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/CExtension Pack** (by Microsoft) - installiert automatisch auch die Extension **C/C++** mit * **PlatformIO IDE** (by PlatformIO) {{howtos_knowledge_base:loxberry_wetterstation:3_software:pasted:20241231-133803.png?600}} 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. {{howtos_knowledge_base:loxberry_wetterstation:3_software:pasted:20241231-134847.png?600}} Nun bereiten wir den Sourcecode für die spezifischen Einstellungen vor, die wir in der Firmware benötigen (u. a. wird hier festgelegt, welche Plugins und damit Sensoren in der Firmware zur Verfügung stehen). Dazu gehen wir links auf den Explorer und navigieren dann zur Datei ''src\Custom-Sample.h''. Kopiert die Datei (''Strg + C'') und fügt sie anschließend an gleicher Stelle noch einmal ein (''Strg + V''). Benennt die neue Datei per rechter Maustaste um in Custom.h (''Rename''). Löscht den gesamten Inhalt in der Datei (''Strg + a'' und ''Entf'') und ersetzt ihn durch den Inhalt der folgenden Datei: <code c Custom.h> #ifndef ESPEASY_CUSTOM_H #define ESPEASY_CUSTOM_H /* To modify the stock configuration without changing the EspEasy.ino file : 1) rename this file to "Custom.h" (It is ignored by Git) 2) define your own settings below 3) define USE_CUSTOM_H as a build flags. ie : export PLATFORMIO_BUILD_FLAGS="'-DUSE_CUSTOM_H'" */ /* ## Your Own Default Settings ## You can basically ovveride ALL macro defined in ESPEasy.ino. Don't forget to first #undef each existing #define that you add below. But since this Custom.h is included before other defines are made, you don't have to undef a lot of defines. Here are some examples: */ // --- Feature Flagging -- // Can be set to 1 to enable, 0 to disable, or not set to use the default (usually via define_plugin_sets.h) #define FEATURE_RULES_EASY_COLOR_CODE 1 // Use code highlighting, autocompletion and command suggestions in Rules #define FEATURE_ESPEASY_P2P 1 // (1/0) enables the ESP Easy P2P protocol #define FEATURE_ARDUINO_OTA 1 // enables the Arduino OTA capabilities #define FEATURE_THINGSPEAK_EVENT 1 // generate an event when requesting last value of a field in thingspeak via SendToHTTP(e.g. sendToHTTP,api.thingspeak.com,80,/channels/1667332/fields/5/last) // #define FEATURE_SD 1 // Enable SD card support // #define FEATURE_DOWNLOAD 1 // Enable downloading a file from an url #ifdef BUILD_GIT # undef BUILD_GIT #endif // ifdef

BUILD_GIT #define BUILD_GIT "My Build: " __DATE__ " " __TIME__ #define DEFAULT_NAME "lbweatherstation" // Enter your device friendly name #define UNIT 1 // Unit Number #define DEFAULT_DELAY 60 // Sleep Delay in seconds // --- Wifi AP Mode (when your Wifi Network is not reachable) ---------- #define DEFAULT_AP_IP 192, 168, 4, 1 // Enter IP address (comma separated) for AP (config) mode #define DEFAULT_AP_SUBNET 255, 255, 255, 0 // Enter IP address (comma separated) for AP (config) mode #define DEFAULT_AP_KEY "configesp" // Enter network WPA key for AP (config) mode // --- Wifi Client Mode ---------- ---------- #define DEFAULT_SSID "" // Enter your network SSID #define DEFAULT_KEY "" // Enter your network WPA key #define DEFAULT_SSID2 "" // Enter your fallback network SSID #define DEFAULT_KEY2 "" // Enter your fallback network WPA key #define DEFAULT_WIFI_INCLUDE_HIDDEN_SSID false // Allow to connect to hidden SSID APs #define DEFAULT_USE_STATIC_IP false // (true false) enabled or disabled static IP #define DEFAULT_IP "192.168.0.50" ***Enter your IP address #define DEFAULT_DNS*** **"*192.168.0.1*" Enter your DNS #define DEFAULT_GW "192.168.0.1"** ***Enter your Gateway #define DEFAULT_SUBNET*** **"*255.255.255.0*" Enter your Subnet #define DEFAULT_IPRANGE_LOW "0.0.0.0"** ***Allowed IP range to access webserver #define DEFAULT_IPRANGE_HIGH*** **"*255.255.255.255*" Allowed IP range to access webserver #define DEFAULT_IP_BLOCK_LEVEL 1** ***0: ALL_ALLOWED 1: LOCAL_SUBNET_ALLOWED 2:*** **ONLY_IP_RANGE_ALLOWED #define DEFAULT_ADMIN_USERNAME "admin" #define DEFAULT_ADMIN_PASS "" #define DEFAULT_WIFI_CONNECTION_TIMEOUT 10000** ***minimum timeout in ms for WiFi to be connected. #define DEFAULT_WIFI_FORCE_BG_MODE false*** **when set, only allow to connect in 802.11B or G mode (not N) #define DEFAULT_WIFI_RESTART_WIFI_CONN_LOST true** ***Perform wifi off and on when connection was lost. #define DEFAULT_ECO_MODE false*** **When set, make idle calls between executing tasks. #define DEFAULT_WIFI_NONE_SLEEP false** ***When set, the wifi will be set to no longer sleep (more power*** **used and need reboot to reset mode) #define DEFAULT_GRATUITOUS_ARP false** ***When set, the node will send periodical gratuitous ARP*** **packets to announce itself. #define DEFAULT_TOLERANT_LAST_ARG_PARSE false** ***When set, the last argument of some commands will be parsed to the end of the line*** **See: https://github.com/letscontrolit/ESPEasy/issues/2724 #define DEFAULT_SEND_TO_HTTP_ACK false** ***Wait for ack with SendToHttp command. #define DEFAULT_AP_DONT_FORCE_SETUP false*** **Allow optional usage of Sensor without WIFI avaiable** ***When set you can use the Sensor in AP-Mode without beeing forced to /setup #define DEFAULT_DONT_ALLOW_START_AP false*** **Usually the AP will be started when no WiFi is defined, or the defined one cannot be found. This flag may prevent it.** ***— Default Controller ———————————————————————— #define DEFAULT_CONTROLLER true*** **true or false enabled or disabled, set 1st controller**

```
// defaults
```

