3a. Allgemeine Software - ESP32

Wir verwenden auf dem ESP32 die Software ESPEasy, um die Sensor-Rohdaten auszulesen und per MQTT an den LoxBerry weiterzuleiten. Natürlich könnt ihr auch Tasmota oder ESPHome o. ä. verwenden. Ihr müsst dann aber darauf achten, dass die Daten exakt so im MQTT Broker eingeliefert werden, wie es das Weatherstation Plugin später erwartet! Auch die Zykluszeiten (also die Sendeintervalle) müssen identisch sein!

Dann gibt es noch einen Unterschied, ob ihr den Blitzsensor AS3935 verwenden wollt oder nicht. Wenn ihr den Blitzsensor benutzen wollt, müsst ihr die Firmware selbst kompilieren (oder unsere vorkompilierte, aber eventuell etwas ältere Firmware verwenden). Ohne Blitzsensor könnt ihr die aktuellen vorkompilierten Versionen von ESPEasy verwenden, was das Ganze erheblich vereinfacht. Aber auch das Selbstkompilieren ist keine Raketenwissenschaft mit etwas PC-Erfahrung - ich gebe Euch hier eine detaillierte Anleitung.

Firmware ohne Blitzsensor AS3935

Folgende Software benötigt ihr:

- ESPEasy Firmware: https://github.com/letscontrolit/ESPEasy/releases

Zunächst ladet ihr Euch das neueste Release der Firmware herunter. Dieses findet ihr unter dem obigen Link unter "Assets". Wählt die Firmware aus, die für euren ESP32 passt - normaler die Standard-ESP32 Firmware, also z.B. `ESPEasy_mega_20241222_ESP32_binaries.zip`

Entpackt die Firmwarefiles mit einem ZIP-Programm in ein beliebiges Verzeichnis. Das war's an dieser Stelle schon. Macht jetzt im Kapitel Firmware flashen weiter.

Firmware mit Blitzsensor AS3935

Leider gibt es keine vorkompilierte Firmware mit ESPEasy, die alle von uns genutzten Sensoren beinhaltet. Deswegen muss die Firmware von uns selbst kompiliert werden. Keine Angst - so schwer ist das nicht. Wer es dennoch nicht hinbekommt, kann sich auch hier die von uns bereits kompilierte Firmware herunterladen. Diese ist vermutlich nicht mehr ganz neu, das macht aber in aller Regel gar nichts. So funktioniert einwandfrei. Download hier:
esp_easy_mega_20241231_custom_esp32_4m316k.zip

Die Datei muss noch mit einem ZIP-Programm in ein beliebiges Verzeichnis entpackt werden. Macht dann im Kapitel Firmware flashen weiter.

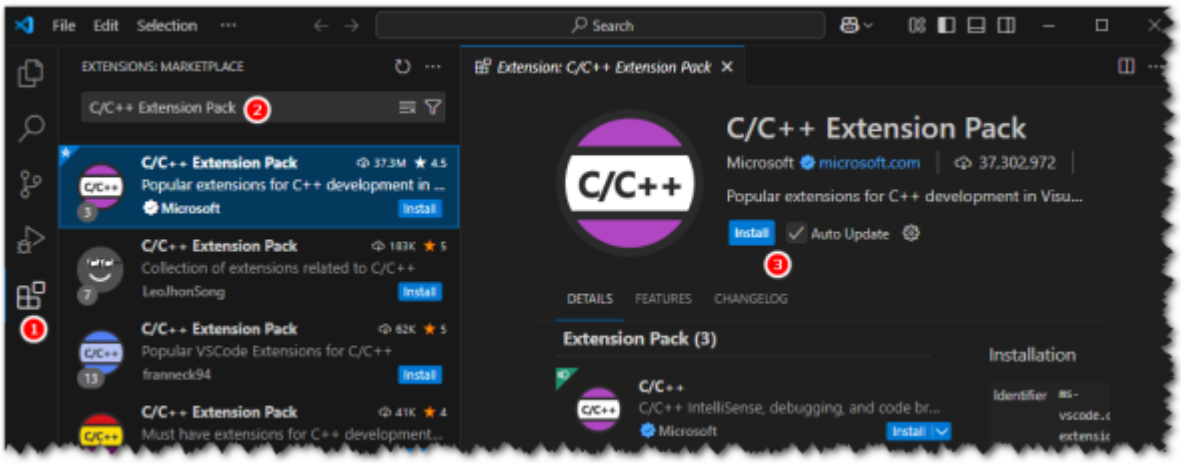
Wenn ihr die Firmware selbst kompilieren wollt, benötigt ihr folgende Software:

- ESPEasy Source Code: https://github.com/letscontrolit/ESPEasy/archive/refs/heads/mega.zip
- Microsoft Visual Code Studio: https://code.visualstudio.com/

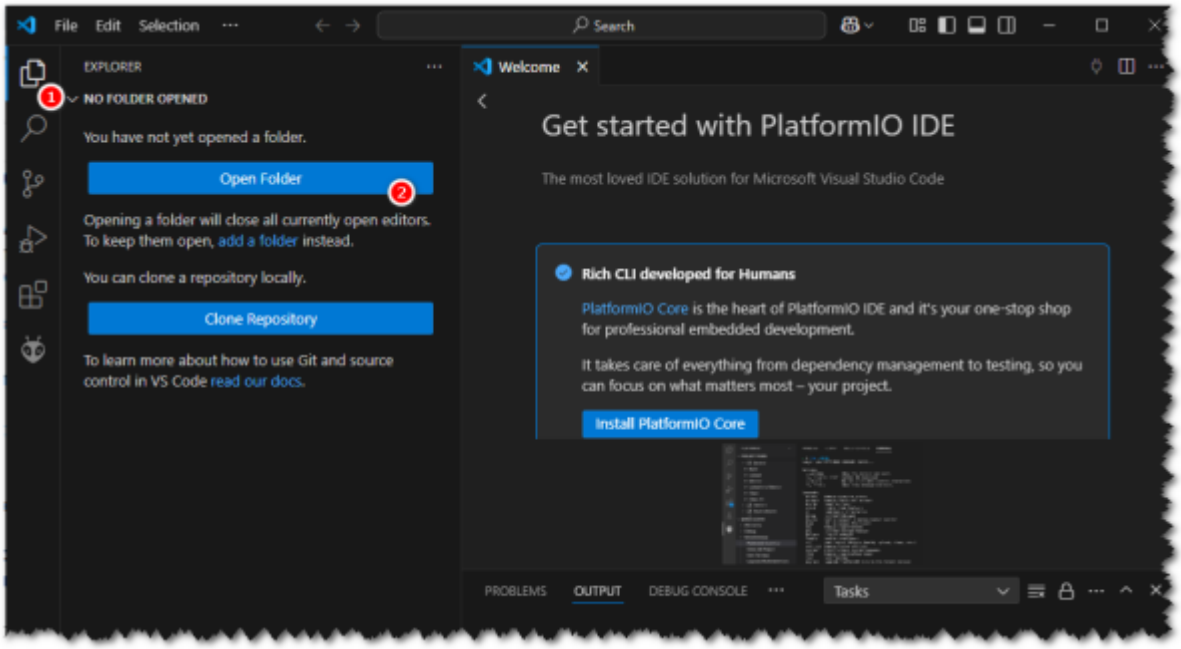
Als erstes ladet ihr Euch den Sourcecode vom ESPEasy Projekt über obigen Link herunter und entpackt diesen mit einem ZIP-Programm in ein beliebiges Verzeichnis.

Zunächst installiert ihr Microsoft Visual Code Studio auf eurem Rechner in startet es. Geht auf der linken Seite dann auf `Extensions` (der "kleine Würfel mit dem herausfliegenden Viertel" ganz unten). Nun sucht ihr der Reihe nach nach folgenden Extensions (über das Suchfeld) und installiert diese über den blauen `Install` Button:

- **C/C++ Extension Pack** (by Microsoft) - installiert automatisch auch die Extension **C/C++** mit
- **PlatformIO IDE** (by PlatformIO)



Die Installation dauert etwas - über ein kleines Fenster unten rechts könnt ihr jeweils den Status sehen. Nachdem die Installation durchgelaufen ist, müsst ihr VS Code einmal neu starten. Nun geht ihr in der rechten Leiste auf den `Explorer` und wählt den blauen Button `Open Folder`. Wechselt in das gerade eben entpackte Archiv mit dem Source Code. Ihr müsst das Verzeichnis auswählen, welches die Unterverzeichnisse `src`, `lib`, `boards` etc. enthält. Setzt den Haken bei `Trust the author of all files in the parrent folder…` und klickt auf `Yes, …` Nun wird der Sourcecode von PlatformIO vorbereitet. Das dauert eine ganze Weile. Der Status wird Euch wieder unten rechts angezeigt.



