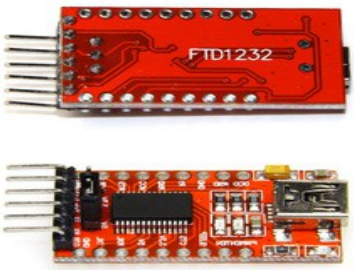
Notes:

1. *Depending on the type of watering system, short times or very long times are available (drip systems).*
2. *The positioning of the hygrometer probe affects the operations: I have positioned it in a point not watered but wet by rain. It is also possible to place it in a watered place, to create a sort of feedback (mode 4).*
3. *It is possible to control a small pump (max 500 W) instead of the solenoid valve.*

hardware

	Sonoff basic WiFi wireless switch https://www.itead.cc/smart-home/sonoff-wifi-wireless-switch.html	$ 4.85
	IP66 Waperproof 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 for c.s.	$ 0.10
	3 (+4) Female To Female Jumper Cable Dupont	$ 0.20
	FE-SP-HDR23-100/22 Surge protector	$ 4.16
	Resistor 220 KΩ	$ 0.10

instruments

	3D printer (I use a 3Drag) (optional, see text)
	Hot glue gun
	Welder with thin tip (I use PBLK 6 A1 Parkside)
	FTDI USB 3.3 V 5.5 V Serial Adapter Module (US$ 2.41)

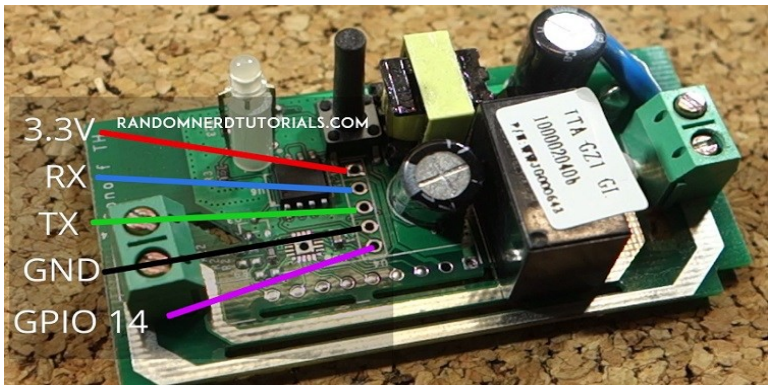
software

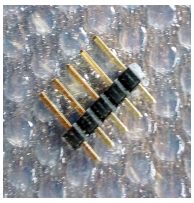
esp_MQTT	https://github.com/martin-ger/esp_mqtt
esptool	https://github.com/espressif/esptool
puTTY	https://putty.org/
e3DHW	https://www.thingiverse.com/thing:2860353
OpenSCAD	http://www.openscad.org/
Notepad++	https://notepad-plus-plus.org/
wpp_pampa	http://www.winpenpack.com/en/index.php
node-red	https://nodered.org/docs/getting-started/installation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

Construction

Preparation of the Sonoff-basic card

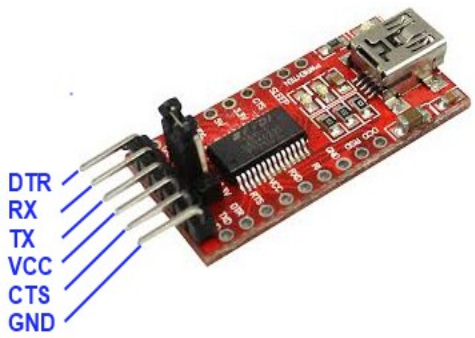
1. **Remove the Sonoff card** from the container
2. **Solder a 5-pin connector** in the Sonoff-Basic pads (to load the firmware and then to connect the humidity sensor)