#define DEFAULT_CONTROLLER_ENABLED true *Enable default controller by default #define DEFAULT_CONTROLLER_USER* "" Default controller user #define DEFAULT_CONTROLLER_PASS "" *Default controller Password* using a default template, you also need to set a DEFAULT PROTOCOL to a suitable MQTT protocol ! #define DEFAULT_PUB "%sysname%/sensors/%tskname%/%valname%" *Enter your pub #define DEFAULT_SUB* "*%sysname%/sensors/#*" Enter your sub #define DEFAULT_SERVER "" *Enter your Server IP address #define DEFAULT_SERVER_HOST* "" Server hostname #define DEFAULT_SERVER_USEDNS false *true: Use hostname. false: use IP #define*

DEFAULT_USE_EXTD_CONTROLLER_CREDENTIALS true true: Allow longer user credentials for controllers #define DEFAULT_PORT 1883 *Enter your Server port value #define DEFAULT_CONTROLLER_TIMEOUT 100* Default timeout in msec #define DEFAULT_PROTOCOL 5 *Protocol used for controller communications* 0 = Stand-alone (no controller set)

```
//   1 = Domoticz HTTP
//   2 = Domoticz MQTT
//   3 = Nodo Telnet
//   4 = ThingSpeak
//   5 = Home Assistant (openHAB) MQTT
//   6 = PiDome MQTT
//   7 = EmonCMS
//   8 = Generic HTTP
//   9 = FHEM HTTP
```

