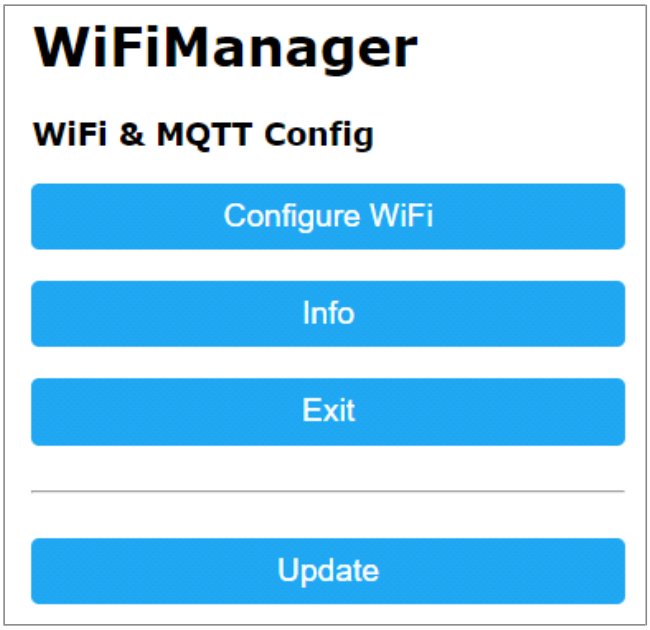
03_AP 192.168.4.1

Freitag, 19. April 2024 10:48

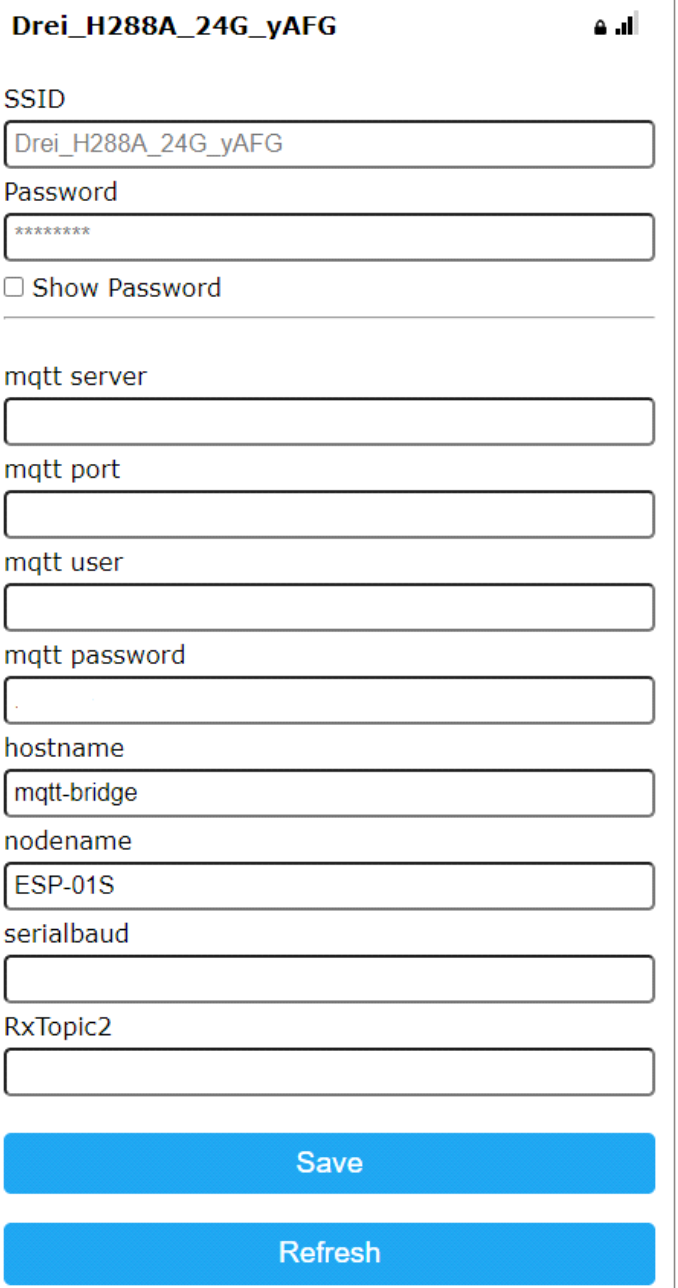
- WLAN am PC aktivieren
- mit "WiFi & MQTT config" verbinden
- 192.168.4.1 im Browser auffrufen.



- Unter **Configure WiFi** können nun die WLAN- & MQTT-Parameter eingegeben bzw. geändert werden.

- Ebenso können folgende Parameter geändert werden, die auch beim Start nach MQTT übertragen werden:
- **nodename** (default das eingestellte Board)
- **serialbaud** (Serial Baudrate)
- **RxTopic2**

Mit **Save** speichern.



192.168.4.1/wifisave	speichert die eingegebenen Werte
192.168.4.1/exit	ist erforderlich, damit der AP geschlossen wird
.../cmd/reboot = 1	dann erst kann reboot erfolgen

04_Befehle via Serial

Freitag, 19. April 2024 10:47

FW (Grund)konfiguration:

• Alle **Befehle** beginnen mit einem **'@'** und können weitere Parameter, getrennt durch Leerzeichen, enthalten und sind mit einem **\n** abzuschliessen.
• Weiteres gibt es **MQTT-Befehle** s. **05_ MQTT-Komunikation**
• Falls kein Befehl erkannt wird, erfolgt eine Übertragung nach MQTT als **ASCII**, wenn **mit \n** abgeschlossen, bzw. als **HEX-String** wenn **kein \n**.