Das war's an dieser Stelle. Macht jetzt im Kapitel Firmware flashen weiter.

Firmware flashen

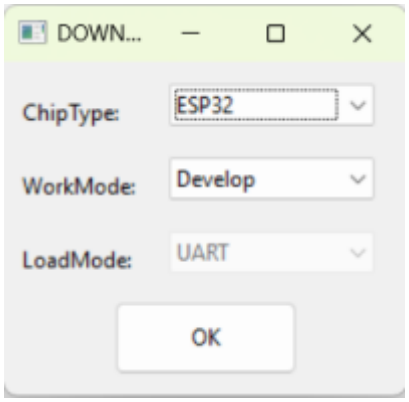
Folgende Software benötigt ihr:

- Flash Download Tool: https://docs.espressif.com/projects/esp-test-tools/en/latest/esp32/production_stage/tools/flash_download_tool.html
- PuTTY: https://www.putty.org/

Als erstes ladet ihr Euch noch die Flashsoftware "Flash Download Tool" herunter. Das Flash Download Tool wird auf euren Rechner mit einem ZIP-Programm entpackt. Jetzt verbindet ihr euren ESP32 per USB mit dem Rechner. Der ESP32 meldet sich als USB-to-UART Bridge CP210x oder FTDI am Rechner an. Sollte der Chip nicht erkannt werden, müsst ihr den entsprechenden Treiber noch installieren, siehe hier: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html

Überprüft nun, welcher COM-Port eurem Device zugeordnet wurde: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html#check-port-on-windows

Wenn wir das alles vorbereitet haben, starten wir die Flashsoftware "Flash Download Tool". Wählt im Startscreen euren ESP32 aus (wenn er nicht gelistet ist, wählt ESP32) und setzt die Software auf `Develop`:



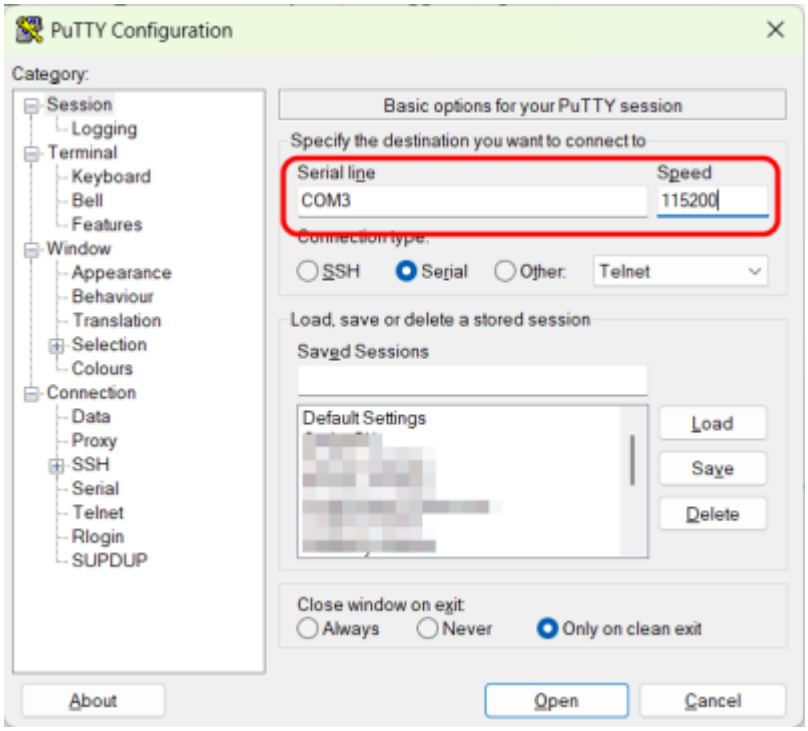
Oben wählt ihr als erstes das Firmware-File aus, welches ihr flashen wollt. Nutzt ihr die fertig kompilierten Firmwares des ESPEasy Projekts, geht ihr dazu in das `bin`-Verzeichnis der entpackten ESPEasy-Firmware und sucht Euch die `collection_G` heraus. Diese beinhaltet alle Plugins die wir benötigen (aber eben ohne den Blitzsensor AS3935). Ihr müsst das **Factory**-File passend für euren ESP32 auswählen, also z. B. `ESP_Easy_mega_20241222_collection_G_ESP32_4M316k.factory.bin`. Wenn ihr die selbst kompilierte Firmware nutzen wollt, dann findet ihr die gerade kompilierte Firmware im Verzeichnis `build_output\bin`. Auch hier müsst ihr das **Factory**-File auswählen.