#ifdef ESP8266 #define DEFAULT_PIN_I2C_SDA 4 #endif #ifdef ESP32 #define DEFAULT_PIN_I2C_SDA 21 *D21 #endif #ifdef ESP8266 #define DEFAULT_PIN_I2C_SCL 5 #endif #ifdef ESP32 #define DEFAULT_PIN_I2C_SCL 22* D22 #endif #define DEFAULT_I2C_CLOCK_SPEED 400000 *Use 100 kHz if working with old I2C chips #define FEATURE_I2C_DEVICE_SCAN 1 #define DEFAULT_SPI 0* 0=disabled 1=enabled and for ESP32 there is option 2 =HSPI #define DEFAULT_PIN_STATUS_LED (-1) #define DEFAULT_PIN_STATUS_LED_INVERSED true #define DEFAULT_PIN_RESET_BUTTON (-1) #define DEFAULT_USE_RULES false *(true|false) Enable Rules? #define DEFAULT_RULES_OLDENGINE true #define DEFAULT_MQTT_RETAIN false* (true|false) Retain MQTT messages? #define DEFAULT_CONTROLLER_DELETE_OLDEST false *(true|false) to delete oldest message when queue is full #define DEFAULT_CONTROLLER_MUST_CHECK_REPLY false* (true|false) Check Acknowledgment #define DEFAULT_MQTT_DELAY 100 *Time in milliseconds to retain MQTT messages #define DEFAULT_MQTT_LWT_TOPIC* "*%sysname%/status*" Default LWT Topic #define DEFAULT_MQTT_LWT_CONNECT_MESSAGE "running" *Default LWT messages if connected #define DEFAULT_MQTT_LWT_DISCONNECT_MESSAGE* "*stopped*" Default LWT messages if disconnected #define DEFAULT_MQTT_USE_UNITNAME_AS_CLIENTID 0 #define DEFAULT_USE_NTP true *(true|false) Use NTP Server #define DEFAULT_NTP_HOST* "*de.pool.ntp.org*" NTP Server Hostname #define DEFAULT_TIME_ZONE 60 *Time Offset (in minutes) #define DEFAULT_USE_DST true* (true|false) Use Daily Time Saving #define DEFAULT_LATITUDE 0.0f *Default Latitude #define DEFAULT_LONGITUDE 0.0f* Default Longitude #define DEFAULT_SYSLOG_IP "" *Syslog IP Address #define DEFAULT_SYSLOG_PORT 0* Standard syslog port: 514 #define DEFAULT_SYSLOG_FACILITY 0 *kern #define DEFAULT_SYSLOG_LEVEL 0* Syslog Log Level #define DEFAULT_SERIAL_LOG_LEVEL LOG_LEVEL_INFO *Serial Log Level #define DEFAULT_WEB_LOG_LEVEL LOG_LEVEL_INFO* Web Log Level #define DEFAULT_SD_LOG_LEVEL 0 *SD Card Log Level #define DEFAULT_USE_SD_LOG false* (true|false) Enable Logging to the SD card #define DEFAULT_USE_SERIAL true *(true|false) Enable Logging to the Serial Port #define DEFAULT_SERIAL_BAUD 115200* Serial Port Baud Rate #define DEFAULT_SYNC_UDP_PORT 8266 *Used for ESPEasy p2p. (IANA registered port: 8266)* Factory Reset defaults #define DEFAULT_FACTORY_RESET_KEEP_UNIT_NAME true #define DEFAULT_FACTORY_RESET_KEEP_WIFI true #define DEFAULT_FACTORY_RESET_KEEP_NETWORK true #define DEFAULT_FACTORY_RESET_KEEP_NTP_DST true #define DEFAULT_FACTORY_RESET_KEEP_CONSOLE_LOG true *#define BUILD_NO_DEBUG* Custom built-in url for hosting JavaScript and CSS files. #define CUSTOM_BUILD_CDN_URL "https://cdn.jsdelivr.net/gh/letscontrolit/ESPEasy@mega/static/" *Special SSID/key setup only to be used in custom builds.* Deployment SSID will be used only when the configured SSIDs are not