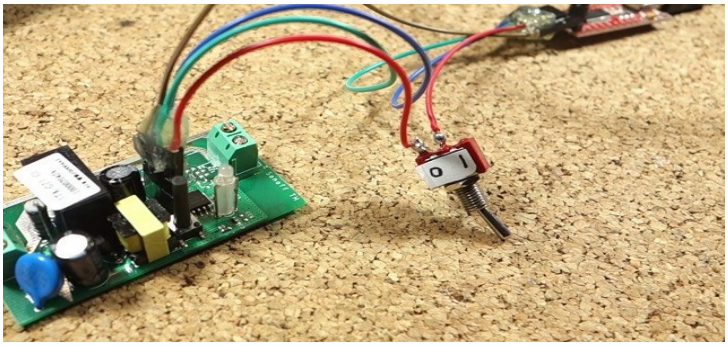




note: *previous versions of Sonoff-basic have only 4 pins (GPIO 14 is missing). In this case you can use RX (GPIO 1) instead of GPIO 14, modifying the script (see https://github.com/martin-ger/esp_mqtt/issues/28).*

3. **Prepare the 4-wire** FTDI ↔ Sonoff connection using female-female jumper cable:





FTDI: VCC → 3.3 (with switch) Sonoff
RX → TX
TX → RX
GND → GND

For full instructions see: http://randomnerdtutorials.com/how-to-flash-a-custom-firmware-to-sonoff/

4. Installation of *esp_MQTT*, using a Windows PC

esp_MQTT does not have to be compiled: *martin_ger*, besides publishing the sources, also publishes the 'bin' files ready to be loaded on Sonoff.

- Download and install esptool (https://github.com/espressif/esptool) needed to copy the firmware,
- Download from https://github.com/martin-ger/esp_mqtt/tree/master/firmware the 'bin' files with the compiled firmware, put them in the same dir of esptool.py
- ***Disconnect Sonoff from the main AC!***
- Connect FTDI via USB to the PC.
- Check the port number used (Control Panel, System, Device Management, Ports) In my case: COM6.
- Configure FTDI for 3.3 V and connect Sonoff.
- Press and hold down the Sonoff button before supplying power to Sonoff (use the switch on the FTDI-Sonoff connection): release the button on Sonoff 1 or 2 seconds after supplying power.
- Use the following command to copy the firmware into Sonoff-basic (COM6 can change). I use a small BAT file (`esptool/write-esp-Sonoff.bat`):

```
esptool.py --port COM6 write_flash -fs 1MB -fm dout 0x00000 0x00000.bin 0x10000
0x10000.bin
```

note: The original Itead firmware can be restored with some complications (see https://wiki.almeroth.com/doku.php?id=projects:sonoff). But this is usually not necessary: in fact, the "script.sonoff", a script from martin-ger, perfectly emulates the original Sonoff Basic operation (see - https://github.com/martin-ger/esp_mqtt/tree/master/scripts) .

note: Writing flash is not always successful at first sight. Try several times. The problem is often due to the little current supplied by FTDI to 3.3V. A simple solution (for me it worked) is to insert an electrolytic capacitor between VCC and GND (I used 100 µF). A more drastic solution is to use a separate 3.3V power supply (e.g. one step-down 5 → 3.3).

***DO NOT POWER AC Sonoff when FTDI is connected !!** An oversight (a screwdriver that falls on the circuit) and your PC (in the best case) is to be thrown away! But you could even risk your life !!*

note: For more information, for Linux etc. see martin-ger: "Building and Flashing".

5. **First run, serial configuration**

When the messages inform us of the correct writing of the flash memory, switch Sonoff off and on with the switch, without pressing the button and leaving FTDI connected.

The basic configuration can now be done with the serial console. You can use *puTTY* (https://putty.org/), with the configuration: 'serial', COM6, 115200

Pressing [ENTER] the '`CMD>`' prompt of esp_MQTT should appear, confirming the correct operation. Now we can give all console commands accepted by esp_MQTT (see https://github.com/martin-ger/esp_mqtt).

Note: ***alternative configuration with Telnet***

Using a laptop, connect via WIFI with Sonoff + esp_MQTT using the AP (enabled by default). Search for "MyAP". Now use PuTTY, previously installed on the laptop, to connect with Sonoff + esp_MQTT (Telnet, IP: 192.168.4.1, port: 7777).

