Shelly Pro 3EM Emulator

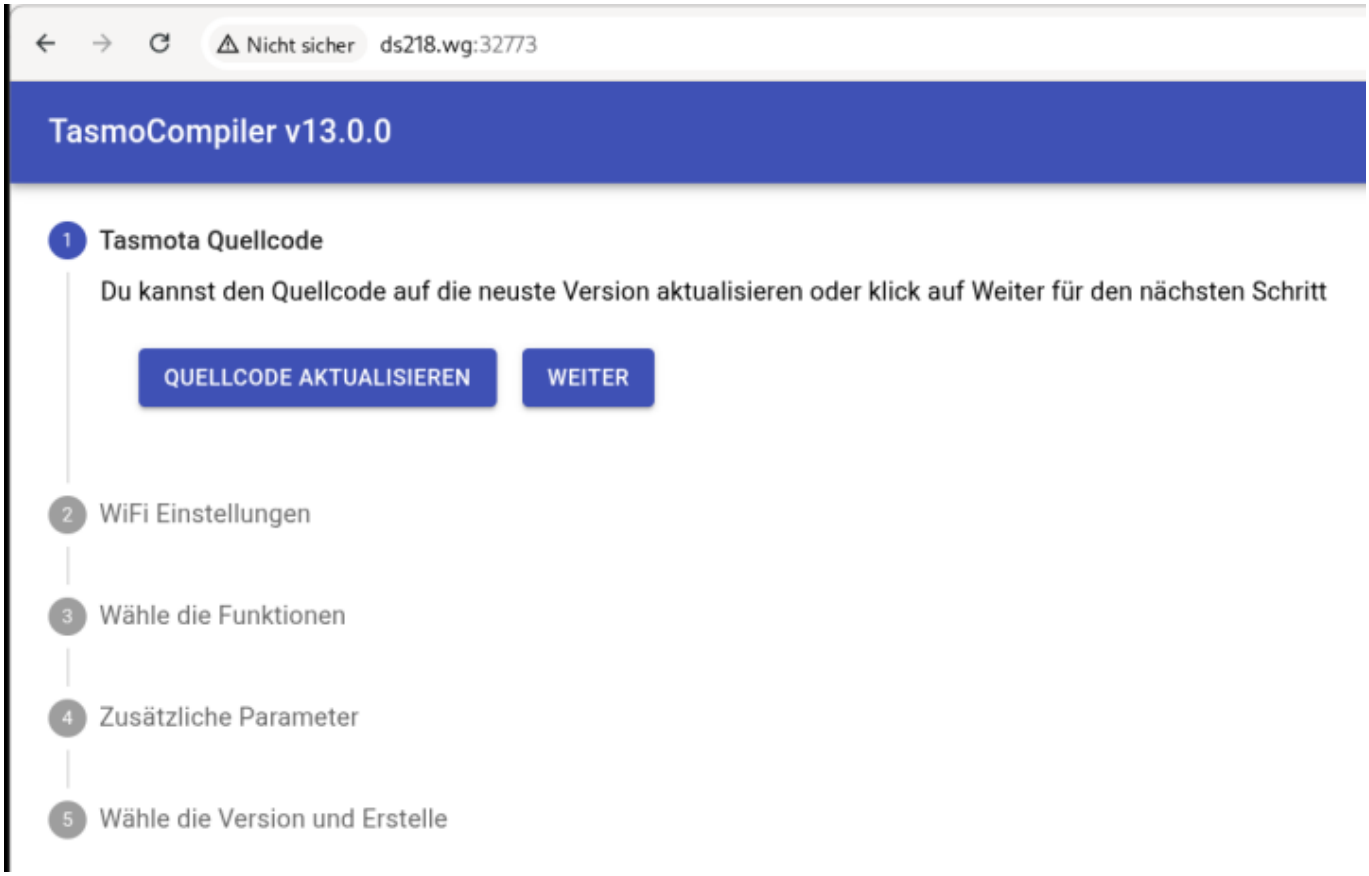
Quellen

- Tasmota Web Installer
- Tasmota Smart Meter Interface Documentation
- Tasmota based Shelly Pro 3EM Emulation
- 2_SML_Script_Chart_PV_ShellyEmu.tas
- Release of Tasmota precompiled binary version 15.1.0 for ESP32
- TasmoCompiler - all in one compiler suite based on Docker container with comfortable web gui
- WEMOS Lolin ESP32-S3 Mini Schaltplan
- Stromzähler mit einem ESP8266 / ESP32 mit Tasmota auslesen und darstellen
- EFR Zähler Scripts
- WEMOS ESP S3 Mini Beschreibung und Pinbelegung

