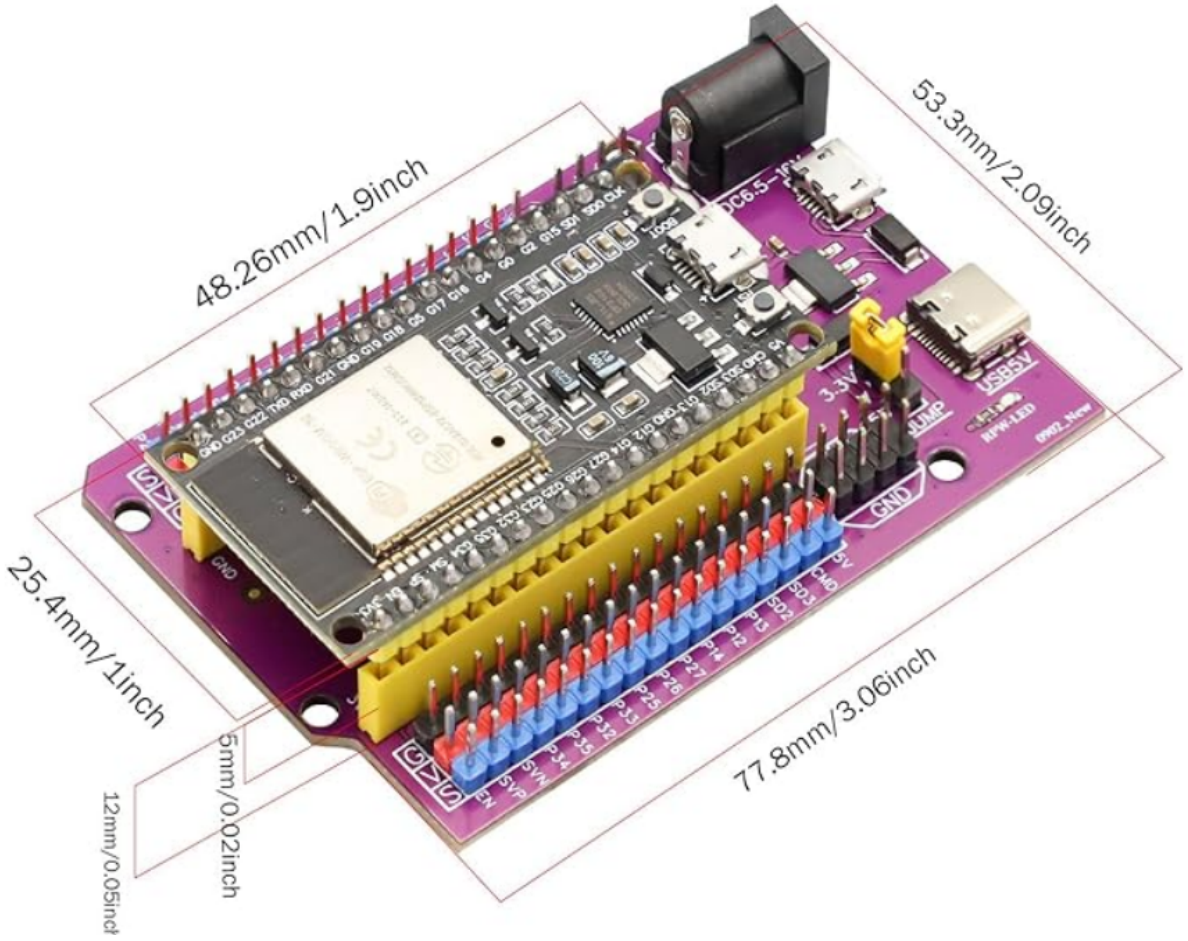
Geräte programmieren

Warum Tasmota?
Tasmota liefert einen Baukasten zur Lösung verschiedenster Aufgaben



Über diesen Beitrag: Wer Tasmota genauer kennen lernen will, sollte unbedingt ein paar Anwendungsbeispiele selbst ausprobieren. *Aber wer tippt schon gerne Programme ab?* Manche technische Einzelheiten werden beim ersten Durchlesen eventuell übersprungen oder sind im Detail nur über Links zugänglich. *Aber wer tippt schon gerne Links ab?* Aus diesen Gründen gibt es den Beitrag in zwei Formen:

1. Die gekürzten Fassung enthält keine Programmbeispiele und (fast) keine Links und ist in der Zeitschrift PCNEWS abgedruckt. Die Kurzfassung soll das Interesse an dem Thema wecken und zum Lesen der Langversion einladen. Ein ☝ im Text weist darauf hin, dass in der Langversion an der Stelle mehr Text, ein Programmcode oder eine detaillierte Anleitung zu finden ist.
2. Die Langversion kann über die Webseiten des ClubComputer abgerufen werden und lädt dazu ein, den weiterführenden Links zu folgen und die Beispiele auch selbst auszuprobieren. Die Links leiten direkt zu jenen Informationen, die nicht mit Werbung zugemüllt sind, ferner anwendbar und aktuell sind, außerdem Begriffe genauer erklären und zu weiterer Beschäftigung mit dem Thema einladen.
   Die Langversion ist unter https://t.ly/slity zu finden und der QR-Code am Ende dieses Beitrags.

   Die Screenshots und Programmbeispiele betreffen die Tasmota-Version 13.4.0.

Was erwartet Sie in diesem Beitrag?

Sie werden...

- Tuya und Sonoff kennenlernen
- vorprogrammierte Bauteile kennenlernen
- Programmiermethoden aufzählen können
- einen ESP32 programmieren (flashen) und konfigurieren
- Steuerbefehle an ein Tasmota-Gerät senden
- externe Sensoren und Aktoren anschließen
- Tasmota-Geräte ohne Broker betreiben
- die Skriptsprache *Berry* kennenlernen
- Matter kennenlernen
- einen wichtigen Unterschied zwischen ZigBee und Tasmota kennenlernen

Tuya und Sonoff

Die Heizung von unterwegs steuern? Das Licht im Garten einschalten? Wenn es regnet, den Rasensprenger abgedreht lassen? Alle das sind Aufgaben, die ein Smart Home, ein automatisiertes Haus, erledigen kann. Die Komponenten dafür sind nicht mehr allzu teuer und laden zum Eigenbau ein. Einige Firmen bieten immer noch Produkte zu stark überhöhten Preisen an. Klondike 2024! Wenn die Produkte außerdem nur zu eigenen Geräte kompatibel sind, ist es besser, die Finger davon zu lassen.

Tuya

Der Markenname Tuya wird dabei oft genannt. Tuya Smart ist nach eigener Definition *ein führender Technologiekonzern, der sich darauf konzentriert, unser Leben eleganter zu machen*. Tuya bietet eine Cloud-Lösung für die Vernetzung von Geräten über das Internet-of-Things (IoT) an.

Im Original liest sich das so: *Tuya is a leading technology company focused on making our lives smarter. Tuya does this through offering a cloud platform that connects a range of devices via the IoT.*

Ja, das Angebot an Tuya-Geräten ist sehr groß und enthält (nach Tuya-Angaben) 2700 Produktgruppen. Geräte und Komponenten werden über Apps gesteuert, die für unterschiedliche Plattformen angeboten werden: Smart Phones, Tabletts, Desktop-Geräte, Uhren und andere. Die Apps heißen *Tuya Smart App* oder *Smart Life App*: vermutlich das weltweit größte Netzwerk für Smart Homes. Die Daten werden natürlich über das Internet übertragen. Die App ist einfach zu installieren und bequem zu bedienen - warum also nicht?

Die gesamte Steuerung läuft über Tuya–Rechenzentren. Eines gibt's in Deutschland, andere stehen in China.

- Beim Anmelden der App sind die Zugangsdaten des Heimnetzwerkes (Name des Netzwerks und *Passwort (!)*) einzutragen — klingt nicht nach einem hohen Sicherheitsstandard. Wollen wir wirklich unsere Zugangsdaten einem unbekannten Server anvertrauen?
- Natürlich laufen auch alle Schaltvorgänge über den Server. Man muss ja nicht unbedingt paranoid sein, aber daraus ist schon eine Menge über unser Leben abzuleiten. Wollen wir das?

- Was, wenn der Server ausfällt oder abgeschaltet wird - aus technischen oder politischen Gründen? Dann haben wir eine Menge Edelschrott im Haus. Aber wollen wir das?

Sonoff

Sonoff ist auch ein Anbieter von Smart Home Geräten. Deren App heißt eWeLink. Wie bei Tuya werden Geräte über einen zentralen Server gesteuert.

Raus aus der Cloud!

Ist die mangelnde Datensicherheit, die Möglichkeit, dass das Benutzerverhalten ausspioniert wird, und die Abhängigkeit von zentralen Diensten tatsächlichen allen Käufern klar? Vermutlich nicht. Oder es ist ihnen egal. Jedenfalls hat Theo (Matheus) Arends eine alternative Firmware entwickelt, die anstelle der Originalfirmware geladen werden kann und darüber hinaus noch etliche neue und zusätzliche Steuerungsmöglichkeiten bietet. Zuerst entwickelt für Geräte der Firma Sonoff wurde die Software auch zum Umprogrammieren von Geräten anderer Hersteller erweitert.
Voraussetzung ist allerdings, dass ein Prozessor des Herstellers Espressif (ESP8266, ESP8285, ESP32, ESP32-S, ESP32-C3 u.a.) verwendet wird.

> **Note**
>
> Neuere Tuya-Geräte verwenden andere Prozessoren oder verhindern das einfache Einspielen von alternativer Firmware.

Die Firmware von Theo Arends verwendet das MQTT-Protokoll und wird direkt über das WLAN, also "Over-the-air" eingespielt. Daraus leitet sich der Name ab: Theo-Arends-Sonoff-MQTT-OTA, Tasmota. Die Firmware kann auch direkt über die serielle Schnittstelle des Prozessors eingespielt werden. Aber trotz aller Änderungen ist der Name *Tasmota* geblieben.

Einige ausgewählte Geräte werden auf der deutschsprachigen Tasmota-Webseite im Detail beschrieben. Sehr nützlich: eine (immer wieder aktualisierte) Liste aller programmierbaren Geräte für eigene Smart-Home-Projekte.

Mit Tasmota vorprogrammierte Bauteile

Der Zwischenstecker A1T von NOUS ist bereits mit der Tasmota-Software ausgerüstet. Details sind in meinem Artikel in den PC NEWS 177 vom Mai 2023 zu finden. Bitte dazu auch die Korrektur am Ende dieses Beitrags (*Nachtrag: Zigbee und Tasmota*) lesen!
Auch andere Hersteller bieten Komponenten an, die bereits mit Tasmota vorprogrammiert sind. In den meisten Fällen sind es ebenfalls Zwischenstecker. Ferner habe ich einen Adapter zum Auslesen der neuen Stromzähler entdeckt. Ich empfehle eine kurze Suche bei Amazon oder beim Händler des Vertrauens.

Programmiermethoden - ein Überblick

Ursprünglich war für Tasmota die Neuprogramierung nur *Over-the-air*, also über das WLAN, vorgesehen. Aber es geht auch anders. Jedenfalls können nur Smart Home Geräte mit Espressif Prozessoren (zum Beispiel ESP8266, ESP32) *tasmotisiert* werden.

Flashen

Was heißt eigentlich *flashen*? Das Wort hat es in den Duden geschafft und wird dort mit *den Flashspeicher überschreiben* erklärt. Na ja - ein Wort durch ein anderes mit demselben Wortstamm zu erklären, ist etwas seltsam. Beim Programmieren nennen wir das *rekursiv*. Die in universitären Kreisen oft verachtete Wikipedia erklärt das Wort (unter anderem) *mit dem Aufspielen einer neuen Firmware auf ein technisches Gerät*.

Over-the-air

Das zu programmierende Gerät muss ein Update über das WLAN zulassen. Dann kann ein Computer, zum Beispiel ein Raspberry, den (oft in China stehenden) Server simulieren. Anstelle eines Updates wird jedoch eine komplett neue Software, die Tasmota-Software, eingespielt. In den meisten Fällen braucht dazu das Gerät nicht geöffnet zu werden.