For our purposes, we want ***Sonoff+esp_MQTT*** to connect as an STA to the home WIFI router:

```
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
CMD>set npt_timezone 2
```

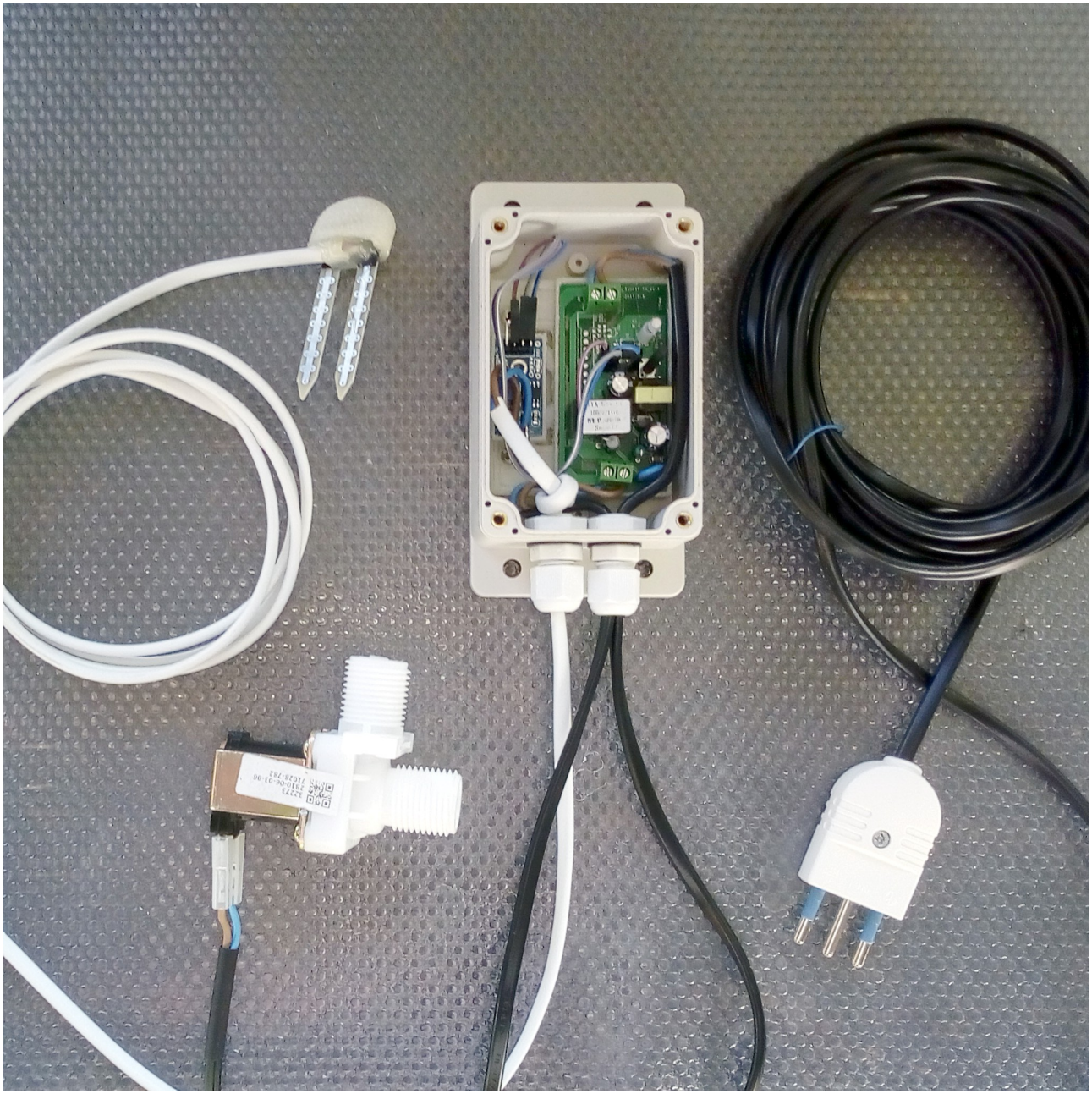
setting up the MQTT server (for simplicity I omit any security configuration):

```
CMD>set mqtt_host <hidro_name>
CMD>save
CMD>reset
```