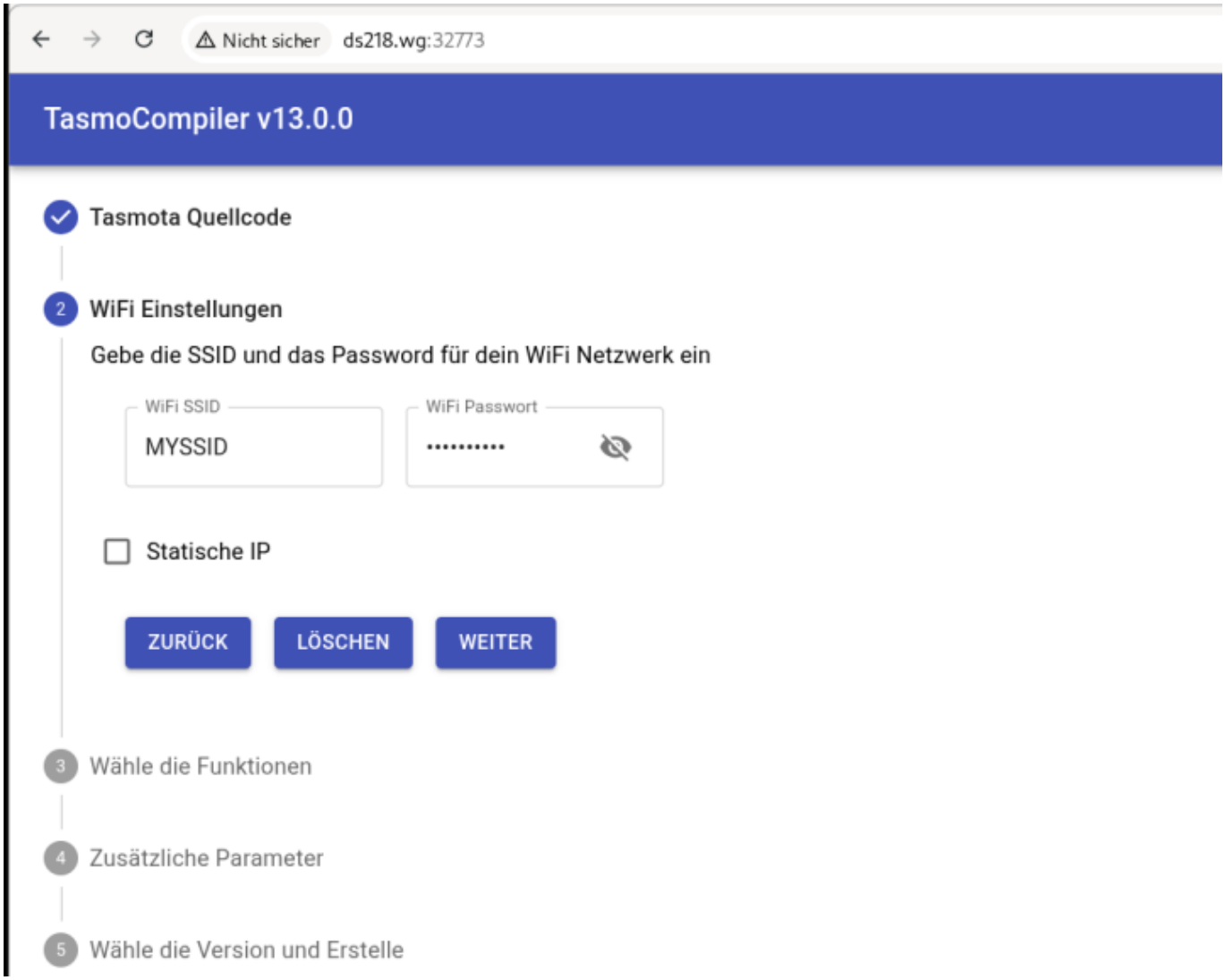
Home Brew Tasmota

Um eine eigene Version des Tasmota Firmware für einen ESP8266 oder ESP32 kompilieren zu könne benötigt man nicht nur den Tasmota Quellcode und alle notwendigen Bibliotheken sonder natürlich auch die `Tool chain`.
Das Aufsetzten der Entwicklungsumgebung inkl. Compiler ist nicht unbedingt eine 1-klick Installation daher habe ich mich für das All-In-One Paket `TasmoCompiler` entschieden.