Über die serielle Schnittstelle

> **Caution**
>
> Beim Programmieren über die serielle Schnittstelle darf das Gerät unter keinen Umständen mit der Netzspannung (230 V) verbunden sein. **LEBENSGEFAHR!**
>
> Beim Programmieren wird das Gerät ausschließlich über die serielle Schnittstelle mit Energie versorgt.

Die Prozessoren haben (mindestens) je eine Datenleitung zum Senden und zum Empfangen von seriellen Signalen und können darüber programmiert werden. Benötigt wird ein PC oder Laptop mit USB-Anschluss, das Programm zum Flashen und ein Adapter, der die USB-Signale für die Pegel der Sende- und Empfangsleitungen (3,3 V) aufbereitet. Zusammen mit der Spannungsversorgung und dem Masseanschluss sind das vier Anschlüsse. Das Prinzip wird auf der Tasmota-Webseite erklärt, das beschriebene Gerät ist aber nicht mehr verfügbar. Trotzdem ist die Erklärung hilfreich, da sie die Vorgänge sehr detailliert schildert.

Manche Komponenten enthalten dafür schon vorbereitete Steckanschlüsse. Beispiele dafür sind Sonoff- oder einige Shelly-Komponenten. Bei anderen Geräten sind zwar die Anschlusspunkte bekannt und auch auf diversen Webseiten dokumentiert, verlangen aber Handarbeit mit einem Lötkolben: vier Drähte oder noch besser Pfostenstecker müssen angelötet und mit dem USB-Adapter verbunden werden. Eine detaillierte Beschreibung ist auch dazu auf der Tasmota-Seite auf Englisch zu finden.

Der ESP-01 gehört ebenfalls zu dieser Familie. Der Winzling hat acht Anschlüsse, von denen zwei bis vier für Steuerungen verwendet werden können. Genau genommen heißt er *ESP8266 ESP-01* oder *ESP8266 ESP-01S*. Für diese Prozessoren gibt es kleine Platinen mit USB-Anschluss, auf die der ESP-01 gesteckt wird. Im Internet werden komplizierte Schaltungen mit Widerständen und Schaltern beschrieben — unnötig, diese Platinen erledigen alles selbst. Espressif bietet die geeignete Software zum Flashen an. Von Tasmota gibt's den sogenannten *Tasmotizer*.

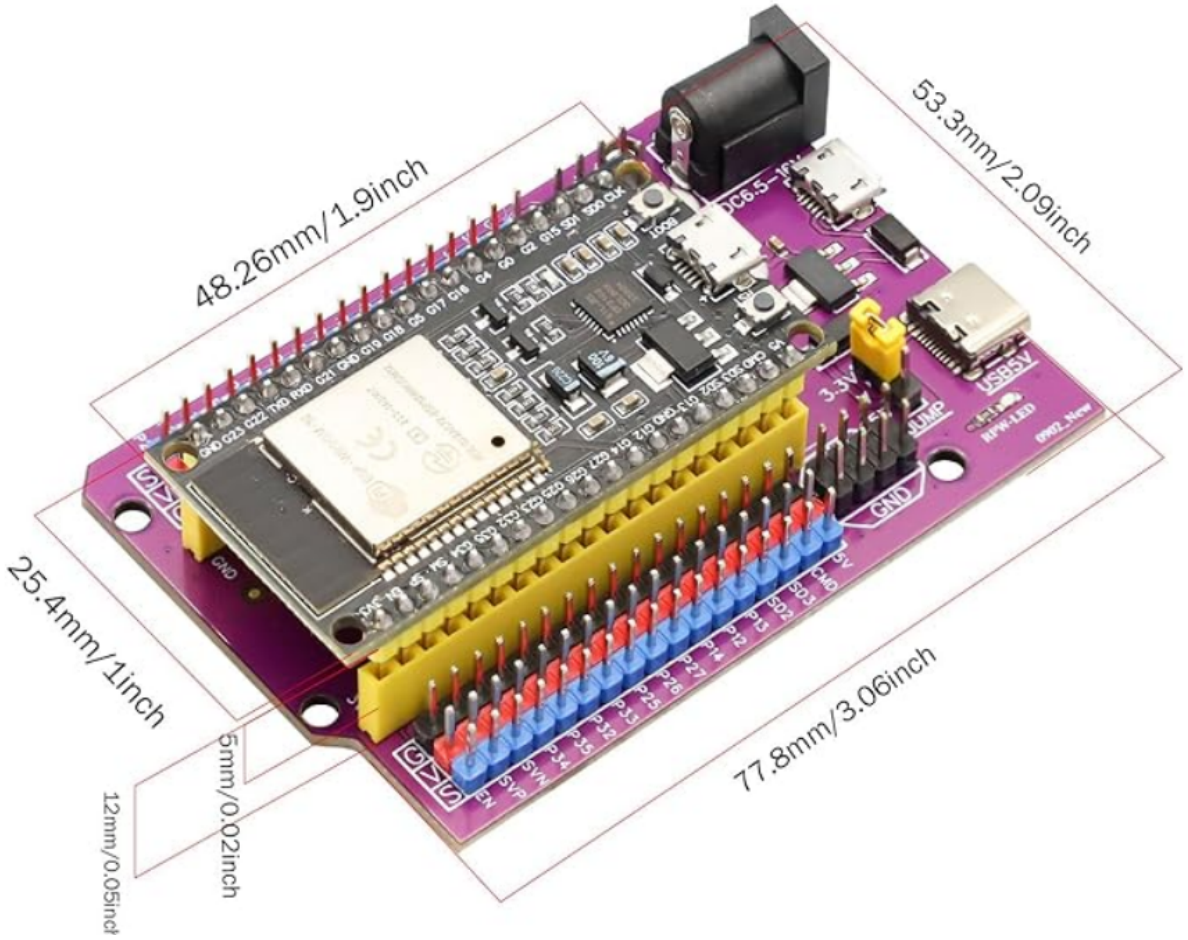
ESP32– oder ESP8266–Experimentierboard

Ein Experimentierboard mit einem ESP32 oder ESP8266 flashen geht besonders einfach, Details dazu im nächsten Abschnitt.

Mehr über den ESP32

Von allen genannten Prozessoren ist der ESP32 der interessanteste: Er ist sehr kostengünstig (schon unter 5 € erhältlich) und außerordentlich vielseitig. Somit ein idealer Kandidat für eine genauere Betrachtung!

Das Titelbild zeigt einen ESP32 (das ist der quadratische Bauteil mit metallischer Oberfläche links im Bild), der auf einer Platine (*Entwicklungsboard*) mit USB-Anschluss aufgelötet ist. (*Brett* ist keine schöne Übersetzung für *board*.) Diese Platine steckt wiederum in einem Erweiterungsboard mit vielen Verbindungspfosten, die die Versuchsaufbauten erleichtern.



Flashen

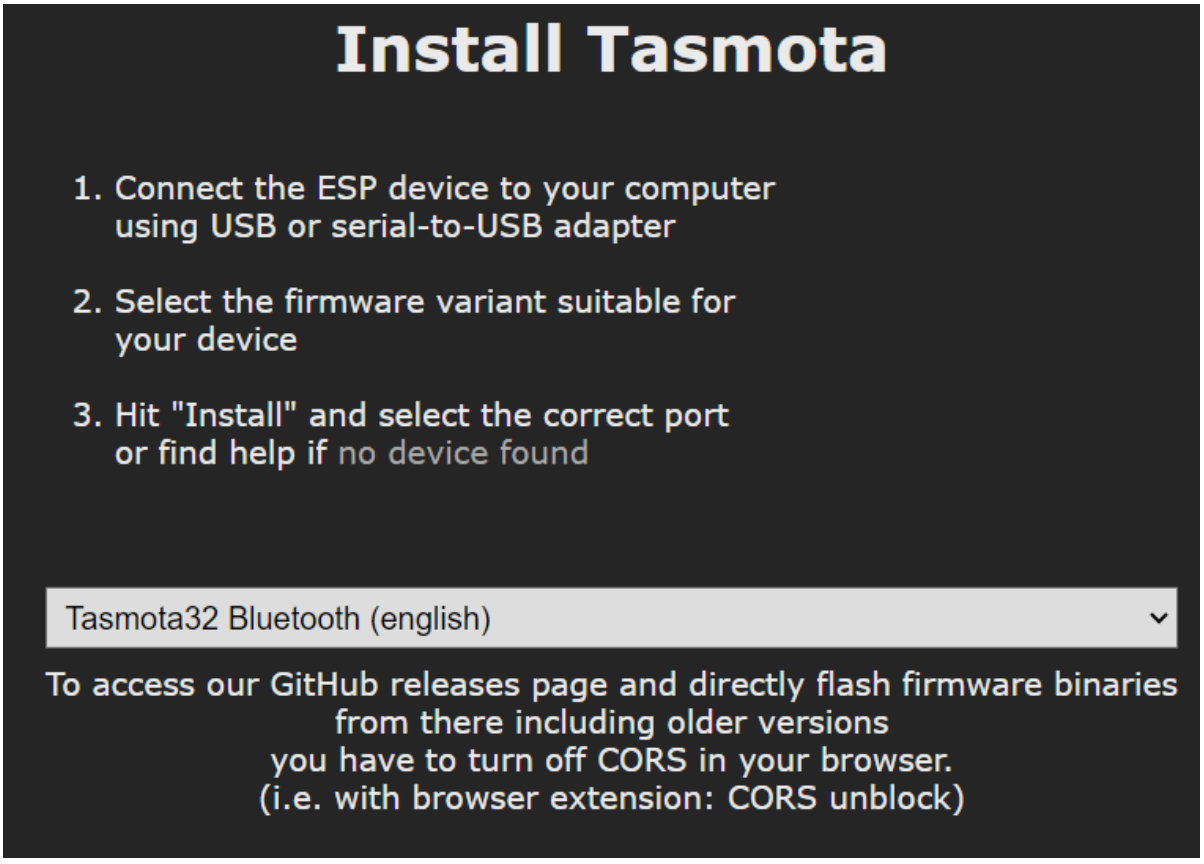
Espressif-Mikroprozessoren mit Entwicklungsbord werden über ein Mikro-USB-Kabel oder USB-C-Kabel mit einem PC verbunden und laut Anleitung auf der Tasmota-Webseite oder über den Tasmota Web Installer geflasht. Der Vorgang setzt keine Vorkenntnisse voraus und kann in 5 Minuten abgeschlossen werden

> **Important**
>
> Das USB-Kabel darf daher kein reines Ladekabel sein, also kein Kabel, das nur die Versorgungsspannung übertragt. Vielmehr müssen auch die Datenleitungen durchverbunden sein. Es sollte höchstens 1 m lang sein, um Spannungsverluste zu verhindern.

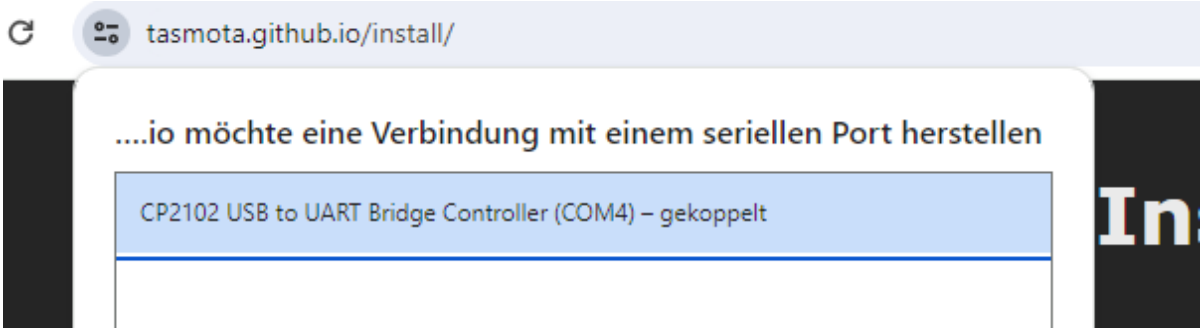
> **Tip**

> Die Stromversorgung muss stabil und leistungsstark genug sein. Nicht alle USB-Verteiler können den notwendigen Strom für die Programmierung liefern. Daher das Kabel an einen USB-Port direkt am PC anstecken!