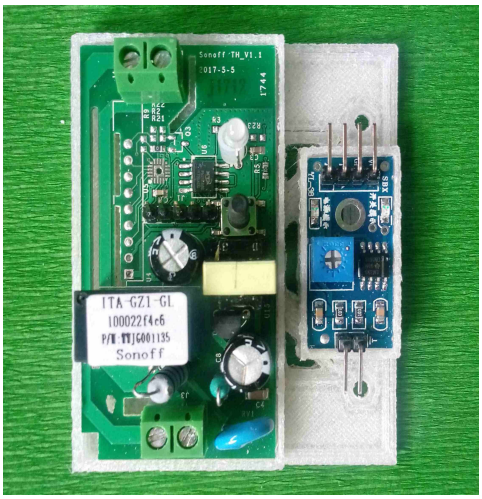
- After the reset ***Sonoff+esp_MQTT*** connects to the router indicated in the configuration.
- We check the connection and the ***Sonoff+esp_MQTT*** address on the router. On the router we must set a fixed address for ***Sonoff+esp_MQTT*** (e.g.: 192.168.0.53).
- From now the ***Sonoff+esp_MQTT*** *console can be remotely activated from every node in the network to the fixed address, with a telnet client* (e.g. *puTTY*) via tcp port 7777 to receive commands and tests.
- The MQTT server can be reached at the same fixed address, port 1883 (default).
- We can definitively disconnect FDTI, and check the correct operation of ***Sonoff+esp_MQTT*** connected to AC (**be careful !**, better to put Sonoff back in its case) with the remote console.

The ***Sonoff+esp_MQTT*** *card is ready: the script with the operating logic will be loaded OTA after assembly.*

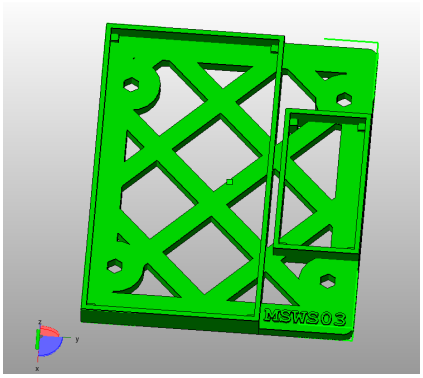
Wiring timer



This assembly is carried out following the **e3DHW** methodology, indeed this project has been used as an example.



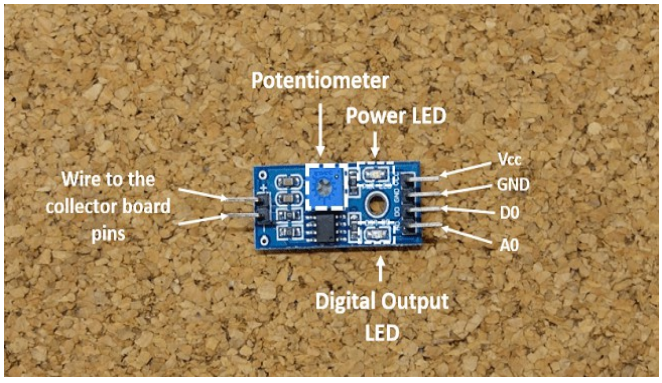
*The goal of **e3DHW** is to provide professional methodologies for the creation of DIY electronic equipment, based on 3D printing, including the simple integration of boards as in this case. See:* https://www.thingiverse.com/thing:2860353



The Sonoff card and the humidity detector card are placed on an ad hoc support obtained with 3D printing, then fixed to the watertight container with two self-tapping screws.