Befehl	Rückmeldung
☑ @**connect** mySSID ☑ @**connect** mySSID myPwd	(on start): [conecting to WiFi] (on success): [wifi connected]
☑ @**mqttuserpass** mymqttUser [mymqttPwd]	[MQTT user & password set]
☑ @**mqttserver** mymqttserver [port]	(on start): [connecting to MQTT server] (default Port: 1883) (on success): [mqtt connected]
☑ @**publish** mytopic mypayload ☑ @**publishretained** mytopic mypayload	[published topic payload] | [wrong publish command] | [MQTT not connected]
☑ @**subscribe** mytopic	[subscription added: topic] | [MQTT not connected] | [subscription exists]
☑ @**unsubscribe** mytopic	[subscription removed: …] | [MQTT not connected]
☑ @**hostname** myhostname	[hostname myhostname]
☑ @**echo** [**on**|**off**]	[echo] check ESP8266 alive, switch DEBUG_printf nach Verbindungsaufbau wird echo off und kann via Serial-Command wieder eingeschaltet werden
☑ @**wifi**	[WiFi connected ssid ip] | [WiFi not connected]
☑ @**mqtt**	[MQTT (not) connected server port user pwd] [all Subscriptions]
☑ **@Xflag** [xxxxxxxx]	8 Zeichen langer String, abfragen/setzen (s. unten)
☑ **@@config**	start AP 192.168.4.1 for web-config
☑ **@@OTAupdate URL**	Self-OTA-Update von http:// und https:// URL möglich
☑ ~~**@help | @?**~~	~~Ausgabe aller Befehle~~
☑ **payload**	MQTT publish payload to default topic • ASCII wenn mit \n • HEX wenn ohne \n
☑ **@@@@@@@@@@HEX\n** ☑ **@@@@@@@@@@ASCII\n**	schaltet Übertragungsmodus auf HEX bzw. ASCII

Xflag	grundsätzlich jedes Zeichen an jeder Stelle möglich
@Xflag	abfragen ohne String
@Xflag xxxxxxxx	setzen mit String mit folgender Bedeutung
0xxxxxxx	echo off
1xxxxxxx	echo on

05_MQTT-Kommunikation

Freitag, 19. April 2024 10:47

• Ein einfaches Verfahren, Änderungen über MQTT-Befehle zu senden, womit auch die Default-Werte geändert werden können und im EEPROM gespeichert werden.
• Voraussetzung nach RST/Boot: WLAN- & MQTT-Verbindung erfolgreich. Sonst **@@config** via Serial.
• MAC-Adresse als String die letzten 2 Bytes als ID der Node (ESP-01S) >> "xxxx" (=**<mac>**)
Damit ist die FW für alle verschiedenen Nodes gleich und nur die **<mac>** wird individuell beim RST gebildet und dient als ID.
• Node sendet auf folgende Topics nach MQTT-connect folgende Grunddaten:

publish bridge/<mac> = <name>	automatische Anmeldung (Name/Funktion/...) default **<name> = ESP-01S** (oder gesetzen)
publish bridge/<mac>/time =	NTP aktuelles Datum & Zeit (dd.mm.yyyy hh:mm:ss)
publish bridge/<mac>/rssi =	zu jedem **time** wird auch der **rssi** gesendet
publish bridge/<mac>/version =	FW-Version (v.r.s)
publish bridge/<mac>/baud =	default oder gesetzte Serial-BaudRate
publish bridge/<mac>/RxTopic2 =	RxTopic2 (= Topic, mit der die Gegenstelle sendet) default leer, muss ggf. gesetzt werden
publish bridge/<mac>/RxEcho =	alle empfangenen Daten werden hier wiedergegeben
• Node hört auf diese Topics:	
subscribe bridge/<mac>/Rx	fixes **RxTopic1** für Datenempfang
subscribe <RxTopic2>	**RxTopic2** (falls gesetzt) für Datenempfang
subscribe bridge/cmd	für globalen Befehlsempfang
subscribe bridge/<mac>/cmd/#	Topic für "persönlichen" Befehlsempfang
• Node sendet auf fixes Topic:	
publish bridge/<mac>/Tx = payload	default Topic für Datensendung (grundsätzlich fix) (ASCII via Serial wenn mit \n) (HexString via Serial wenn kein \n)

• MQTT-Befehle für eine erweiterte Kommunikation & Vermittlungsmonitor, wobei globale Befehle für alle Nodes und individuelle Befehle für Nodes mit <mac> unterschieden werden:

bridge/<mac>/cmd/**name = ...**	setzt Node mit <mac> den Namen/Funktion > EEPROM
bridge/<mac>/cmd/**baud = ...**	setzt Node mit <mac> Serial-Baudrate > EEPROM
bridge/<mac>/cmd/**RxTopic2 = ...**	setzt Node mit <mac> das RxTopic2 > EEPROM = konfigurieren der Gegenstelle-Tx
bridge/<mac>/cmd/**mode = HEX | ASCII**	setzt Node in HEX bzw. ASCII Übertragungsmodus
bridge/<mac>/cmd/**reboot = 1**	RST/reboot
bridge/<mac>/cmd/**configAP = 1**	start AP 192.168.4.1
bridge/<mac>/cmd/**OTAupdate = URL**	http- | https-URL (s. spezielle Beispiele)
bridge/<mac>/cmd/**Xflag = xxxxxxxx**	setzen von Xflag (abfragen mit globalen cmd=?)
bridge/**cmd = ?**	[global] fordert alle Nodes auf, Grunddaten zu senden