Setzt vorne den Haken und gebt hinten als Startadresse hinter dem @-Zeichen `0x0` ein. Den SPI-Mode setzt ihr noch auf `DOUT` und wählt ganz unten den korrekten COM-Port aus, also z. B. `COM3`. Danach klickt ihr auf START und der Flashvorgang sollte starten. Er dauert ungefähr 2 Minuten.

Das war's. Macht jetzt im Kapitel Konfiguration weiter.

Konfiguration

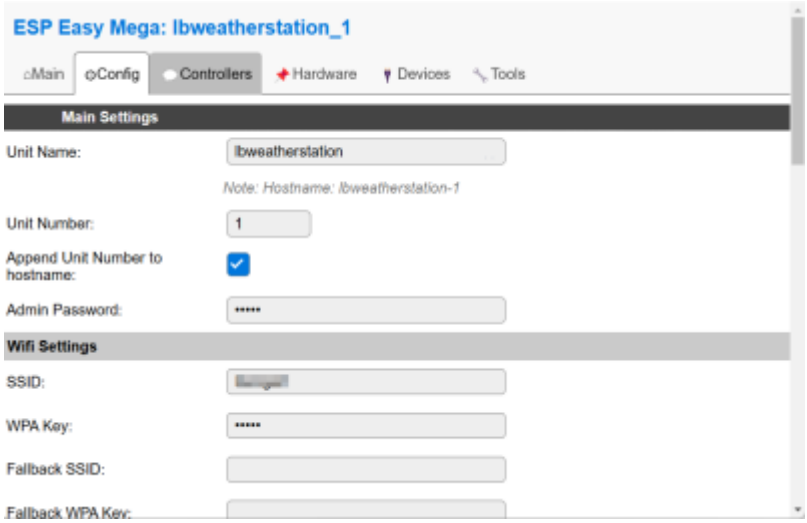
Zieht den ESP32 vom USB ab und installiert und startet dann PuTTY. PuTTY stellt ihr auf eine Serielle Verbindung ein und gebt den COM-Port an, der eurem ESP32 von Windows zugewiesen wurde. Die Geschwindigkeit stellt ihr auf `115200`.



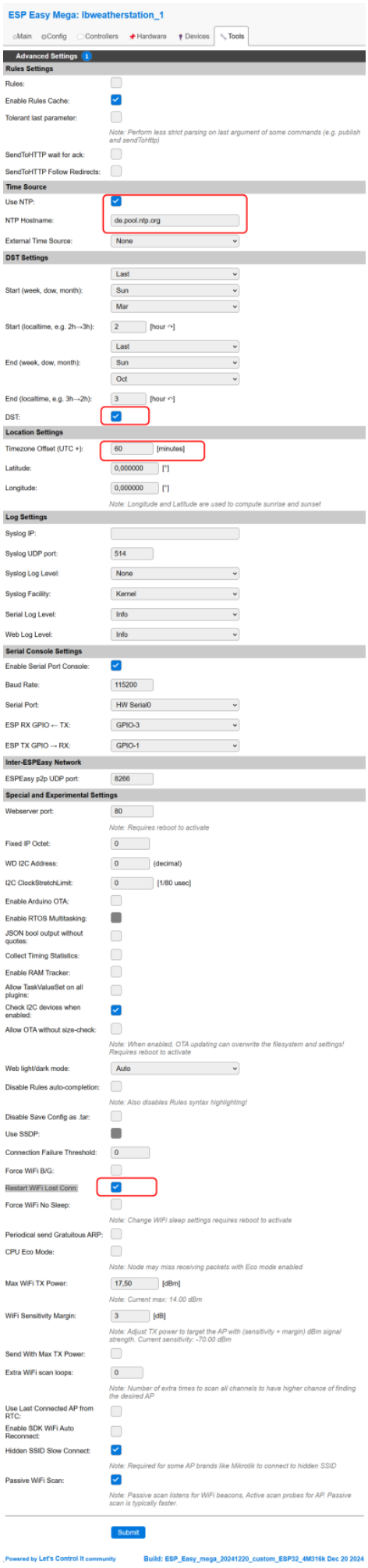
Nun verbindet ihr den ESP32 wieder mit dem Rechner, wartet 1-2 Sekunden, bis der ESP erkannt wurde und klickt in PuTTY dann auf `Open`. Ihr solltet nun in PuTTY die Bootmeldungen des ESP32 sehen und er sollte das WLAN zur Erstkonfiguration aufmachen - die Meldungen sollten in etwa so aussehen:

Der ESP macht wie üblich ein eigenes WLAN auf - der Name variiert je nach Firmware etwas. Er lautet `ESP_Easy`, `ESPEasy`, `ESPEasy_Collection_G`, `lbweatherstation` oder ähnlich. Das WLAN-Passwort ist `configesp`. Verbindet Euch mit dem WLAN und konfiguriert euer eigenes WLAN wie üblich. Anschließend startet der ESP neu und verbindet sich mit eurem eigenen WLAN. Wie immer findet ihr in eurem Router die IP-Adresse des neuen Gerätes heraus. Ruft die IP-Adresse in eurem Browser aus.

Unter `Config` vergebt ihr nun einen Unit-Namen (Hostnamen) und eine Unit-Nummer. Die Unit-Nummer ist beliebig, sollte aber eindeutig sein - sie dient der Kommunikation verschiedener ESPEasy Installationen untereinander. Wir nutzen dieses Feature allerdings nicht weiter. Vergebt ein Admin-Passwort für das Webinterface. Der Zugang ist dann durch den User `admin` mit passendem Kennwort geschützt. Den Rest lasst ihr hier auf den Standardeinstellungen.



Im Tab `Tools` öffnet ihr nun noch unter System die `Advanced` Options. Unter `Time Source` aktiviert ihr NTP und nutzt den NTP-Server `de.pool.ntp.org`. Prüft die `DST Settings` (Sommer-/Winterzeit) und die `Location Settings` (hier die ''Timezone Offset (UTC +)''). Unter `Special and Experimental Settings` aktiviert ihr noch ''Restart WiFi Lost Conn'', damit der ESP die WLAN-Verbindung automatisch neu aufbaut, wenn er sie verliert.


ESP Easy Mega: lbweatherstation_1
Main
Config
Controllers
Hardware
Devices
Tools
Advanced Settings
Rules Settings
Rules:
Enable Rules Cache:
Tolerant last parameter:
Note: Perform less strict parsing on last argument of some commands (e.g. publish and sendToHttp)
SendToHTTP wait for ack:
SendToHTTP Follow Redirects:
Time Source
Use NTP:
NTP Hostname:
de.pool.ntp.org
External Time Source:
None
DST Settings
Start (week, dow, month):
Last
Sun
Mar
Start (localtime, e.g. 2h→3h):
2
[hour ↷]
End (week, dow, month):
Last
Sun
Oct
End (localtime, e.g. 3h→2h):
3
[hour ↶]
DST:
Location Settings
Timezone Offset (UTC +):
60
[minutes]
Latitude:
0,000000
[°]
Longitude:
0,000000
[°]
Note: Longitude and Latitude are used to compute sunrise and sunset
Log Settings
Syslog IP:
Syslog UDP port:
514
Syslog Log Level:
None
Syslog Facility:
Kernel
Serial Log Level:
Info
Web Log Level:
Info
Serial Console Settings
Enable Serial Port Console:
Baud Rate:
115200
Serial Port:
HW Serial0
ESP RX GPIO ← TX:
GPIO-3
ESP TX GPIO → RX:
GPIO-1
Inter-ESPEasy Network
ESPEasy p2p UDP port:
8266
Special and Experimental Settings
Webserver port:
80
Note: Requires reboot to activate
Fixed IP Octet:
0
WD I2C Address:
0
(decimal)
I2C ClockStretchLimit:
0
[1/80 usec]
Enable Arduino OTA:
Enable RTOS Multitasking:
JSON bool output without quotes:
Collect Timing Statistics:
Enable RAM Tracker:
Allow TaskValueSet on all plugins:
Check I2C devices when enabled:
Allow OTA without size-check:
Note: When enabled, OTA updating can overwrite the filesystem and settings! Requires reboot to activate
Web light/dark mode:
Auto
Disable Rules auto-completion:
Note: Also disables Rules syntax highlighting!
Disable Save Config as .tar:
Use SSDP:
Connection Failure Threshold:
0
Force WiFi B/G:
Restart WiFi Lost Conn:
Force WiFi No Sleep:
Note: Change WiFi sleep settings requires reboot to activate
Periodical send Gratuitous ARP:
CPU Eco Mode:
Note: Node may miss receiving packets with Eco mode enabled
Max WiFi TX Power:
17,50
[dBm]
Note: Current max: 14.00 dBm
WiFi Sensitivity Margin:
3
[dB]
Note: Adjust TX power to target the AP with (sensitivity + margin) dBm signal strength. Current sensitivity: -70.00 dBm
Send With Max TX Power:
Extra WiFi scan loops:
0
Note: Number of extra times to scan all channels to have higher chance of finding the desired AP
Use Last Connected AP from RTC:
Enable SDK WiFi Auto Reconnect:
Hidden SSID Slow Connect:
Note: Required for some AP brands like Mikrotik to connect to hidden SSID
Passive WiFi Scan:
Note: Passive scan listens for WiFi beacons, Active scan probes for AP. Passive scan is typically faster.
Submit
Powered by Let's Control It community
Build: ESP_Easy_mega_20241220_custom_ESP32_4M316k Dec 20 2024