The file "`e3DHW_base_lib.1.1.scad`" contains just this support as an example. The `e3DHW_base_lib.stl` file is also available. (*The '`stl`' file is for printing, the '`scad`' file is the design, modifiable with OpenSCAD*)

Note: *without access to a 3D printer this solution can be simulated by cutting a flat plastic according to the model dimensions: size 67 x 56 mm, distances between the holes: 43 and 44 mm. The two boards can be attached to the plastic with hot glue. But every serious electronic hobbyist should have a 3D printer !!*

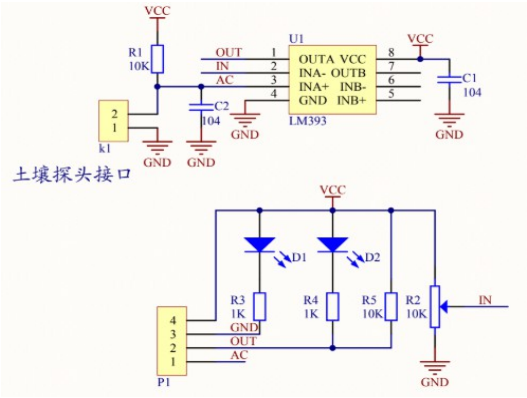


Positioned and if necessary fastened with hot glue the circuits, we pass to the wiring, consisting of 3 internal cables with female-female connectors, and three external bipolar cables:

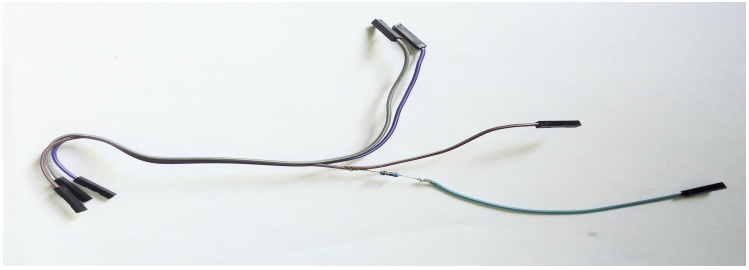
1 – from hygrometer-Vcc to Sonoff-3.3V

2 – from hygrometer-GND to Sonoff-GND

3 – from hygrometer-DO to Sonoff-GPIO 14



As we can see, the circuit uses a simple comparator, which generates a series of switchings close to the threshold. To improve the circuit we can transform it into a "comparator with hysteresis", more immune to noise, adding a 220 KΩ resistor between OUT (DO) and AC (AO). I made it on the cables.



Furthermore:

4 - The cable that carries the main AC (black) is connected to the two IN terminals of *Sonoff-basic*

Note*: The WIFI connection is quite critical: you can not choose where to place the timer without checking the reliability of the WIFI. I recommend, after connecting only the AC, to try to position the timer in various positions, checking the reliability of both the MQTT (node-red) and telnet (PuTTY) connections: for these tests, the timer script will be loaded (see below).*
Once you have found a suitable position, proceed with the wiring.

4 - the cable that connects the external probe (white in the photos) is soldered directly to the 2 pins. The probe is also welded, then inserted into a top protection and fixed with hot glue. (Protection is 3D printed: see files `3d/sensor.scad` and `3d/sensor.stl`).

6 - The noise suppressor and the cable connecting the solenoid valve (black) are connected in parallel to the two OUT terminals of Sonoff-basic.

As for the solenoid valve, in the photo you see my solution, removed the external cover, made from a plastic container.

Functioning logic

The "`ws0103.eub`" file is the script used for our Timer. The easiest way to load a script into ***Sonoff+esp_MQTT*** via OTA is to use a WEB server.

In development it is convenient to have the local server in the main PC: in my case I used the portable server "*wpp_pampa*" from http://www.winpenpack.com/en/index.php contained in a USB stick "*winPenPack*". Otherwise you can use any *WAMP / LAMP* server or even a remote server on the Internet, for example by downloading directly from Gitub.

To create the script I used *Notepad++* (https://notepad-plus-plus.org/), an excellent text editor for programmers. Note: *the script must be codified ANSI*.

Using the remote console (*puTTY*) you must give the command '`CMD> script <url_to_script>`'. For convenience the command to be used on the console is written in the second line of the file: modify it to adapt it to the used configuration, then use copy-paste.

