

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](#)
- [IR-Schreib-Lesekopf mit TTL-Interface](#)
- [Tasmota SML Dekoder](#)

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 Aufsetzen 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 Hilfe 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:

← → ↻ ⚠ Nicht sicher ds218.wg:32773

Tasmocompiler v13.0.0

- 1 Tasmota Quellcode**

Du kannst den Quellcode auf die neuste Version aktualisieren oder klick auf Weiter für den nächsten Schritt

QUELLCODE AKTUALISIEREN **WEITER**
- 2 WiFi Einstellungen
- 3 Wähle die Funktionen
- 4 Zusätzliche Parameter
- 5 Wähle die Version und Erstelle

WLAN SSID und Schlüssel konfigurieren:

← → ↻ ⚠ Nicht sicher ds218.wg:32773

Tasmocompiler v13.0.0

- ✓ Tasmota Quellcode
- 2 WiFi Einstellungen**

Gebe die SSID und das Passwort für dein WiFi Netzwerk ein

WiFi SSID: MYSSID

WiFi Passwort:

Statische IP

ZURÜCK LÖSCHEN WEITER
- 3 Wähle die Funktionen
- 4 Zusätzliche Parameter
- 5 Wähle die Version und Erstelle

EPS Modell und benötigte Funktionen auswählen:

← → ↻ Nicht sicher ds218.wg:32773

✓ WiFi Einstellungen

3 Wähle die Funktionen

Wähle die Hardware, auf dem dein Projekt basiert

ESP8266

Generic Wemos/NodeMCU Shelly-type SonOff Zigbee Bridge

ESP32

Generic Webcam Solo1 ESP32 C2

ESP32 C3 ESP32 C6 ESP32 S2 ESP32 S3

Welche Funktionen sollen in der Firmware enthalten sein? [↗](#)

<input type="checkbox"/> Abstandssensoren	<input type="checkbox"/> Amazon Alexa	<input checked="" type="checkbox"/> Anzeige Versorgungsspannung	<input type="checkbox"/> Arduino Klient
<input type="checkbox"/> Berry-Skripte	<input type="checkbox"/> Bluetooth	<input type="checkbox"/> Displays (I2C/ SPI)	<input type="checkbox"/> Domoticz
<input type="checkbox"/> Home Assistant	<input type="checkbox"/> I/O Erweiterungsboard	<input type="checkbox"/> IR Unterstützung	<input type="checkbox"/> KNX
<input type="checkbox"/> Lichtsensoren	<input type="checkbox"/> Luft/ Gas Sensoren	<input type="checkbox"/> LVGL	<input type="checkbox"/> Matter protocol
<input type="checkbox"/> mDNS Discovery	<input type="checkbox"/> Modbus Bridge	<input checked="" type="checkbox"/> MQTT mit TLS	<input type="checkbox"/> Regeln
<input type="checkbox"/> RF-Transceiver	<input type="checkbox"/> SD Karte/ LittleFS	<input type="checkbox"/> Shutters and Blinds	<input checked="" type="checkbox"/> Skript
<input type="checkbox"/> Stromsensoren	<input type="checkbox"/> Temperatur/ Luftfeuchtigkeitssensoren	<input checked="" type="checkbox"/> Timers	<input type="checkbox"/> Tuya MCU
<input checked="" type="checkbox"/> Web Interface	<input checked="" type="checkbox"/> WS2812 LEDs	<input type="checkbox"/> Zigbee	

ZURÜCK WEITER

Tasmota Version und Sprache auswählen und Kompilervorgang starten::

← → ↻ ⚠ Nicht sicher ds218.wg:32773

Tasmocompiler v13.0.0

- ✓ Tasmota Quellcode
- ✓ WiFi Einstellungen
- ✓ Wähle die Funktionen
- ✓ Zusätzliche Parameter
- 5 Wähle die Version und Erstelle

Wähle die Tasmota Version und die Sprache.

Tasmota Version

Sprache

ZURÜCK KOMPILIEREN

Nach erfolgreicher Kompilierung wird die erstellte Firmware zum Download angeboten:

5 Wähle die Version und Erstelle

Wähle die Tasmota Version und die Sprache.

Tasmota Version: Sprache:

ZURÜCK KOMPILIEREN

Fortschritt der Kompilierung

```
Environment      Status      Duration
-----
tasmota32s3      SUCCESS     00:16:48.336
===== 1 succeeded in 00:16:48.336 =====
Finished. Exit code: 0.
Welcome stranger!
Welcome stranger!
```

If Tasmocompiler is useful to You, please consider supporting the project:

Dateien die zum Erstellen der Firmware benutzt wurden und die Binary herunterladen:

FIRMWARE.BIN FIRMWARE.FACTORY.BIN PLATFORMIO_OVERRIDE.INI USER_CONFIG_OVERRIDE.H

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

Install Tasmota

1. Connect the ESP device to your computer using USB or serial-to-USB adapter
2. Select the firmware variant suitable for your device
3. Hit "Install" and select the correct port or find help if no device found

CONNECT

You can flash your provided factory firmware at offset 0 by clicking the CONNECT button above.