Der TasmoCompiler kann am einfachsten als Docker Container verwendet werden. Wer z.B. eine Synology DiskStation sein Eigen nennt kann mit Hilde des `Container Manager` einfach den TasmoCompiler als Docker Image installieren.
Die folgenden Screenshots zeigen den sehr kurzen Weg zur Home-Brew Tasmota Firmware mit einem ganz eigenen Feature-Set:

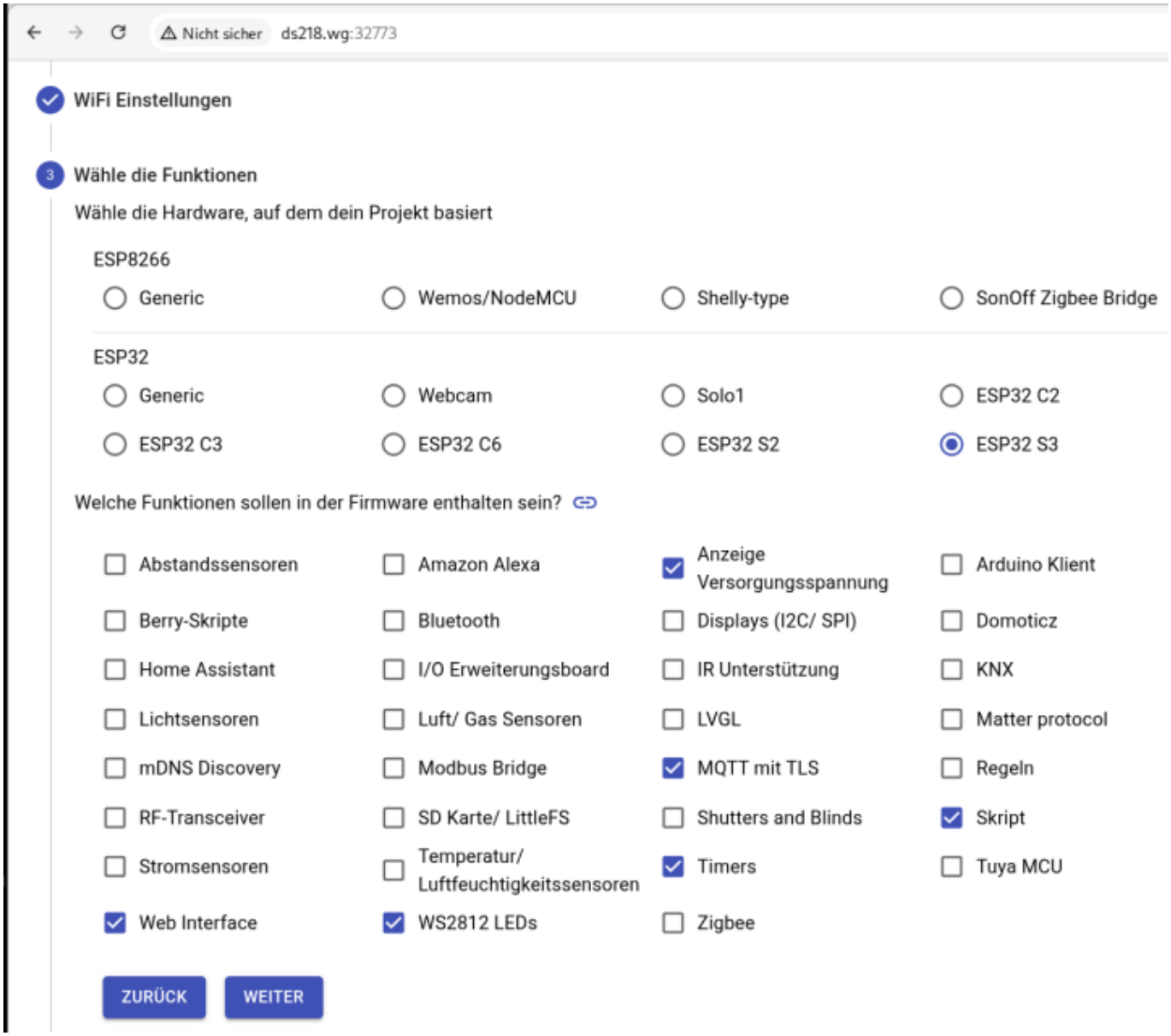
Tasmota Quellcode herunterladen oder vorhandenen Stand aktualisieren:



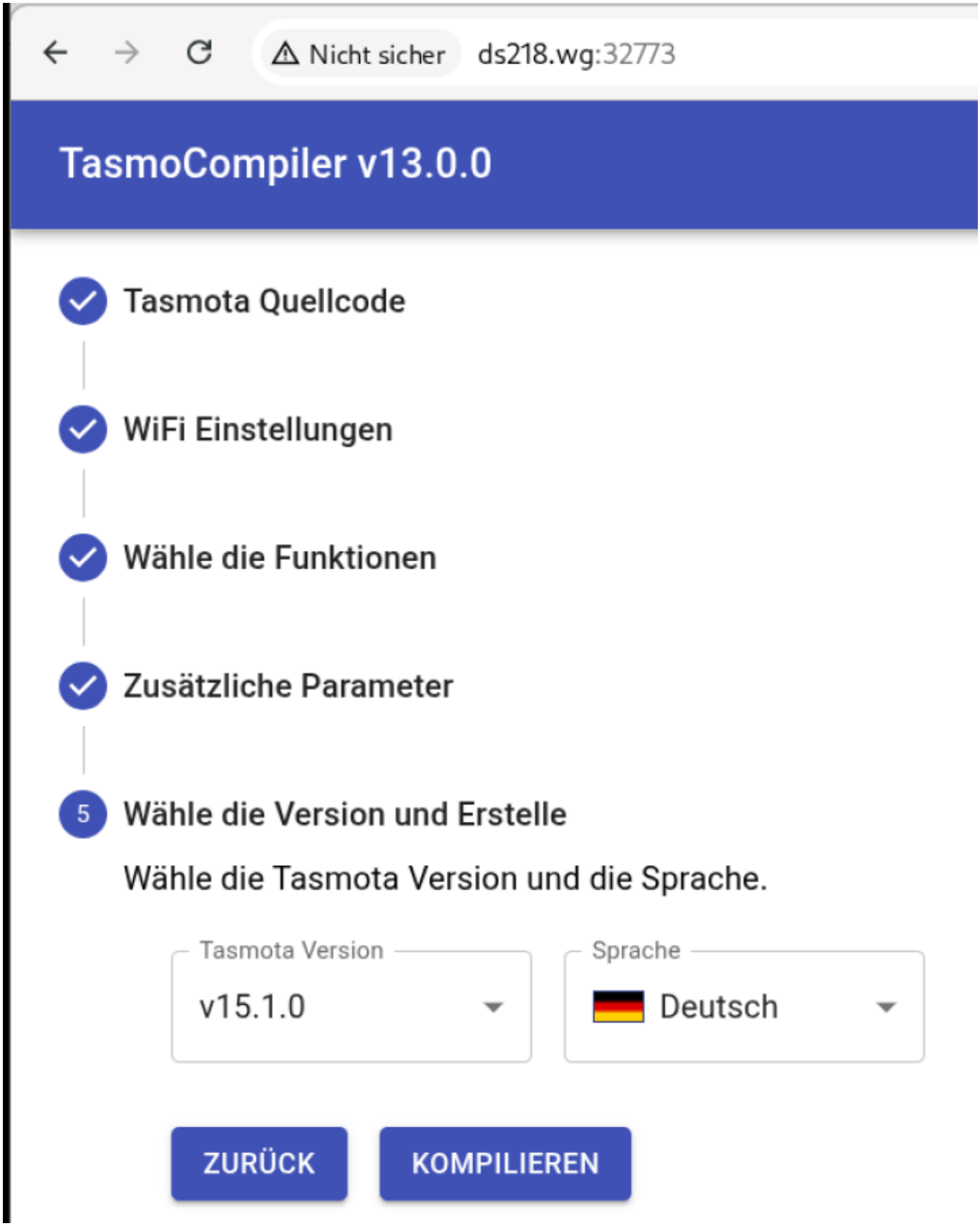
WLAN SSID und Schlüssel konfigurieren:



EPS Modell und benötigte Funktionen auswählen:

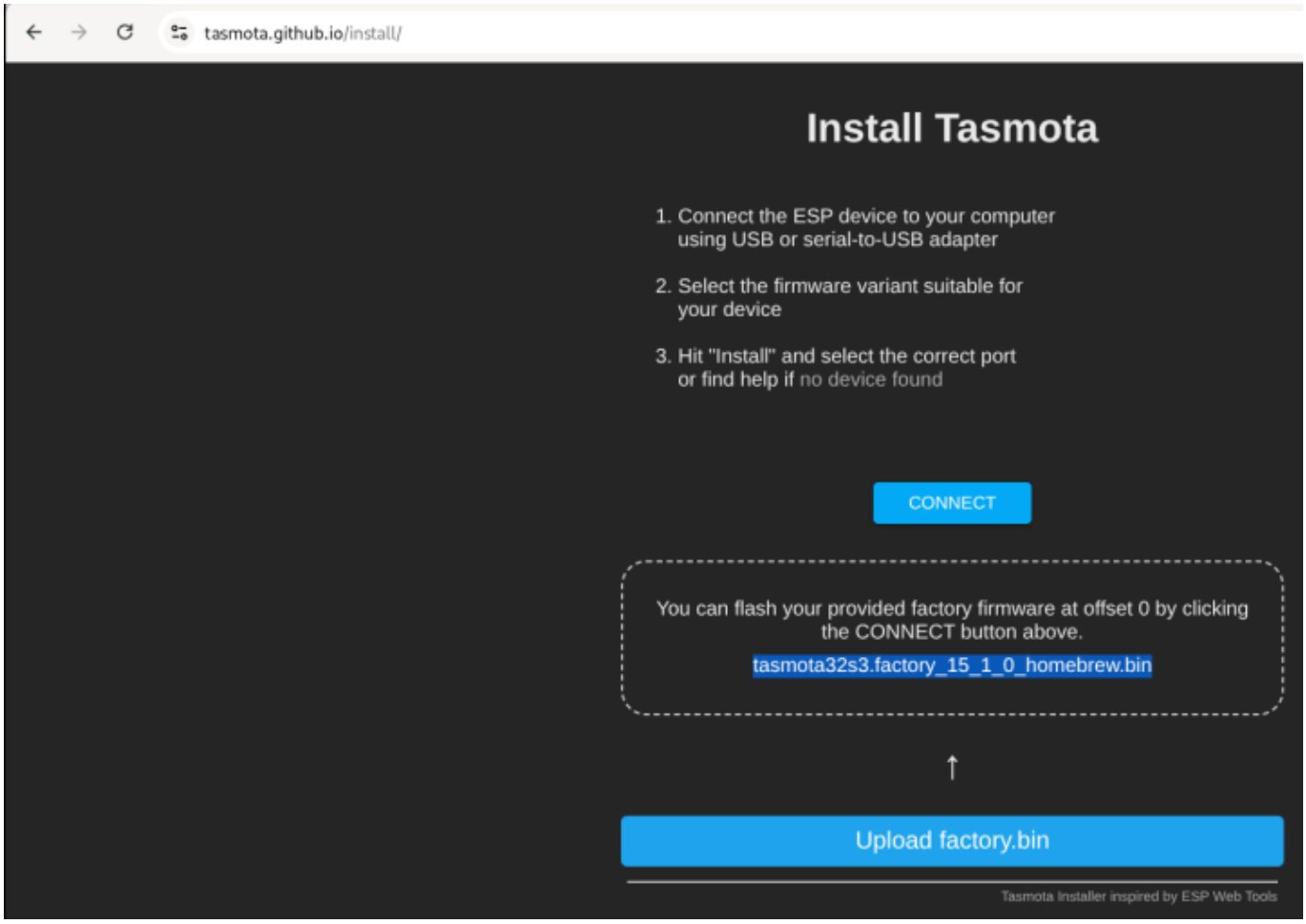


Tasmota Version und Sprache auswählen und Kompiliervorgang starten::



Nach erfolgreicher Kompilierung wird die erstellte Firmware zum Download angeboten:

Das fertige Firmware Image `tasmota32s3.factory_15_1_0_homebrew.bin` kann auf den ESP32 z.B. via Tasmota Web Installer programmiert werden.