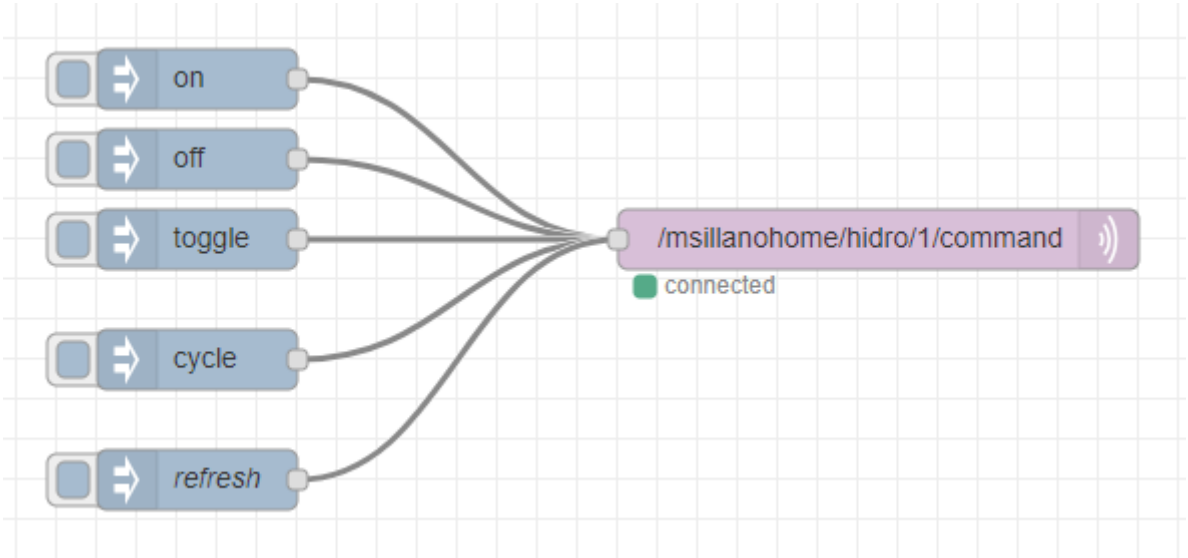
To stay within the memory limits (4,000 bytes), in the script I have:

1) eliminated the indentation
2) minimized comments
3) limited debug 'println'
4) minimized the initial 'config' commands
5) used names of 2-3 letters for the variables
6) reused the $tmp variable several times
7) eliminated the checks on the configuration values received

With these warnings the script is about 3880 bytes (for full commented script see `ws103-source.pdf`). Required customization:

```
2       % telnet: script http://192.168.178.23:85/www/sonoff/ws0103.eub    – set real URL
4       setvar @6 = "/msillanohome/hidro/1/"                               – modify topic and ID
```