Im Tab `Controllers` fügen wir einen neuen Controller hinzu (bzw. editieren den bereits unter 1 angelegten Controller). Controller dienen der Kommunikation - wir richten hier unseren MQTT Broker ein. Als Protokoll wählt ihr `Home Assistent (openHAB) MQTT` aus und gebt die IP-Adresse und Port eures MQTT Brokers ein (normalerweise die IP-Adresse euer LoxBerry und Port 1883). Weiter unten unter `Credentials` aktiviert ihr ''Use Extended Credentials'' und gebt User und Passwort für euren MQTT Broker ein. Die notwendigen Einstellungen findet ihr im LoxBerry im MQTT Widget. Unter ''MQTT'' setzt ihr nun noch folgende Einstellungen:

- Controller Subscribe: `%sysname%/sensors/#`
- Controller Publish: `%sysname%/sensors/%tskname%/%valname%`
- Controller LWT Topic: `%sysname%/status`
- LWT Connect Message: `running`
- LWT Disconnect Message: `stopped`
- Send LWT to Broker: `Ja`
- Enabled: `Ja`

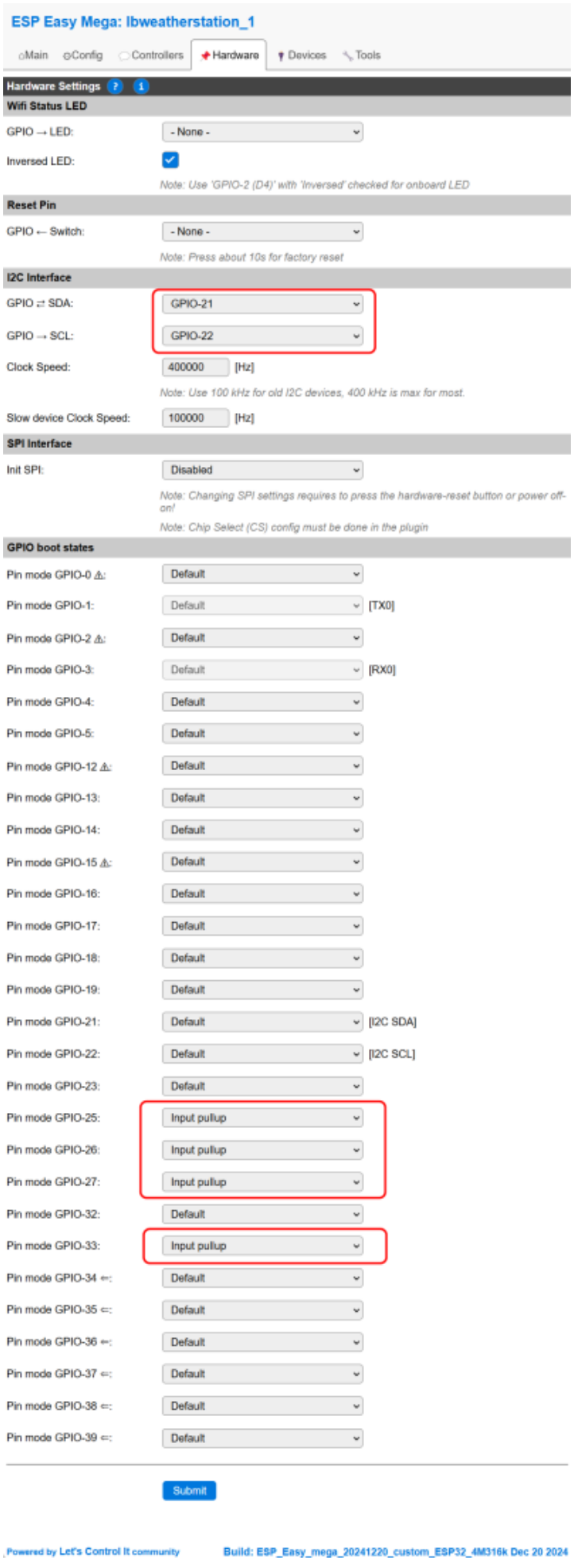
Anschließend sollte sich der ESP mit eurem MQTT Broker verbinden. Prüft den Status im `Config` Tab (gebt ihm ein paar Sekunden um sich zu verbinden).