Nun folgt der Aufruf des Tasmota Web Installers über *Google Chrome* oder *Microsoft Edge*, `Tasmota32 Bluetooth (english)` auswählen:



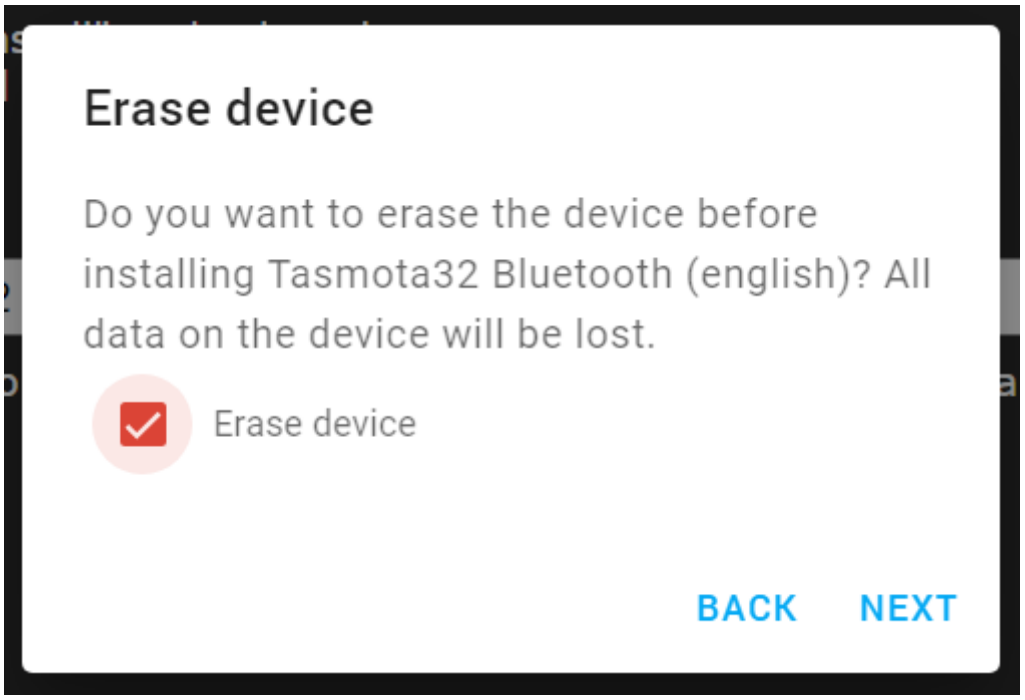
Als nächstes `CONNECT` anklicken.



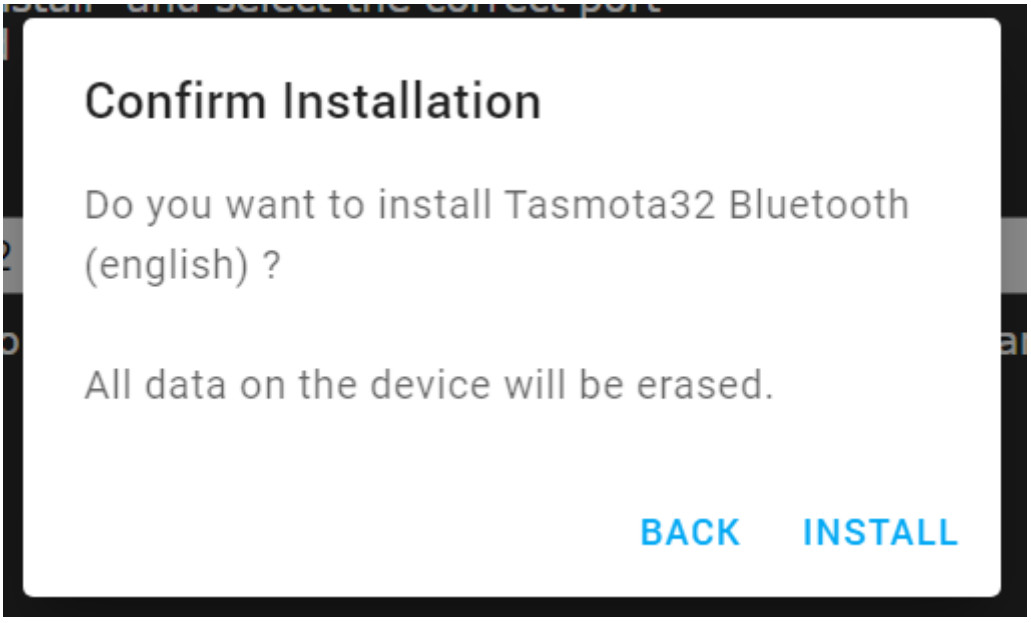
Mit `Verbinden` bestätigen.

> **Tip**
>
> Die seriellen USB-Signale müssen für den ESP32 aufbereitet werden. Das macht der kleine quadratische Prozessor mit der Aufschrift CP2102 oder CH430. Falls die dazu gehörenden Treiber am PC nicht installiert sind, wird die Schnittstelle nicht erkannt. Dann muss der passenden Treiber nachinstalliert werden.

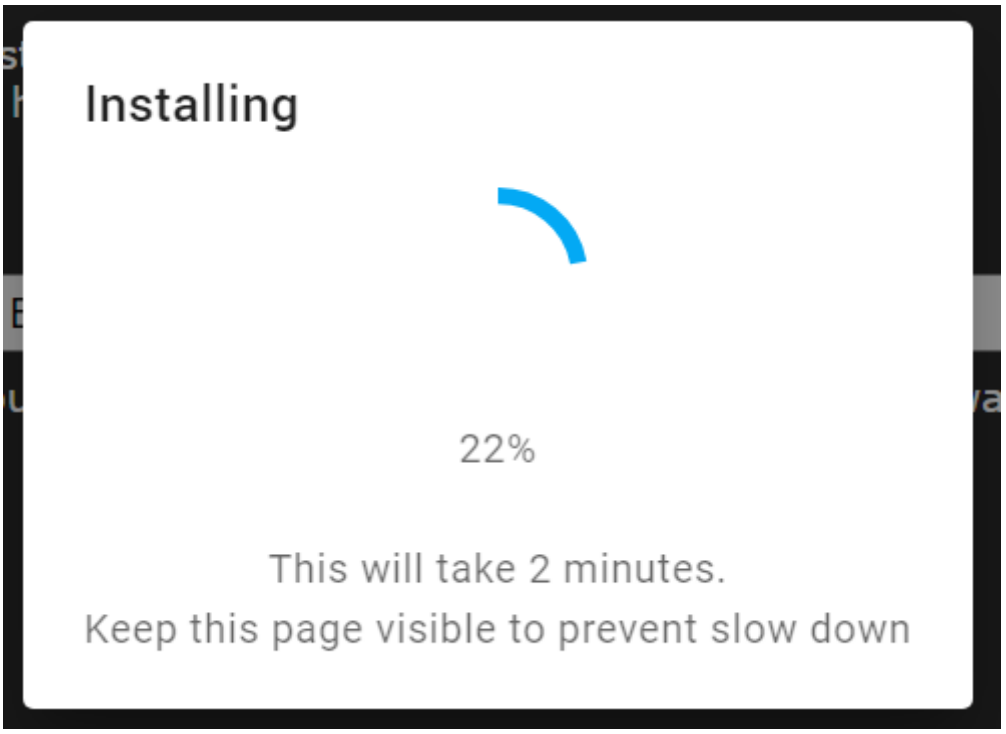
Dann `INSTALL TASMOTA32 BLUETOOTH (ENGLISH)` wählen



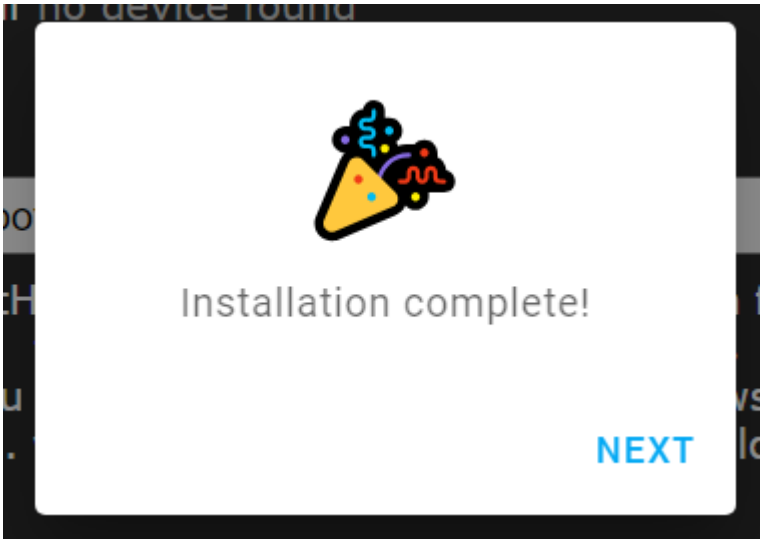
`Erase device` erlauben und weiter mit `NEXT`



Noch einmal mit `INSTALL` bestätigen und die Installation startet.

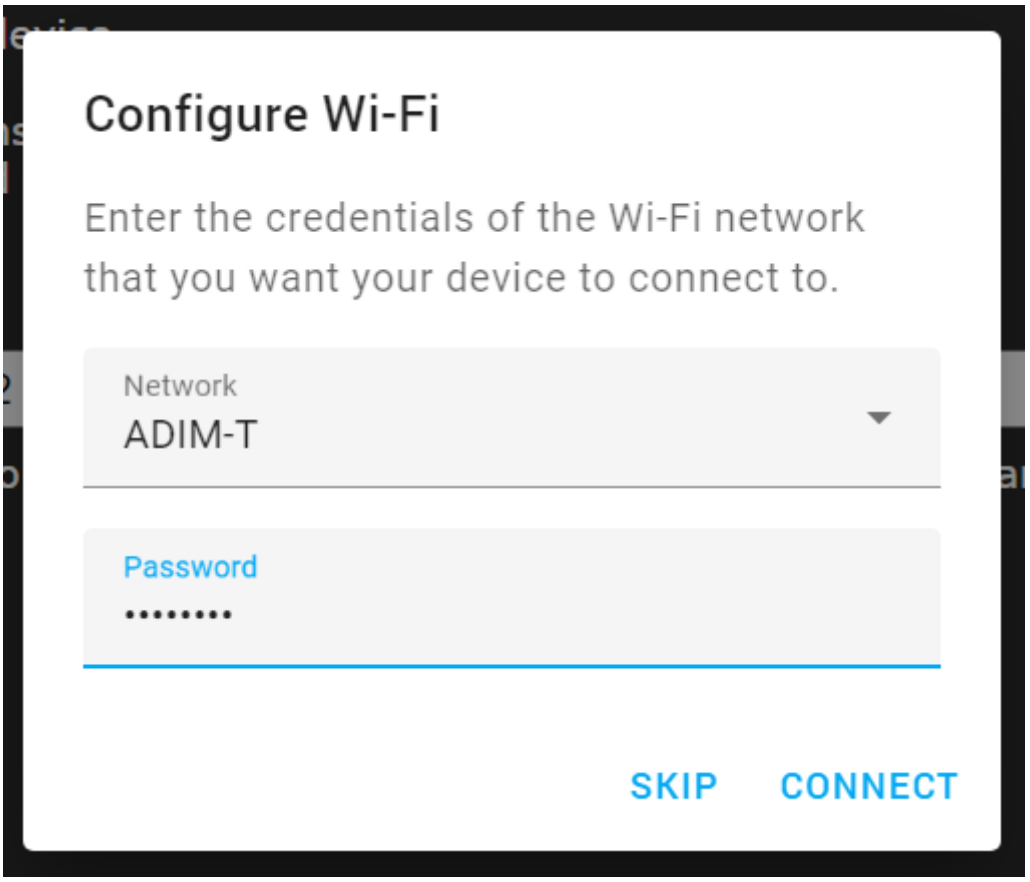


Fertig!