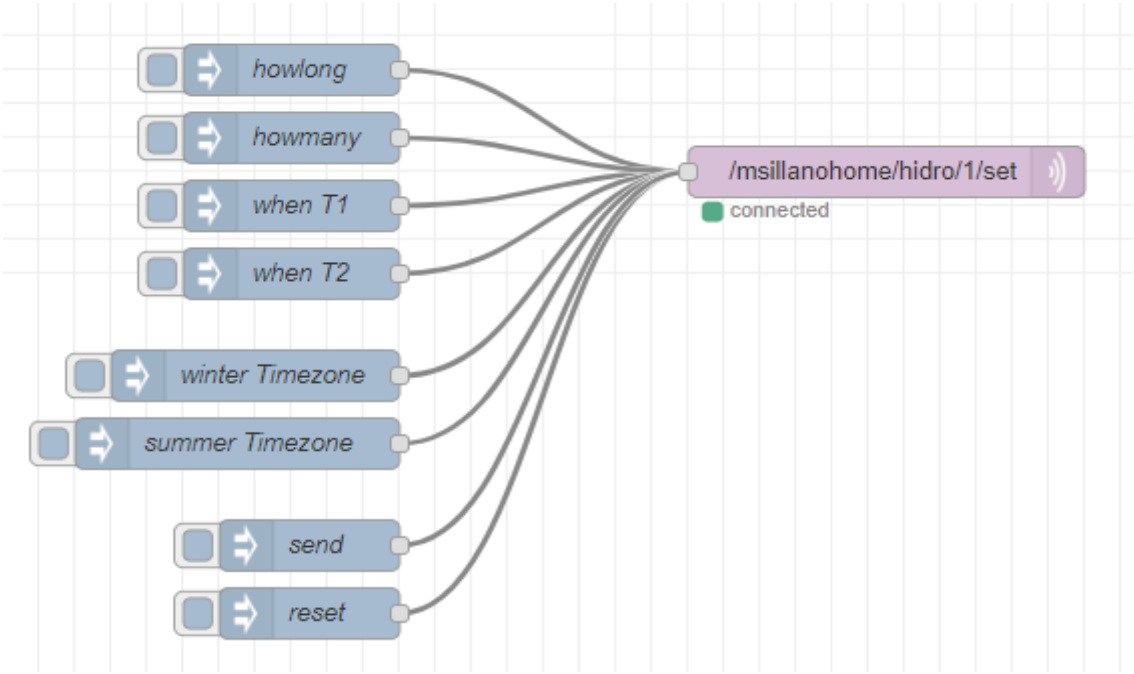
Sonoff+esp_MQTT: MQTT messages



Subscribe Command: `/msillanohome/hidro/<num>/command`

on : sets watering ON unconditional
off : sets watering OFF unconditional

toggle : reverses the state of watering, , unconditional
cycle : starts immediate watering cycle (conditional)
send : (refresh) re-send the last *satus/watering* and *status/moisure* retained



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

{“data” : “long” , “value” : <n>} : duration of the watering cycle, in minutes (1..180)
{“data” : “mode” , “value” : <n> } :(howmany) mode of operation (0..4)
{“data” : “when1” , “value” : “<HH:MM>” } : time T1
{“data” : “when2” , “value” : “<HH:MM>” } : time T2
{“data” : “timezone” , “value” : <h>} : set timezone, winter/summer time
{“data” : “send” , “value” : 0} : command to re-send info status/configuration
{“data” : “reset” , “value” : 0} : immediate reset command
note: *The values sent are wired into the node's properties so this flow only serves as test.*

Publish Status-config: `/msillanohome/hidro/<num>/status/config`
(after each modification, after 'config/send')

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

Publish Status-watering: `/msillanohome/hidro/<num>/status/watering`
(after every command, after 'command/send)

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

Publish Status-moisure: `/msillanohome/hidro/<num>/status/moisure`
(after each change, after 'command/send')

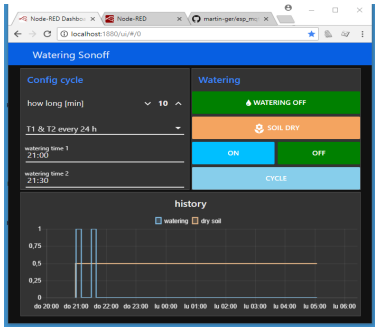
{"Time" : "<timestamp>", "moisure" : <0|1>} : 0 = soaked, 1 = dry

Notes:
`/msillanohome/hidro/` defines the type of device (in "`ws0103.eub`", it can be changed)
`<num>` device number (in "`ws0103.eub`", you can change it)