Das Shelly Pro 3EM Emulator Script

EFR SGM-D4 Konfiguration

```
>D
>B
->sensor53 r
>M 1
+1,3,s,16,9600,SGM,1
1,770701000108O0ff@1000,Verbrauch,kWh,E_in,3
1,770701000208O0ff@1000,Einspeisung,kWh,E_out,3
1,770701001007O0ff@1,akt. Leistung,W,Power,0
1,=h----
1,770701006001O0ff@#,Server-ID,,ID,0
#
```

Script

```
>D 250
; this script emulates a shelly pro, with small modifications may also
emulate an ecotracker
```

```
; proven to work an marstek Venus, Jupiter and B2500
res=0
c1p=0
c2p=0
c3p=0
c1c=0
c2c=0
c3c=0
cpwr=0
str=""
tstr=""
cstr=""
mstr1=""
mstr2=""
mstr3=""
header=""
once=0

>B
=>sensor53 r
; if you modify this section you must restart tasmota

>ah
; http rpc handler
res=won(1 "/rpc/*")
; http status
res=won(2 "/status")
; http ecotacker status
res=won(3 "/v1/json")

>on1
;print here comes http rpc request
str=warg
res=ins(str "EM.GetStatus")
if res>=0 {
wcs so(4)
=#htph
wcs %mstr1%
=#getsrc
wcs %header%
=#getstat
wcs %mstr1%
wcs %mstr2%
wcf
break
}
res=ins(str "Shelly.GetDeviceInfo")
if res>=0 {
wcs so(4)
```

```
=#htph
wcs %mstr1%
=#getsrc
=#getdefi
wcs %header%
wcs %mstr1%
wcs %mstr2%
wcf
break
}
res=ins(str "EM.GetConfig")
if res>=0 {
wcs so(4)
=#htph
wcs %mstr1%
=#getsrc
=#getcfg
wcs %header%
wcs %mstr1%
wcf
break
}
res=ins(str "EMData.GetStatus")
if res>=0 {
wcs so(4)
=#htph
wcs %mstr1%
=#getsrc
=#egetstat
wcs %header%
wcs %mstr1%
wcf
break
}

print unknown http equest: %str%

>on2
;print here comes the status request
wcs so(4)
=#htph
wcs %mstr1%
dp(0 2)
wcs {"Power": %cpwr%,"E_in":%sml[1]%,"E_out":%sml[2]%}
wcf

>on3
;print here comes the v1/json for ecotracker
wcs so(4)
=#htph
wcs %mstr1%
```

```
dp(0 2)
wcs
{"energyCounterIn":%sml[1]%,"energyCounterOut":%sml[2]%,"powerAvg":%cpwr%,"e
nergyCounterInT1":0,
wcs "energyCounterInT2":0,"power":%cpwr%}
wcf

#htph
mstr1="HTTP/1.1 200 OK\r\nContent-type: application/json\r\n\r\n"

#getcfg
mstr1="{\"id\":0,\"name\":null,\"blink_mode_selector\":\"active_energy\",\"p
hase_selector\":\"a\",\"monitor_phase_sequence\":true,\"ct_type\":\"120A\"}}
"

#getdefi
mstr1="{\"name\":\""+tstr+"\",\"id\":\""+tstr+"\",\"mac\":\""+maca+"\",\"slo
t\":1,\"model\":\"SPEM-003CEBEU\","
mstr2="\"gen\":2,\"fw_id\":\"20241011-114455/1.4.4-g6d2a586\","
mstr2+="\"ver\":\"1.4.4\",\"app\":\"Pro3EM\",\"auth_en\":0,\"profile\":\"tri
phase\"}}"

#getstat
dp(0 2)
mstr1="{\"id\":0,\"a_current\":"+s(c1c)+",\"a_voltage\":230,\"a_act_power\":
"+s(c1p)+",\"a_aprt_power\":"+s(c1p)+",\"a_pf\":1,\"a_freq\":50,"
mstr1+="\"b_current\":"+s(c2c)+",\"b_voltage\":230,\"b_act_power\":"+s(c2p)+