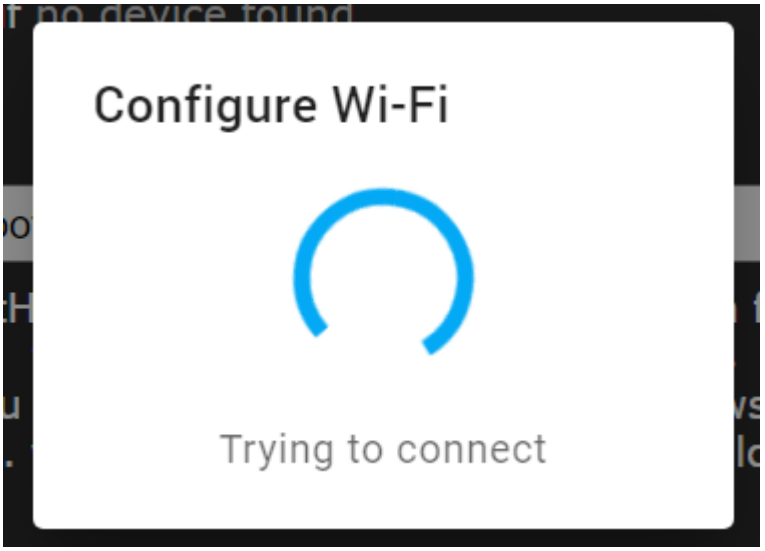
Weiter mit `NEXT`.

Nun muss der ESP32 mit unserem Wi-Fi Netz verbunden werden. Nein, der ESP32 telefoniert nicht nach Hause, alles spielt sich im lokalen Netz ab.

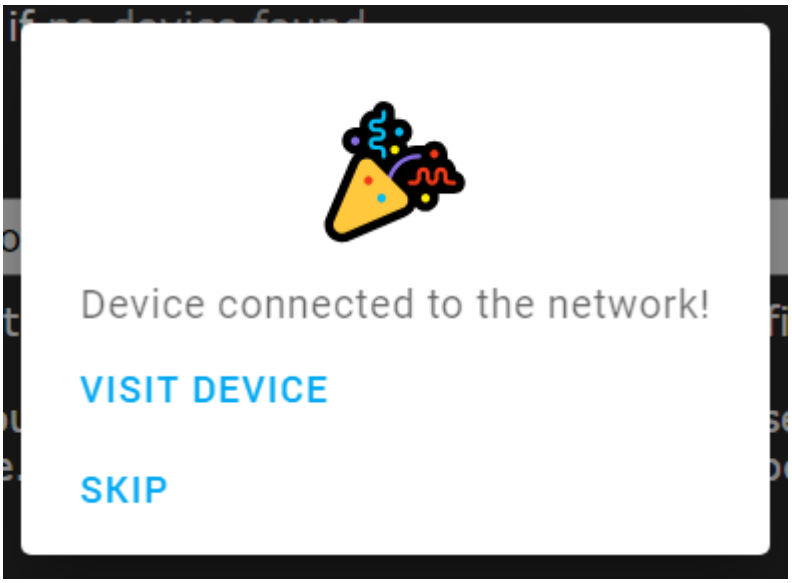


> **Tip**
>
> Dabei sollte ein Access-Point gewählt werden, bei dem die Kombination SSID / Kanalnummer in der Reichweite des ESP32 kein zweites Mal vorkommt. Mesh-Netzwerke bereiten Probleme! Bei mir funktioniert Tasmota nicht mit dem AiMesh von ASUS.

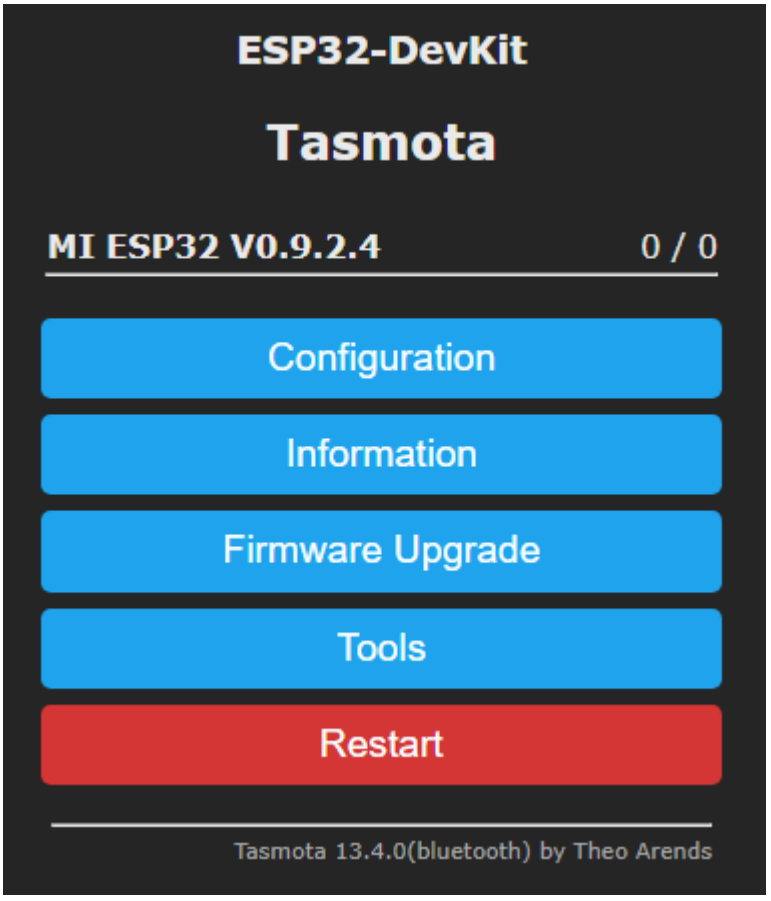
Weiter mit `CONNECT`



Wenn die Verbindung hergestellt ist, `VISIT DEVICE` „



... und dann ist das Hauptmenü zu sehen:



Die ESP32-Software wird ständig weiter entwickelt, die Version 0.9.2.4 ist somit nur eine Momentaufnahme.

Konfigurieren

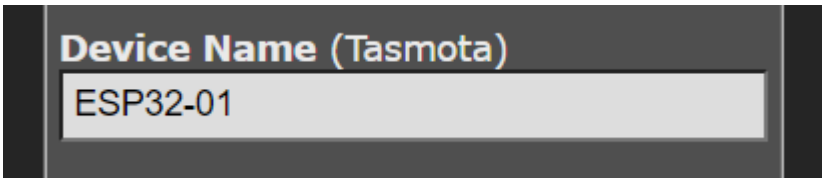
IP-Adresse

Eine fixe IP-Adresse ist recht praktisch, weil damit mehrere Prozessoren leichter verwaltet werden können. In dem Beispiel soll die Adresse auf 192.168.50.1 geändert werden:

Über `Tools` → `Console` die Konsole aufrufen. Mit `IPAddress1 192.168.50.1` und `Tools` → `Main Menu` → `Restart` erhält der ESP32 seine neue Adresse und wird nun über `http://192.168.50.1` angesprochen.

Device Name

Wer dem ESP32 einen schönen Namen geben will, wählt `Configuration` → `Configure other`

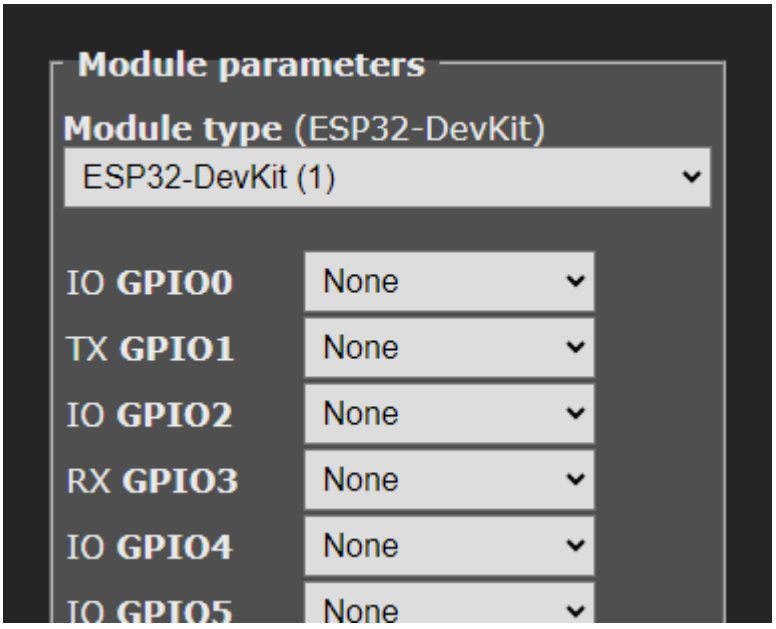


Na gut, vielleicht gibt es kreativere Bezeichnungen.

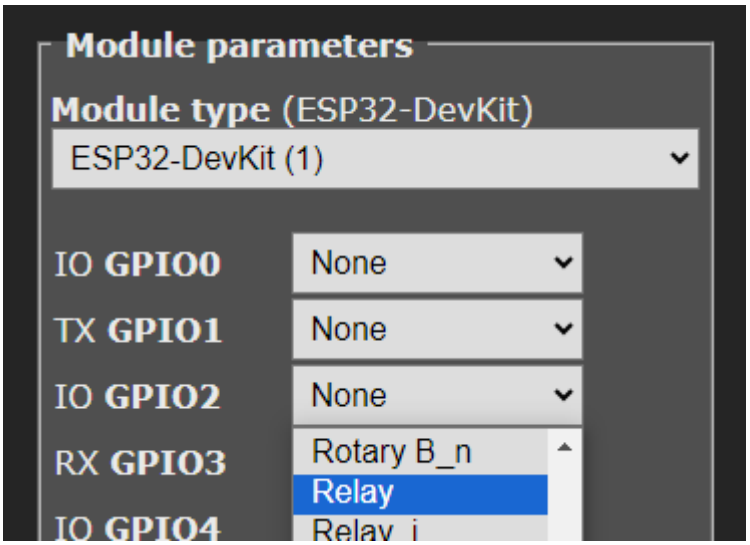
Ein-/Ausgabeleitungen

Die meisten Anschlüsse können wahlweise als Eingangs- oder Ausgangsleitungen konfiguriert werden, Eingangsleitungen auch mit einem Pullup- oder Pulldown-Widerstand. Ferner sind Konfigurationen als serielle Schnittstelle, I^2C-, CAN- oder SPI-Anschluss möglich. Auch analoge Signale können ein- und ausgegeben werden. Und das ist immer noch nur ein kleiner Überblick über mögliche Funktionen.

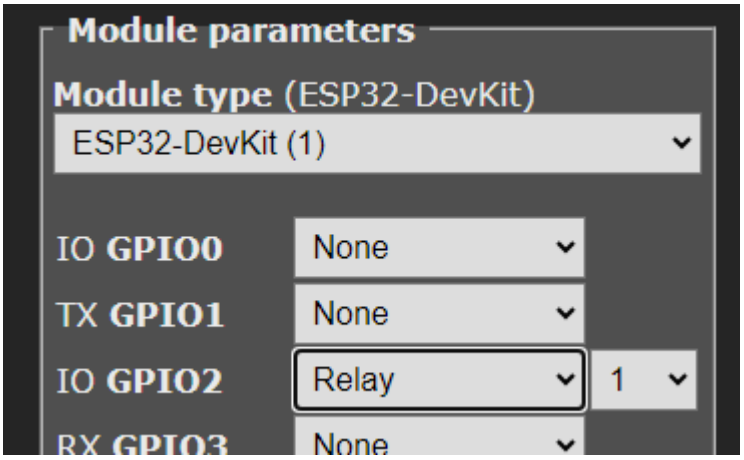
In Projekten mit dem ESP32 werden Programmbibliotheken und Entwicklungswerkzeuge eingesetzt. Mit Tasmota geht das viel einfacher. Über `Configuration` → `Configure Module` erscheint eine Liste der Ein-/Ausgabeleitungen (GPIO = General Purpose Input/Output, *Allzweckein-/ausgabe*). Hier ein Ausschnitt:



Um beispielsweise der Leitung 2 (GPIO2) eine bestimmte Funktion zuzuweisen, wird die Auswahl-Liste neben GPIO2 angeklickt und eine Funktion, zum Beispiel `Relay` ausgewählt:



Das Relay *(oder Relais)* erhält automatisch die Nummer 1 — das kann auch geändert werden.