[tasmota32s3.factory_15_1_0_homebrew.bin](#)

↑

Upload factory.bin

Tasmota Installer inspired by ESP Web Tools

Der Lesekopf

Verbindung zwischen ESP32 und TTL Lesekopf:

Funktion	ESP32 Pin	Lesekopf Pin
VCC: 3,3 V	3V3	3V3
GND	GND	GND
Daten Senden	43	Tx
Daten Empfangen	44	Rx

Einfacher Lesetest

Um das Lesen der Zählerdaten zu testen kann das folgende Skript verwendet werden. Zu beachten ist unbedingt die korrekte Konfiguration der Tx und RX Pins am ESP32 S3 Mini.

```
>D
>B
->sensor53 r
>M 1
+1,44,s,16,9600,SGM,43
1,77070100010800ff@1000,Verbrauch,kWh,E_in,3
1,77070100020800ff@1000,Einspeisung,kWh,E_out,3
1,77070100100700ff@1,akt. Leistung,W,Power,0
1,=h- - -
1,77070100600100ff@#,Server-ID,,ID,0
#
```

The screenshot displays the Tasmota web interface for an ESP32S3. At the top, the device name 'ESP32S3' and 'Tasmota' are shown in large white text on a black background. Below this, energy statistics are listed in white text: 'SGM Verbrauch 3996.500 kWh', 'SGM Einspeisung 366.684 kWh', and 'SGM akt. Leistung 179 W'. A separator line '----' is present. Below the separator, the 'SGM Server-ID' is shown as '"0a0145465223047c3470"'. At the bottom of the interface, there are five large, rounded rectangular buttons: 'Configuration', 'Information', 'Firmware Upgrade', 'Tools', and 'Restart'. The first four buttons are blue, and the 'Restart' button is red.

Das Shelly Pro 3EM Emulator Script

EFR SGM-D4 Konfiguration

>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
; if you modify this section you must restart tasmota
=>sensor53 r
; set tele period to 1 second
tper=1
PowerDelta 110

>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%, "energyCounterInT1":0,
wcs "energyCounterInT2":0, "power":%cpwr%}
wcf

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

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

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

#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, \"b_current\": "+s(c2c)+", \"b_voltage\":230, \"b_act_power\": "+s(c2p)+", \"b_aprt_power\": "+s(c2p)+", \"b_pf\":1, \"b_freq\":50, \"c_current\": "+s(c3c)+", \"c_voltage\":230, \"c_act_power\": "+s(c3p)+", \"c_aprt_power\": "+s(c3p)+", \"c_pf\":1, \"c_freq\":50, \"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(c3p)+", \"total_act\": "+s(cpwr)+", \"total_act_ret\": "+s(cpwr)+"}"

#getsrc
tstr="shellypro3em-"+mac
header="{\"id\":0, \"src\": \"\"+tstr+\"\", \"result\": \">S
if upsecs>10
then

```

```
smlj|=1

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
;1
;1,1-0:96.1.0*255(@#),Identifikation,,96_1_0,0
;#

>M 1
+1,44,s,0,9600,SGM-D4,43
;<M>,<decoder>@<scale><offs>,<label>,<UoM>,<var>,<precision>
; <precision> number of decimal places. Add 16 to transmit the data
immediately.
;           Otherwise it is transmitted on TelePeriod only.
;1,77070100100700FF@1,akt. Leistung,W,Power_curr,0
1,77070100010800FF@1,Zählerstand,Wh,E_in,3

```

```
;1,77070100010800FF@1000,Zählerstand,kWh,Total_in,2
1,77070100020800FF@1,Einspeisung,Wh,E_out,3
;1,77070100020800FF@1000,Einspeisung,kWh,Total_out,2
1,77070100100700FF@1,akt. Leistung,W,Power,0
1,77070100240700FF@1,Leistung L1,W,36_7_0,3
1,77070100380700FF@1,Leistung L2,W,56_7_0,3
1,770701004C0700FF@1,Leistung L3,W,76_7_0,3
1,77070100200700FF@1,Spannung L1,V,32_7_0,1
1,77070100340700FF@1,Spannung L2,V,52_7_0,1
1,77070100480700FF@1,Spannung L3,V,72_7_0,1
;
;1,770701000E0700FF@1,Netzfrequenz,Hz,netzfrequenz,1
;1,77070100200700FF@1,Phasenspannung L1,V,momentanspannung_l1,1
;1,770701001F0700FF@1,Phasenstrom L1,A,momentanstrom_l1,2
;1,77070100240700FF@1000,Wirkleistung L1,kW,summenwirkleistung_l1,3
;1,77070100340700FF@1,Phasenspannung L2,V,momentanspannung_l2,1
;1,77070100330700FF@1,Phasenstrom L2,A,momentanstrom_l2,2
;1,77070100380700FF@1000,Wirkleistung L2,kW,summenwirkleistung_l2,3
;1,77070100480700FF@1,Phasenspannung L3,V,momentanspannung_l3,1
;1,77070100470700FF@1,Phasenstrom L3,A,momentanstrom_l3,2
;1,770701004C0700FF@1000,Wirkleistung L3,kW,summenwirkleistung_l3,3
#
```

From:

<http://www.xn--vonthlen-b6a.de/> - Christophs DokuWiki

Permanent link:

<http://www.xn--vonthlen-b6a.de/doku.php/wiki/projekte/shellypro3em/uebersicht>

Last update: **2026/01/14 20:02**