",\"b_aprt_power\":"+s(c2p)+",\"b_pf\":1,\"b_freq\":50,"
mstr2="\"c_current\":"+s(c3c)+",\"c_voltage\":230,\"c_act_power\":"+s(c3p)+"
,\"c_aprt_power\":"+s(c3p)+",\"c_pf\":1,\"c_freq\":50,"
mstr2+="\"total_current\":"+s(c1c+c2c+c3c)+",\"total_act_power\":"+s(cpwr)+"
,\"total_aprt_power\":"+s(cpwr)+"}}"

#egetstat
dp(0 2)
mstr1="{\"id\":0,\"a_total_act_energy\":"+s(c1p)+",\"a_total_act_ret_energy\
":"+s(c1p)+",\"b_total_act_energy\":"+s(c2p)+",\"b_total_act_ret_energy\":"+
s(c2p)+",\"c_total_act_energy\":"+s(c3p)+",\"c_total_act_ret_energy\":"+s(c3
p)+",\"total_act\":"+s(cpwr)+",\"total_act_ret\":"+s(cpwr)+"}}"

#getsrc
tstr="shellypro3em-"+maca
header="{\"id\":0,\"src\":\""+tstr+"\",\"result\":"

>S
if year<2000 {
break
}
```

```
; adapt this to your meter
; update every 3 seconds
if upsecs%3==0 {
cpwr=sml[3]
c1p=sml[4]
c2p=sml[5]
c3p=sml[6]
}

; use this if you only have only one phase meter values
;c1p=cpwr/3
;c2p=cpwr/3
;c3p=cpwr/3

; calculate phase currents
c1c=c1p/230
c2c=c2p/230
c3c=c3p/230

if once==0 {
; start mdns for Shelly second parameter "-" means use device mac
res=mdns("shellypro3em-" "-" "shelly")
; start udp rpc handler on port 1010 or on port 2220 (for b2500)
res=udp(0 1010)
;res=udp(0 2220)
once=1
}

; evaluate udp input
str=udp(1)
if str!="" {
;print udp rpc payload=%str%
res=ins(str "EM.GetStatus")
if res>=0 {
=#getsrc
=#getstat
udp(2 header mstr1 mstr2)
;print >> %header%
;print >> %mstr1%
;print >> %mstr2%
break
}

res=ins(str "Shelly.GetDeviceInfo")
if res>=0 {
=#getsrc
=#getdefi
udp(2 header mstr1 mstr2)
;print >> 1 %mstr1%
;print >> 2 %mstr2%
break
```

```
}

res=ins(str "EM.GetConfig")
if res>=0 {
=#getsrc
=#getcfg
udp(2 header mstr1)
;print >> 1 %mstr1%
break
}

res=ins(str "EMData.GetStatus")
if res>=0 {
=#getsrc
=#egetstat
udp(2 header mstr1)
;print >> 1 %mstr1%
break
}
}

; adapt your own meter descriptor here
>M 1
+1,5,o,16,9600,eBZ,4
1,1-0:1.8.0*255(@1,Verbrauch,kWh,E_in,3
1,1-0:2.8.0*255(@1,Einspeisung,kWh,E_out,3
1,1-0:16.7.0*255(@1,akt. Leistung,W,Power,0
1,1-0:36.7.0*255(@1,Leistung L1,W,36_7_0,0
1,1-0:56.7.0*255(@1,Leistung L2,W,56_7_0,0
1,1-0:76.7.0*255(@1,Leistung L3,W,76_7_0,0
1,1-0:32.7.0*255(@1,Spannung L1,V,32_7_0,1
1,1-0:52.7.0*255(@1,Spannung L2,V,52_7_0,1
1,1-0:72.7.0*255(@1,Spannung L3,V,72_7_0,1
1,1-0:96.1.0*255(@#),Identifikation,,96_1_0,0
#
```

From:
https://www.von-thuelen.de/ - **Christophs DokuWiki**

Permanent link:
https://www.von-thuelen.de/doku.php/wiki/projekte/shellypro3em/uebersicht?rev=1761857814

Last update: **2025/10/30 20:56**