Sonoff+esp_MQTT: node-red client

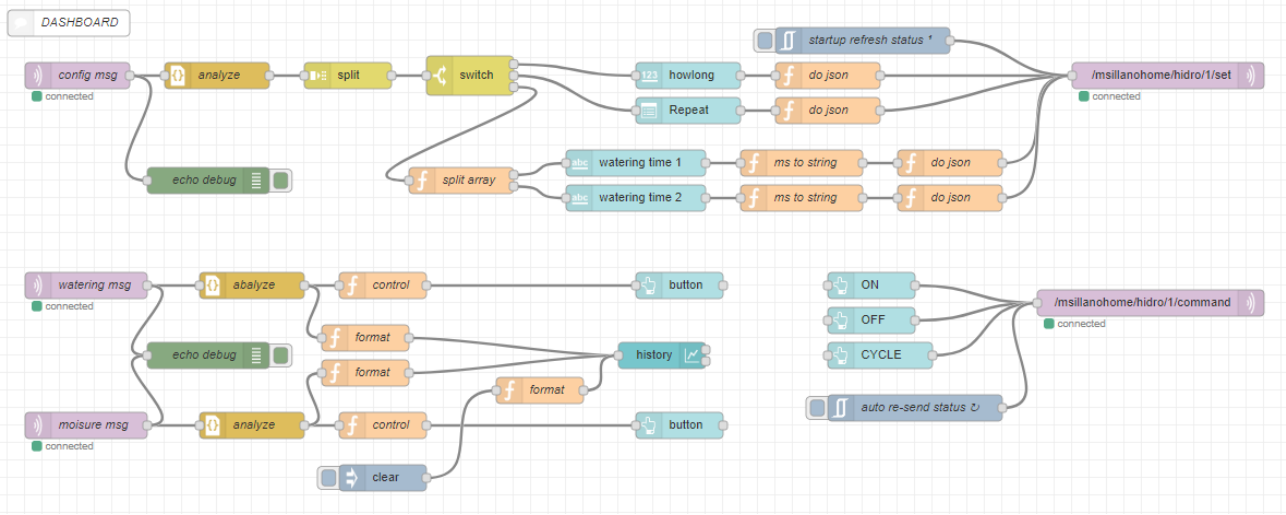
This client allows you to monitor the status of the timer and its complete configuration.

A historical graph is also available (if you leave the PC switched on) for the last 3 days.



Data storage on DB has not been implemented, but if desired, it is easy to add (unfortunately it requires a 24/7 server).

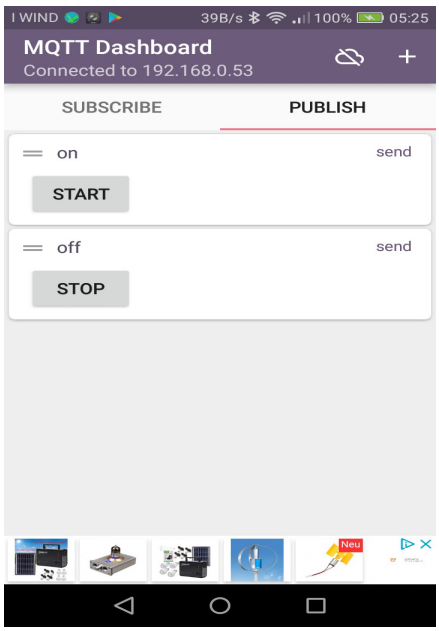
Full flow with *node-red* (for *node-red* installation, see https://nodered.org/docs/getting-started/installation).



Node-red can be installed on every PC (Win / Linux) and also on *Raspberry pi 3* if you want a low-cost dedicated computer (but failed to install in my smartphone Android 6.0):

- If not installed by default, also import the library '`node-red-dashboard`'.
- Copy the contents of the "`node-red/ws103-client-red.txt`" file to the clipboard and import it into a new *node-red* dashboard tab.
- Configure access to ***Sonoff+esp_MQTT*** as broker.

Sonoff+esp_MQTT: Android client



Because the timer don't have immediate commands, this simple client allows you to control irrigation from anywhere in the house using an Android smartphone.

For this client I chose "*IoT MQTT Dashboard*": see https://play.google.com/store/apps/details?id=com.thn.iotmqttdashboard free, but there are many other alternatives. Search in Google Play.

Just configure ***Sonoff+esp_MQTT*** as broker, and add (publish) at least the two ON and OFF commands.
Example:

Topic:	/msillanohome/hidro/1/command
Value to publish:	on
Button text:	START

Note: *the response time is slow, so you can lose data update messages (subscribe). But in this application it is not critical.*

Note: *I believe you can find app with MQTT + dashboard function also for O.S. different from Android.*