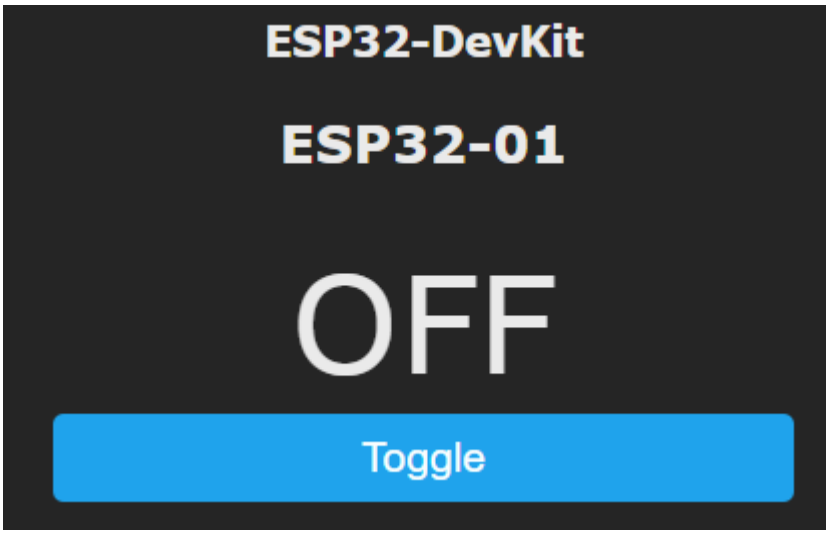


> **Note**
>
> Die Ausgangsleitung wird hier als `Relay` `1` ausgewählt, in der Steuerung aber als `Power1` bezeichnet. Etwas verwirrend…

Nicht vergessen: mit `Save` abspeichern und mit `Main Menu` zurück ins Hauptmenü. Nach ein paar Sekunden sehen wir den Zustand des neuen Relay-Ausgangs (*OFF*) und eine zusätzlichen Schaltfläche `Toggle` zum Umschalten:

Mit einigen wenigen Anweisungen wird eine Ausgangsleitung festgelegt.

- Wir könnten eine LED samt Vorwiderstand anschließen, um die folgenden Schritte verfolgen zu können: siehe Abschnitt Externe Sensoren und Aktoren.
- Oder wir beobachten den Zustand von *Relay 1* einfach im Hauptmenü:



Steuerung eines Tasmota-Geräts

Befehle (*Commands*) zur Steuerung können über das Web User Interface (*webUI*), über die Webschnittstelle (*web request*), über die Konsole, über den seriellen Anschluss und über MQTT abgesetzt werden. Die letzten zwei Varianten werden hier nur in der Langversion behandelt.

Steuerung über die Benutzeroberfläche

Die Benutzeroberfläche haben wir im vorherigen Abschnitt gezeigt. Mit Klicks auf `Toggle` wird das Ausgangssignal umgeschaltet.

Ein anderes Beispiel ist im Abschnitt LED-Lichterkette zu finden.

Steuerung über die Webschnittstelle

Jede Tasmota-Komponente hat eine IP-Adresse. Über diese Adresse sind Steuerungsbefehle möglich. Nachteil: die Verbindung ist nicht bidirektional und muss für jeden Befehl neu aufgebaut werden.

So wird ein Befehl über einen Browser-Aufruf geschickt.

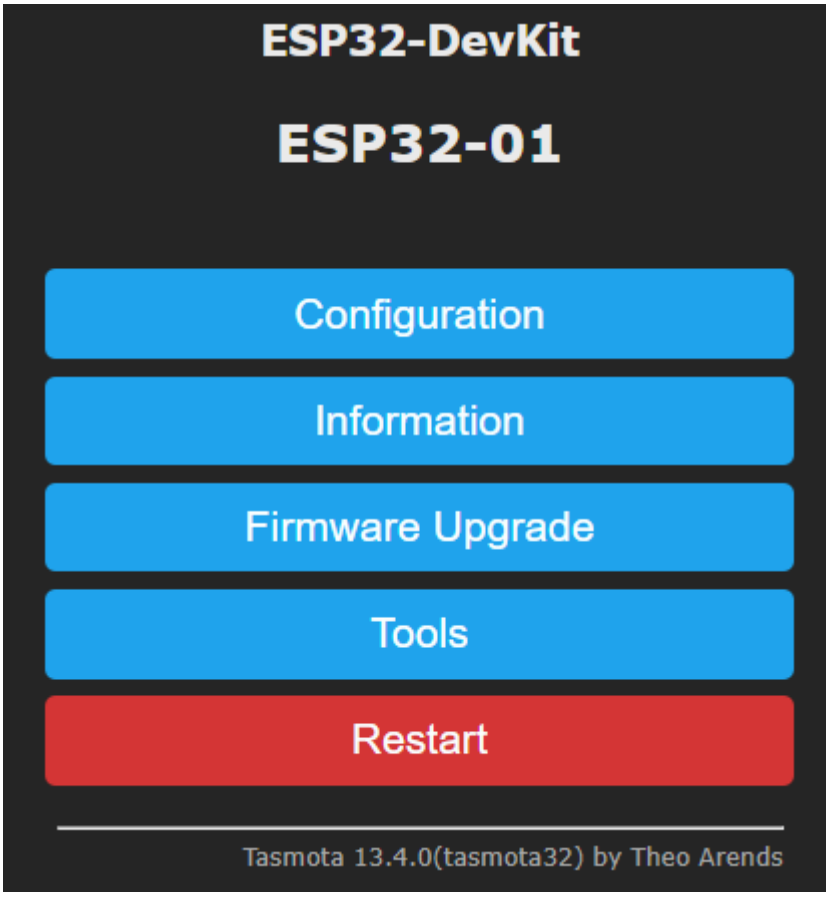
```
http://IP/cm?cmnd=COMMAND%20DATA
```

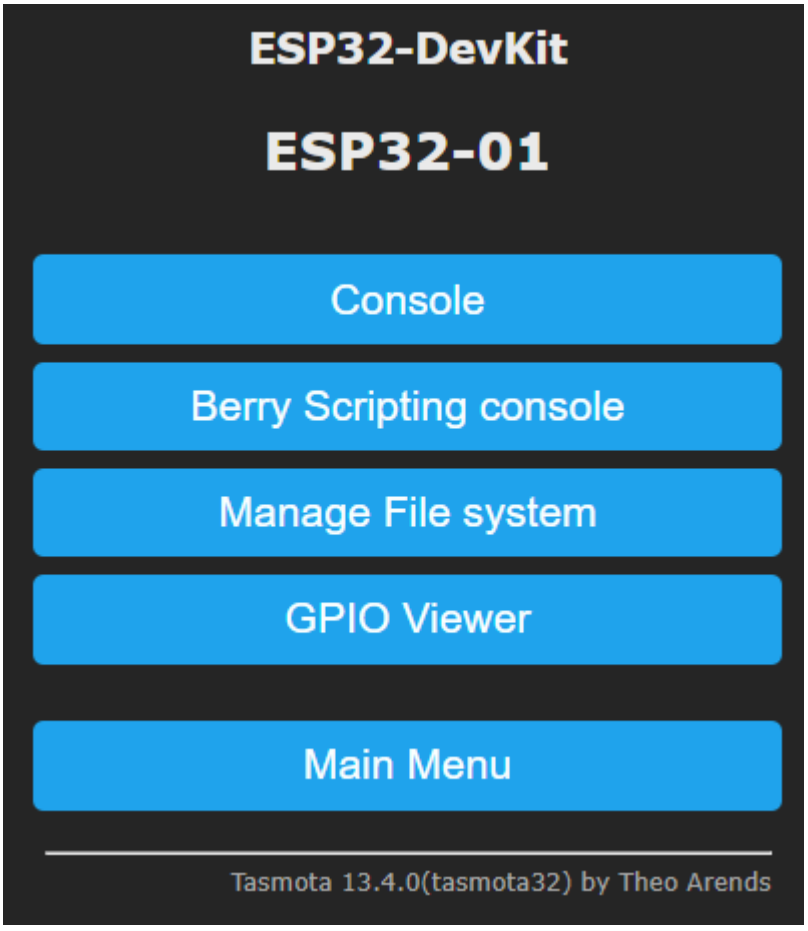
- `IP`: die IP-Adresse der Komponente
- `COMMAND`: der Befehl
- `%20`: steht für das Leerzeichen zwischen COMMAND und DATA
- `DATA`: Daten

> Beispiel: Umschalten des ersten Relais-Ausganges der Tasmota-Komponente mit der Adresse 192.168.50.1:
>
> ```
> http://192.168.50.1/cm?Power1%20TOGGLE
> ```
>
> Das Leerzeichen wird durch %20 ersetzt, da es in einem HTTP-Aufruf nicht vorkommen darf. Auch andere spezielle Zeichen müssen durch ihre ASCII-Codes ersetzt werden.

Steuerung über die Konsole

Die Konsole wird (ab Version 13.4) mit `Tools` → `Console` aufgerufen, bei Tasmota 13.0 bis 13.3 über `Consoles` → `Console` oder vor 13.0 nur über `Console`.

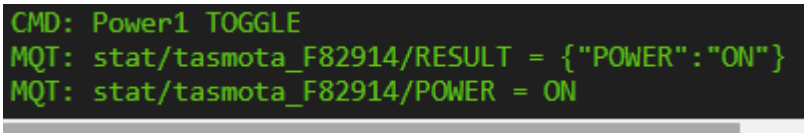




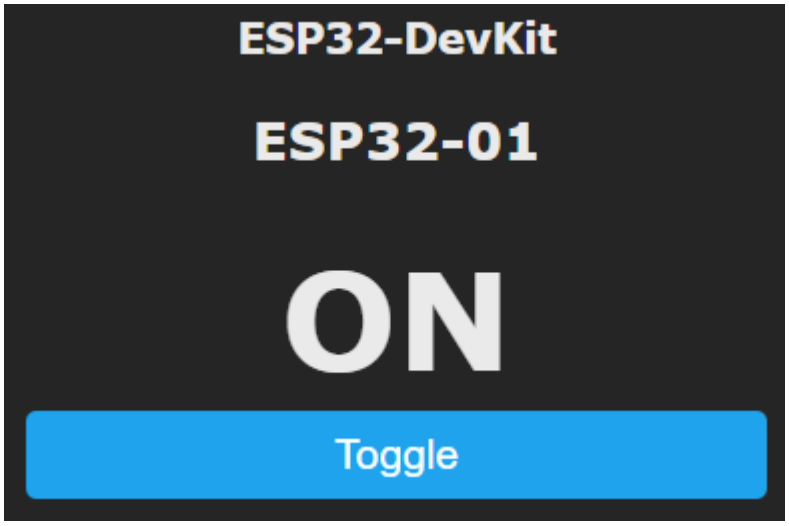
Über die Konsole können alle Befehle, die die Tasmota-Software kennt, eingegeben werden. Die Webschnittstelle ist im Gegensatz dazu nicht so allgemein verwendbar.
Ein Beispiel: der Konsolen-Befehl

```
Power1 TOGGLE
```

schaltet den Ausgang GPIO2, also unser Relais 1, von *aus* auf *ein*.



Zum Beweis, dass hier tatsächlich die Ausgangsleitung umgeschaltet worden ist, öffnen wir ein zweites Browser-Fenster oder gehen über `Tools` → `Main Menu` zurück auf die Startseite (in das Hauptmenü):



Nochmal in der Konsole `Power1 TOGGLE` eingeben und der Ausgang schaltet wieder aus.