Unter `Hardware` konfigurieren wir nun noch das I2C Interface und unsere GPIOs, die wir verwenden wollen. Zunächst konfigurieren wir unter I2C Interface die beiden GPIOs für SDA und DCL:

- GPIO SDA: `GPIO-21`
- GPIO SCL: `GPIO-22`

Danach setzen wir noch die GPIOs, die wir als digitale Eingänge verwenden wollen, als Eingang mit aktiviertem PullUp-Widerstand:

- GPIO-25: `Input pullup`
- GPIO-26: `Input pullup`
- GPIO-27: `Input pullup`
- GPIO-33: `Input pullup`


ESP Easy Mega: lbweatherstation_1
Main
Config
Controllers
Hardware
Devices
Tools
Hardware Settings
Wifi Status LED
GPIO → LED:
- None -
Inversed LED:
Note: Use 'GPIO-2 (D4)' with 'Inversed' checked for onboard LED
Reset Pin
GPIO ← Switch:
- None -
Note: Press about 10s for factory reset
I2C Interface
GPIO ⇄ SDA:
GPIO-21
GPIO → SCL:
GPIO-22
Clock Speed:
400000
[Hz]
Note: Use 100 kHz for old I2C devices, 400 kHz is max for most.
Slow device Clock Speed:
100000
[Hz]
SPI Interface
Init SPI:
Disabled
Note: Changing SPI settings requires to press the hardware-reset button or power off-on!
Note: Chip Select (CS) config must be done in the plugin
GPIO boot states
Pin mode GPIO-0 ⚠:
Default
Pin mode GPIO-1:
Default
[TX0]
Pin mode GPIO-2 ⚠:
Default
Pin mode GPIO-3:
Default
[RX0]
Pin mode GPIO-4:
Default
Pin mode GPIO-5:
Default
Pin mode GPIO-12 ⚠:
Default
Pin mode GPIO-13:
Default
Pin mode GPIO-14:
Default
Pin mode GPIO-15 ⚠:
Default
Pin mode GPIO-16:
Default
Pin mode GPIO-17:
Default
Pin mode GPIO-18:
Default
Pin mode GPIO-19:
Default
Pin mode GPIO-21:
Default
[I2C SDA]
Pin mode GPIO-22:
Default
[I2C SCL]
Pin mode GPIO-23:
Default
Pin mode GPIO-25:
Input pullup
Pin mode GPIO-26:
Input pullup
Pin mode GPIO-27:
Input pullup
Pin mode GPIO-32:
Default
Pin mode GPIO-33:
Input pullup
Pin mode GPIO-34 ⇐:
Default
Pin mode GPIO-35 ⇐:
Default
Pin mode GPIO-36 ⇐:
Default
Pin mode GPIO-37 ⇐:
Default
Pin mode GPIO-38 ⇐:
Default
Pin mode GPIO-39 ⇐:
Default
Submit
Powered by Let's Control It community
Build: ESP_Easy_mega_20241220_custom_ESP32_4M316k Dec 20 2024

Damit ist die Grundkonfiguration abgeschlossen! Die einzelnen Eingänge und Sensoren folgenden dann in den jeweiligen Unterkapiteln.