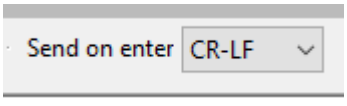
Allgemein:

- Jeder Befehl besteht aus dem Kommando und (optional) den Daten.
- In der Tasmota-Dokumentation wird ein Befehl allgemein so angegeben:
  `COMMAND<x> DATA`
  Dabei ist `<x>` ein Index, üblicherweise eine ganze Zahl ab 1. Zwischen `COMMAND` und steht *kein Zwischenraum*, zwischen `COMMAND<x>` und `DATA` *genau ein*er. Ist der Index gleich `1`, kann er auch weggelassen werden.
- Befehle ohne Parameter (ohne `DATA`) geben den aktuellen Wert der Datengröße zurück.

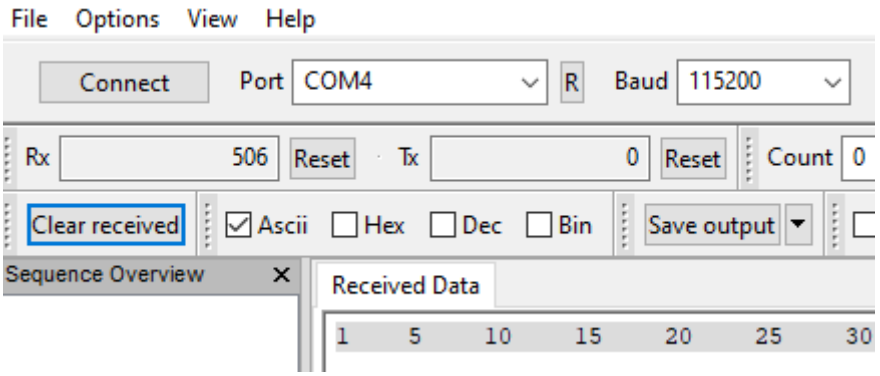
Steuerung über die serielle Schnittstelle

Der ESP32 wird über ein USB-Kabel mit dem PC verbunden. Ferner wird ein Terminalprogramm benötigt, zum Beispiel hterm. Der Port wird automatisch vorgeschlagen, die Baudrate ist 115200.

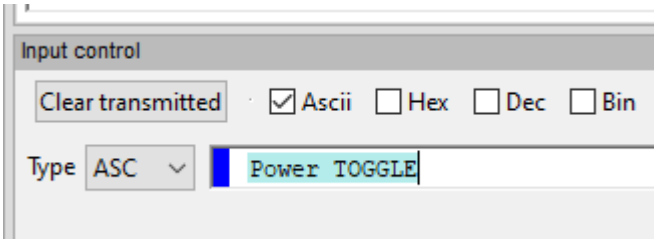
`Send on enter` auf `CR-LF` stellen.



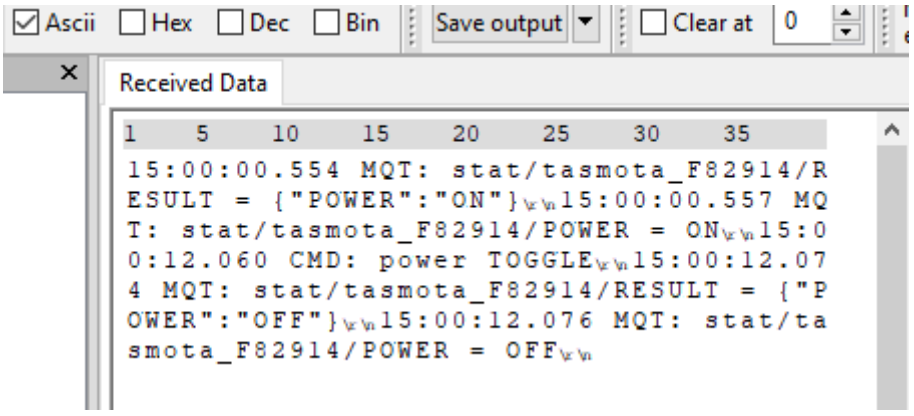
Dann `Connect`



Schon kann es losgeben:



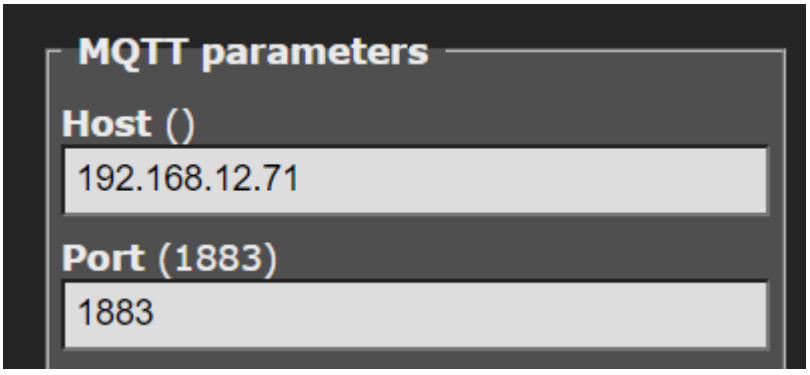
Als Antwort wird



angezeigt. Im Hauptmenü ist zu sehen, dass der Ausgang umgeschaltet worden ist.

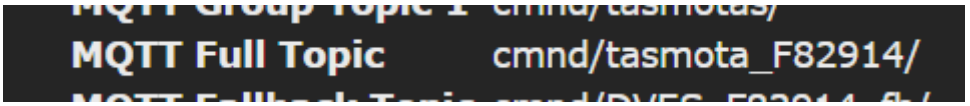
Steuerung über MQTT

Um MQTT-Befehle verwenden zu können, muss im LAN ein MQTT-Broker installiert und die Adresse dieses Brokers in `Configuration` → `Configure MQTT` eingetragen sein:

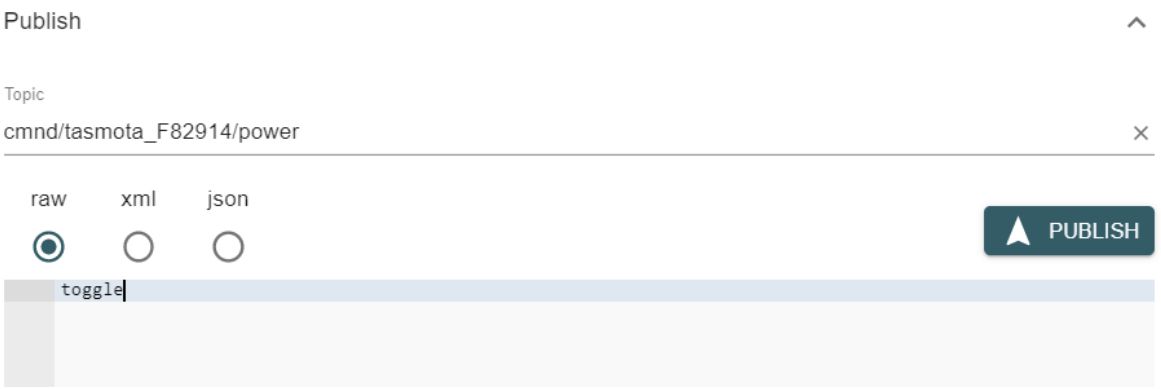


Die IP-Adresse entsprechend dem eigenen Netz ändern. Der Rest kann bleiben.

Jede Tasmota-Komponente hat ihr eigenes *MQTT Full Topic*, zu finden unter `Information`. Damit werden die MQTT-Befehle an das gewünschte Gerät geschickt.



Um MQTT-Befehle absetzen zu können, ist auch noch ein MQTT-Client-Programm notwendig, zum Beispiel der MQTT Explorer. Sobald die eigenen Daten auf der Startseite eingetragen sind (Protokoll: mqtt://, Host (bei mir): 192.168.12.71, Port:1883) und die Verbindung mit `CONNECT` hergestellt ist, kann es losgehen:



Ein Klick auf `PUBLISH` und der Ausgang wird umgeschaltet.

Zusammenfassung

Mit diesen Befehlen kann jedes Tasmota-Gerät gesteuert werden, so auch beispielsweise ein Zwischenstecker mit Tasmota-Software wie der A1T von NOUS. Siehe PCNEWS 177.

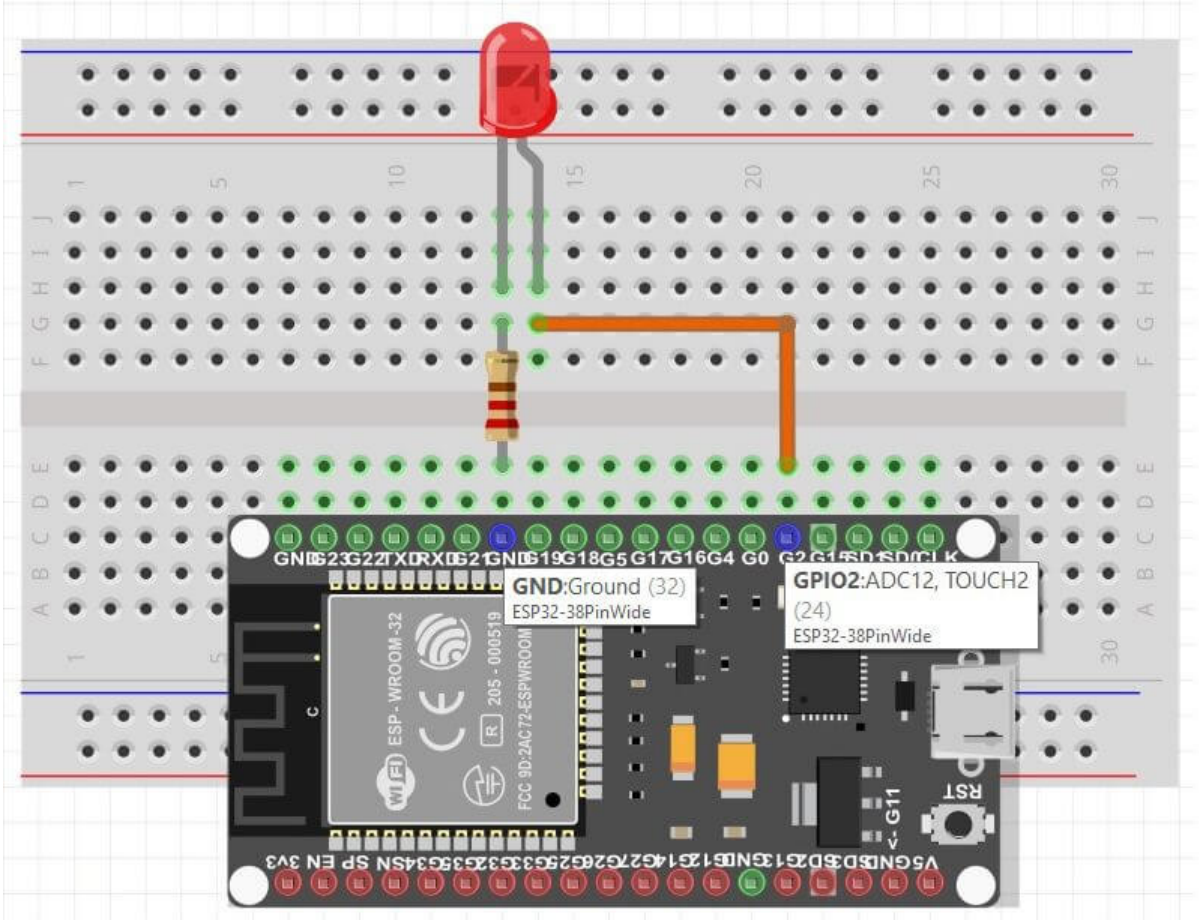
Externe Sensoren und Aktoren

LED anschließen

Bis jetzt haben wir den Zustand des Ausgangs nur über die Software überprüft.

Wer will, kann eine beliebige LED zwischen Anschluss GPIO 2 (Pin 24) und über einen 220 Ω-Widerstand an Masse (Ground, GND, Pin 32) anschließen und damit den Umschaltvorgang direkt beobachten.

Bitte beachten: Der lange Anschluss der LED ist die Anode und gehört an den GPIO-Anschluss. Der kurze Anschluss ist die Kathode: diese wird über den Widerstand an Masse gelegt. Beim Kathodenanschluss ist das Gehäuse der LED außerdem abgeflacht.



Dieses Bild habe ich bei uelectronics.com gefunden. Die Spannung wird über ein Mikro-USB-Kabel und den Stecker rechts zugeführt (hier nicht abgebildet).

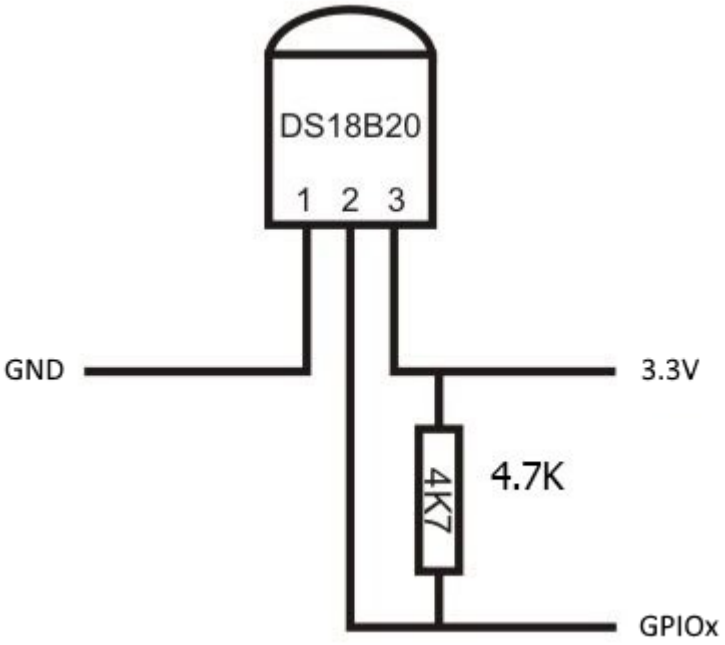
Diese Pin-Nummern gelten für den ESP32 in der 38-Pin-Variante:

Quelle: ebenfalls uelectronics.com

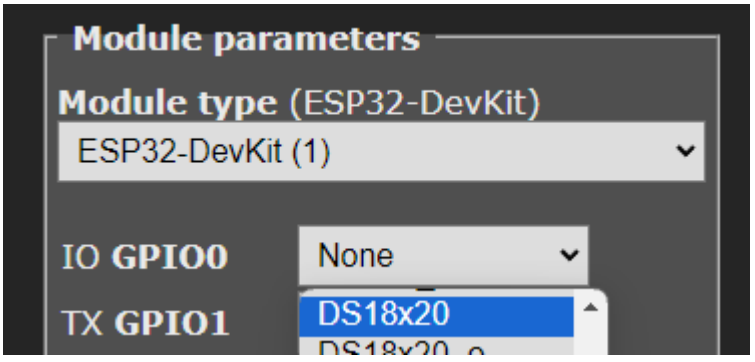
Temperaturen messen mit dem DS18B20

Temperaturen wurden früher nur analog mit temperaturabhängigen Widerstanden gemessen. Mit dem DS18B20 geht das nun digital in 0,5 °C Schritten (oder genauer).

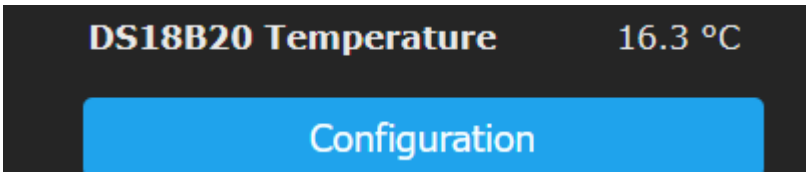
Der DS18B20 arbeitet mit Versorgungsspannungen von 3,0 Volt bis 5,5 Volt und kann daher aus derselben Spannungsquelle wie der ESP32 (3,3 Volt) betrieben werden. Den DS18B20 gibt es verschiedenen Varianten. Bei der Ausführung mit einer dreiadrigen Anschlussleitung wird die rote Leitung(3) an die Versorgungsspannung (*3.3 V*), die schwarze an Masse (*GND*) (1) und die gelbe Datenleitung (2) — an einen GPIOx, zum Beispiel *GPIO0* = GPIO-Null, angeschlossen. Dann ist noch ein Pullup-Widerstand mit 4,7 kΩ (Farbcode gelb-violett-rot) zwischen Versorgungsspannung und Datenleitung zu schalten - fertig!



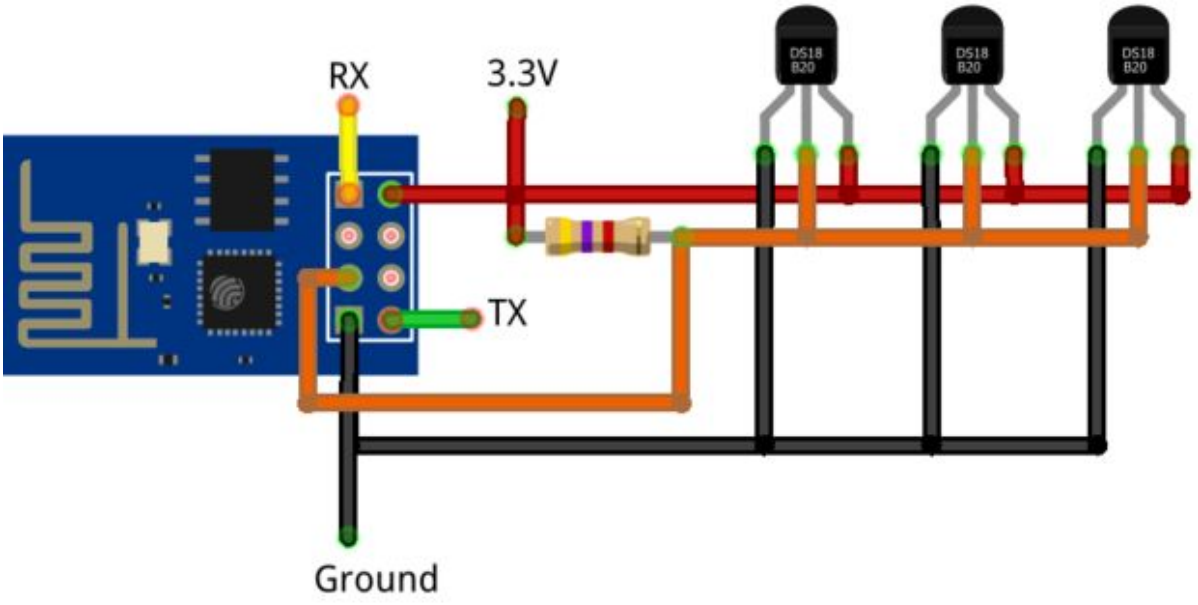
Die Messwerte werden über ein serielles Protokoll ausgelesen, das im Datenblatt genau beschrieben ist. Aber auch hier haben die Tasmota-Leute vorgesorgt: die Einstellung über die Tasmota-Firmware ist genau so einfach, wie die Hardwarevorbereitung. Vom Hauptmenü aus zu `Configuration` → `Configure Module` gehen und für einen GPIO-Anschluss (hier: GPIO0) `DS18x20` auswählen.



Nach einem `Save` wird im Hauptmenü die Temperatur 16,3 °C angezeigt:



8 bis 11 Stück DS18B20 können an *einen* GPIO und *einen* Pullup-Widerstand parallel angeschlossen werden, hier ein Beispiel mit einem ESP01 und dem GPIO2. Bei mehr Sensoren gibt es Probleme mit der Anzeige. Der Pullup-Widerstandswert ist ggf. zu verringern, zum Beispiel auf 3,3 kΩ. Außerdem kann auch *mehr als ein* GPIO-Anschluss zur Temperaturmessung verwendet werden.



Jeder DS18B20 hat eine eindeutige 64-Bit-Adresse — damit werden die einzelnen Messwerte unterschieden.

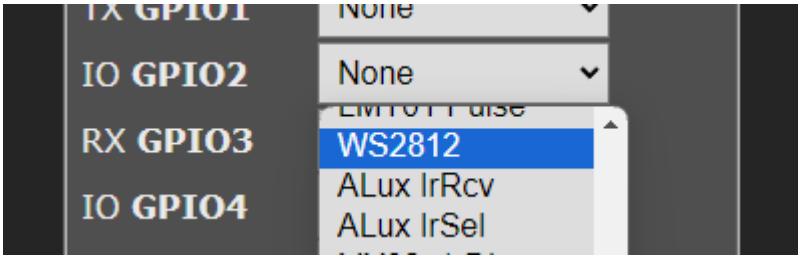
> **Tip**
>
> Wird *nur ein* DS18B20 an einen ESP32 angeschlossen, kann statt des externen Pullup-Widerstands der interne verwendet werden. Dazu ist über die Konsole `SetOption74 1` einzugeben.

LED-Lichterkette

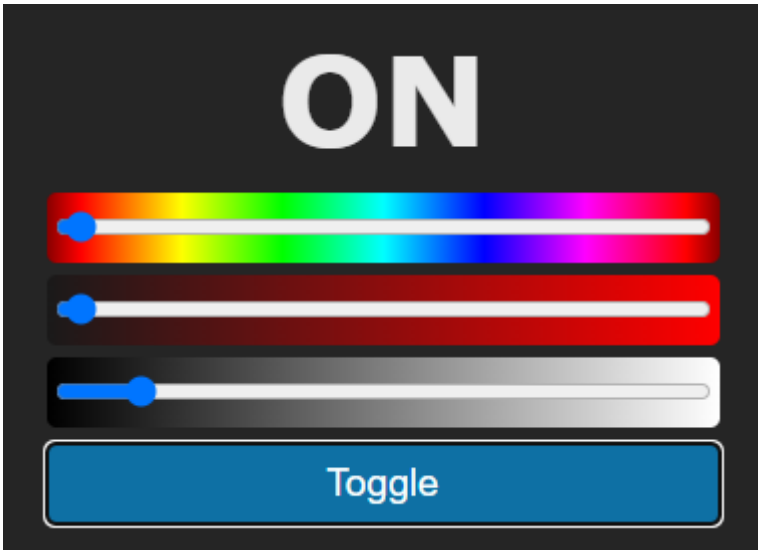
Der Bauteil WS2812 besteht aus einer dreifarbigen LED (RGB-LED) und der Steuerelektronik mit Schieberegister samt Speicher. Die vier Anschlüsse laut Datenblatt und ihre Verbindung mit dem ESP32 sind:

- der Dateneingang (DIN): über einen 270 Ω Widerstand mit *GPIO2* verbinden
- der Datenausgang (DOUT): bleibt frei oder wird mit dem nächsten WS2812-Dateneingang verbunden
- die Versorgungsspannung (VDD): verbinden mit dem *Vin 5V* Anschluss und
- Masse (VSS), verbinden mit *GND* (Ground)

Tasmota ist softwaremäßig dafür schon vorbereitet: über `Configuration` → `Configure module` wird als Ausgangssignal für den `GPIO2` `WS2812` ausgewählt.



Nach `Save` startet der Prozessor neu. So sieht jetzt das Hauptmenü aus;



Über die drei Schieberegler werden die Farbe (Hue), die Sättigung (Sat) und die Helligkeit (Brightness) eingestellt. `Toggle` schaltet alle LEDs gleichzeitig ein bzw. aus.

Mit mehreren hintereinander geschalteten WS2812 werden Lichterketten oder Ringe aufgebaut. Tasmota stellt dafür viele Befehle bereit. Sogar eine Uhr mit unterschiedlichen Farben für Stunden-, Minuten- und Sekunden-*Zeiger* ist mit ein paar Befehlen über die Konsole eingerichtet!

Noch besser klappt die Steuerung einer Lichterkette mit Berry-Programmen (siehe unten).

Tasmota ohne Broker

Ein Tasmota-Gerät steuert ein anderes (Teil 1)

Um Tasmota-Komponenten in einem Smart Home zu verwenden, wird normalerweise eine Steuerungszentrale, in der Regel ein MQTT-Broker, verwendet. Aber können Tasmota-Komponenten im Netzwerk auch einander, also ohne Zentrale, beeinflussen? Die Antwort lautet ja und ist im Detail in der Langfassung beschrieben.

Das erste Tasmota-Gerät (mit der Adresse http://192.168.50.1) bleibt unverändert, das Fenster bleibt offen. Für das zweite Tasmota-Gerät (zum Beispiel für einen ESP32 mit der Adresse http://192.168.50.2) wird ein neues Fenster zum Senden der Steuerbefehle geöffnet.

Wir öffnen die Konsole im Fenster des zweiten Geräts mit `Tools` → `Console`. Der zweite ESP32 steuert den ersten über `websend`. Wie der Name nahe legt, wird über das *Web* (das Netz) ein Kommando ge*send*et:

Eingabe: Die eckigen Klammern gehören dazu. Ob Groß- oder Kleinbuchstaben verwendet werden, ist unerheblich.

```
websend [192.168.50.1] /cm?cmnd=power%20toggle
```

Das Leerzeichen vor `toggle` muss nicht durch `%20` ersetzt werden.

```
websend [192.168.50.1] /cm?cmnd=power toggle
```

Da der Befehl an ein anderes Tasmota-Gerät gesendet wird, geht es noch einfacher:

```
websend [192.168.50.1] Power TOGGLE
```

Rules zum Senden des Befehls

Wenn ein bestimmtes Ereignis auf einem Tasmota-Gerät eine Reaktion auslösen soll, sind Regeln *(Rules)* zu verwenden. Rules sind mächtige, aber auch etwas gewöhnungsbedürftige Werkzeuge. Das Ziel war wohl, Regeln möglichst platzsparend abzuspeichern

Grundsätzlicher Aufbau einer Regel:

```
ON <trigger> DO <command> ENDON
```

- `ON`, `DO`, `ENDON` sind Schlüsselworte und genau so zu übernehmen. Groß- und Kleinbuchstaben sind aber irrelevant.
- `<trigger>` steht für eine Bedingung, die den Befehl `<command>` auslöst. Zum Triggern gibt es eine große Auswahl an Möglichkeiten. Ein Triggersignal kann beispielsweise das Drücken einer Taste, eine Messwertänderung oder der Aufbau der WLAN-Verbindung sein. Zum Testen wird hier `<trigger>` auf `Time#Minute` gesetzt, das heißt der Trigger wird zu jeder vollen Minute aktiviert.
- `<command>` ist der auszuführenden Befehl. Um den Relaisausgang des Prozessors wie vorhin umzuschalten, lautet der Befehl::

  `Power TOGGLE`

Eine Regel oder mehrere Regeln zusammen bilden einen *Regelsatz*. Dieser wird mit `Rule<x>` eingeleitet. `<x>` ist dabei die Nummer der Regelsatzes: 1, 2 oder 3.

> **Note**
>
> In der Dokumentation wird statt des Wortes *Regelsatz* auch das Wort *Regel* sowohl für eine einzelne Regel wie auch für einen Regelsatz verwendet - das ist etwas verwirrend.

Alles zusammen bildet den Regelsatz 1. Zurück zur Konsole des ersten Geräts.: hier wird eingegeben:

```
Rule1 ON Time#Minute DO Power TOGGLE ENDON
```

Der Regelsatz 1 (*Rule1*) muss noch aktiviert werden, das heißt sein Zustand (*State*) wird auf 1 gesetzt.

```
Rule1 1
```

Nun schaltet das erste Gerät im Minutentakt den Ausgang um.

```
CMD: Rule1 ON Time#Minute DO Power TOGGLE ENDON
RUL: Stored uncompressed, would compress from 36 to 29 (-19%)
MQT: stat/tasmota_F82914/RESULT = {"Rule1":{"State":"OFF","Once":"OFF
CMD: rule1 1
MQT: stat/tasmota_F82914/RESULT = {"Rule1":{"State":"ON","Once":"OFF"
RUL: TIME#MINUTE performs "Power TOGGLE"
MQT: stat/tasmota_F82914/RESULT = {"POWER":"ON"}
MQT: stat/tasmota_F82914/POWER = ON
```

Aus Platzgründen kann nicht die gesamte Zeile dargestellt werden.

> **Note**
>
> Die Abfrage `Time#Minute` funktioniert nur, wenn Tasmota die aktuelle Zeit kennt. Das geschieht normalerweise automatisch, da Tasmota das Network Time Protocol (ntp) verwendet und auf ntp-Server, wie beispielsweise pool.ntp.org, zugreift. Wegen eines Problems bei der Namensauflösung in IPv4 bzw. IPv6 war bei mir der Zugriff auf die Timeserver nicht möglich. Ich habe daher beim ersten Timeserver-Eintrag den Namen durch die echte IP-Adresse ersetzt. Das ist nicht ideal, da bei einer Änderung der IP-Adresse des Timeservers die Zeitsteuerung von Tasmota wieder ausfällt, aber zumindest war das Problem gelöst. Eingabe in der Console:
>
> ```
> ntpserver1 83.137.41.14
> ```

Ein Tasmota-Gerät steuert ein anderes (Teil 2)

Die letzten beiden Beispiele können zusammengefasst werden: das zweites Tasmota-Gerät soll das erste zeitgesteuert umschalten.

Dazu deaktivieren wir auf dem ersten Gerät den Regelsatz 1:

```
Rule1 0
```

Auf dem zweiten Gerät tragen wir einen Regelsatz zum zeitgesteuerten Umschalten von Power des ersten Tasmota-Gerät ein und aktivieren diesen Regelsatz:

```
Rule1 ON Time#Minute DO WebSend [192.168.50.1] Power TOGGLE ENDON
Rule1 1
```

```
CMD: Rule1 ON Time#Minute DO WebSend [192.168.50.1] Power TOGGLE ENDON
RUL: Stored uncompressed, would compress from 59 to 48 (-19%)
MQT: stat/tasmota_69291C/RESULT = {"Rule1":{"State":"OFF","Once":"OFF","Sto
CMD: rule1 1
MQT: stat/tasmota_69291C/RESULT = {"Rule1":{"State":"ON","Once":"OFF","Stop
```

Am ersten Gerät können wir nun in der Konsole, im Hauptmenu oder über eine angeschlossene LED beobachten, dass der Ausgang zu jeder vollen Minute umgeschaltet wird.

Berry

Mit Rules können Signale in Abhängigkeit von Ereignissen oder Bedingungen geschaltet werden. Das ist auch dann interessant, wenn ein Tasmota-Gerät ohne einen Broker betrieben werden soll. Der Funktionsumfang von Rules zwar sehr umfangreich, aber dadurch beschränkt, dass keine *Abläufe* programmiert werden können. Berry ist eine gut ausgebaute Skriptsprache, die (durch Compilieren) sehr wenig Speicherplatz benötigt, einfach zu verwenden ist und mit einigen Erweiterungen auf alle Tasmota-Funktionen zugreifen kann. Allerdings ist ein ESP32 notwendig, da Berry in einem ESP8266 zusammen mit der restlichen Tasmota-Firmware keinen Platz hat. Etliche Tasmota-Geräte scheiden somit für Berry aus. Steuerungen, die auf einem ESP32 beruhen, eignen sich dagegen hervorragend für den Einsatz von Berry.

Die Programmiersprache Berry erinnert an Python oder MicroPython. Die Sprache kann auch online über eine eigene Webseite mit Testumgebung oder direkt am ESP32 erprobt werden. Mit `Tools` → `Berry Scripting console` erscheint (ab Tasmota Version 13) eine spezielle Konsole mit kleinem Editor.

Zum Ausprobieren eine Berry-Anweisung in das weiße Feld eingeben – z. B. `print ("Hallo Berry")` – und zweimal(!) die `Enter`-Taste drücken:

Im Berry-Cookbook werden ein paar Anwendungen beschrieben. Die Idee ist gut, aber es sollten viel mehr Beispiele (auch besonders einfache!) und wesentlich umfangreichere Dokumentationen betreffend die Tasmota-spezifischen Funktionen veröffentlich werden. Ich sammle und veröffentliche gerne eigene Erfahrungen von Lesern. Die Telegram-Gruppe The Berry Lang ist nicht sehr aktiv, Berry-Themen werden vor allem in den Telegram Tasmota-Gruppen behandelt.

Berry eröffnet viele Möglichkeiten und ist einen eigenen Artikel wert.

Matter

Viele große Unternehmen haben zu Beginn der Smart Home-Welle eigene Lösungen entwickelt. Und wie so oft bei neuen technischen Entwicklung war Kompatibilität nicht das oberste Ziel - sehr zum Leidwesen der Konsumenten! Die Versuchung, den eigenen Gewinn nach Möglichkeit durch proprietäre Lösungen zu maximieren, statt gemeinsam mit anderen Markteilnehmern Standards zu vereinbaren, ist offenbar sehr groß. Solange die Konsumenten ahnungslos sind, klappt das auch meistens. Inkompatibilitäten können jedoch zum Bumerang führen und eine Marke nachhaltig beschädigen. (Wer weiß heute noch, was der *Internet Explorer* war?)

ZigBee war schon ein guter Schritt in Richtung Zusammenarbeit. Besser ist Matter: ein gemeinsamer Standard der größten Unternehmen mit einer stark erweiterten Liste von Produktgruppen. Da Matter als lizenzfreier, offener Stand definiert wurde, ist eine rasch wachsende Verbreitung zu erwarten und zu erhoffen.

In Tasmota ist ebenfalls schon ein Teil des Matter-Standards im Konfigurationsmenü abrufbar:

Zusammenfassung

Dieser Beitrag hat verschiedene Aspekte der Tasmota-Software aufgezeigt und in der Langfassung dazu angeregt, selbst Experimente zu wagen. Gerade für eigene Experimente mit dem Mikroprozessor ESP32 ist die Tasmota-Firmware sehr gut geeignet, da sie viele Funktionen des ESP32 ohne großen Programmieraufwand zur Verfügung stellt.

Nachtrag: ZigBee und Tasmota

Ich habe schon zweimal über die Themen *Das automatische Heim* und *ZigBee und Tasmota* geschrieben. Hier die Hinweise zu den früheren Artikeln.

- Das automatische Heim mit ZigBee
  PCNEWS 172, 09.02.2022
- ZigBee, Zwischenstecker und Tasmota
  PC NEWS 177, 06.05.2023

Leider ist mir dabei ein sinnstörender Fehler passiert.

- ZigBee-Komponenten bauen ein eigenes Mesh-Netzwerk auf, somit ein Netzwerk, in dem die Komponenten selbstständig die optimalen Routen zum Datentransfer ermitteln.
- Tasmota-Komponenten sind nicht Teil dieses ZigBee-Mesh-Netzwerks, verwenden das im Haus installierte WLAN und können bei stark ausgelasteten WLAN-Kanälen durch gegenseitiges Stören Schwierigkeiten mit der Kommunikation bekommen.

Komponenten, die mit Tasmota geflasht (*tasmotisiert*) worden sind, kommunizieren also **nach wie vor über TCP/IP** und müssen über das WLAN erreichbar sein.

ZigBee und der Tasmota-WLAN-Zugang arbeiten nur im 2,4-GHz-Band. Ein Ausweichen auf 5-GHz-Kanäle – die oft noch nicht so stark belegt sind - ist nicht möglich.

Zur Steuerung wird in beiden Fällen das Protokoll MQTT verwendet:

- Bei ZigBee-Komponenten wird die MQTT-Verbindung durch das *Pairing* oder *Anlernen* eingerichtet.
- Bei Tasmota-Komponenten wird nach Aufruf der IP-Adresse über `Configuration` → `Configure MQTT` → `MQTT parameters` → *Host* der MQTT-Host (auch MQTT-Broker oder MQTT-Server genannt) eingetragen.

Weitere Formen der Steuerungen:

- Tasmota-Komponenten können direkt über ihre http-Adresse angesprochen werden. Diese Verbindung ist nicht bidirektional und muss immer neu aufgebaut werden.
- Ferner kann ein Tasmota-Gerät über eine serielle Schnittstelle (falls vorhanden) angesprochen werden. Diese Funktion ist beim Testen recht nützlich, da sie unabhängig von einer eventuell instabilen Funkverbindung arbeitet.

Viel Erfolg und viel Spaß mit Tasmota!