reachable and/or no credentials are set. *This to make deployment of large number of nodes easier #define CUSTOM_DEPLOYMENT_SSID* "" Enter SSID not shown in UI, to be used on custom builds to ease deployment #define CUSTOM_DEPLOYMENT_KEY "" *Enter key not shown in UI, to be used on custom builds to ease deployment #define CUSTOM_SUPPORT_SSID* "" Enter SSID not shown in UI, to be used on custom builds to ease support #define CUSTOM_SUPPORT_KEY "" *Enter key not shown in UI, to be used on custom builds to ease support* Emergency fallback SSID will only be attempted in the first 10 minutes after reboot. *When found, the unit will connect to it and depending on the built in flag, it will either just connect to it, or clear set credentials.* Use case: User connects to a public AP which does need to agree on an agreement page for the rules of conduct (e.g. open APs) *This is seen as a valid connection, so the unit will not reconnect to another node and thus becomes inaccessible. #define CUSTOM_EMERGENCY_FALLBACK_SSID* "" Enter SSID not shown in UI, to be used to regain access to the node #define CUSTOM_EMERGENCY_FALLBACK_KEY "" *Enter key not shown in UI, to be used to regain access to the node #define CUSTOM_EMERGENCY_FALLBACK_RESET_CREDENTIALS false #define CUSTOM_EMERGENCY_FALLBACK_START_AP false #define CUSTOM_EMERGENCY_FALLBACK_ALLOW_MINUTES_UPTIME 10* Allow for remote provisioning of a node. *This is only allowed for custom builds.* To setup the configuration of the provisioning file, one must also define FEATURE_SETTINGS_ARCHIVE *Default setting is to not allow to configure a node remotely, unless explicitly enabled.* #define FEATURE_CUSTOM_PROVISIONING 1 #define FEATURE_SSDP 1 #define FEATURE_EXT_RTC 1 *Support for external RTC clock modules like PCF8563/PCF8523/DS3231/DS1307 #define FEATURE_PLUGIN_STATS 1* Support collecting historic data + computing stats on historic data #ifdef ESP8266 # define PLUGIN_STATS_NR_ELEMENTS 16 #endif *ifdef ESP8266 # ifdef ESP32 # define PLUGIN_STATS_NR_ELEMENTS 64 #endif* ifdef ESP32 #define FEATURE_CHART_JS 1 *Support for drawing charts, like PluginStats historic data* Optional alternative CDN links: *Chart.js: (only used when FEATURE_CHART_JS is enabled)* #define CDN_URL_CHART_JS "https://cdn.jsdelivr.net/npm/chart.js@4.1.2/dist/chart.umd.min.js" *JQuery:* #define CDN_URL_JQUERY "https://code.jquery.com/jquery-3.6.0.min.js" *#define FEATURE_SETTINGS_ARCHIVE 1* #define FEATURE_I2CMULTIPLEXER 1 *#define FEATURE_TRIGONOMETRIC_FUNCTIONS_RULES 1* #define PLUGIN_USES_ADAFRUITGFX *Used by Display plugins using Adafruit GFX library* #define ADAGFX_ARGUMENT_VALIDATION 0 *Disable argument validation in AdafruitGFX_helper* #define ADAGFX_SUPPORT_7COLOR 0 *Disable the support of 7-color eInk displays by AdafruitGFX_helper #define FEATURE_SEND_TO_HTTP 1* Enable availability of the SendToHTTP command #define FEATURE_POST_TO_HTTP 1 *Enable availability of the PostToHTTP command #define FEATURE_PUT_TO_HTTP 1* Enable availability of the PutToHTTP command *#define FEATURE_I2C_DEVICE_CHECK 0* Disable the I2C Device check feature *#define FEATURE_I2C_GET_ADDRESS 0* Disable fetching the I2C address from I2C plugins. Will be enabled when FEATURE_I2C_DEVICE_CHECK is enabled *#define FEATURE_RTTTL 1* Enable rtttl command *#define FEATURE_ANYRTTTL_LIB 1* Use AnyRttl library for RTTTL handling *#define FEATURE_ANYRTTTL_ASYNC 1* When AnyRttl enabled, use Async (nonblocking) mode instead of the default Blocking mode *#define FEATURE_RTTTL_EVENTS 1* Enable RTTTL events for Async use, for blocking it doesn't make sense #if FEATURE_CUSTOM_PROVISIONING *For device models, see src/src/DataTypes/DeviceModel.h* #ifdef ESP32 *#define DEFAULT_FACTORY_DEFAULT_DEVICE_MODEL 0* DeviceModel_default *#endif* #ifdef ESP8266 *#define DEFAULT_FACTORY_DEFAULT_DEVICE_MODEL 0* DeviceModel_default *#endif* #define DEFAULT_PROVISIONING_FETCH_RULES1 false *#define DEFAULT_PROVISIONING_FETCH_RULES2 false* #define DEFAULT_PROVISIONING_FETCH_RULES3 false *#define DEFAULT_PROVISIONING_FETCH_RULES4 false* #define DEFAULT_PROVISIONING_FETCH_NOTIFICATIONS false *#define DEFAULT_PROVISIONING_FETCH_SECURITY false* #define

DEFAULT_PROVISIONING_FETCH_CONFIG false *#define DEFAULT_PROVISIONING_FETCH_PROVISIONING false* #define DEFAULT_PROVISIONING_FETCH_FIRMWARE false *#define DEFAULT_PROVISIONING_SAVE_URL false* #define DEFAULT_PROVISIONING_SAVE_CREDENTIALS false *#define DEFAULT_PROVISIONING_ALLOW_FETCH_COMMAND false* #define DEFAULT_PROVISIONING_URL "" *#define DEFAULT_PROVISIONING_USER* "" #define DEFAULT_PROVISIONING_PASS "" #endif /* ## ##*

```
Defining web interface
```

*## ## */* #define MENU_INDEX_MAIN_VISIBLE true /* #define MENU_INDEX_CONFIG_VISIBLE false #define MENU_INDEX_CONTROLLERS_VISIBLE false #define MENU_INDEX_HARDWARE_VISIBLE false #define MENU_INDEX_DEVICES_VISIBLE false #define MENU_INDEX_RULES_VISIBLE false #define MENU_INDEX_NOTIFICATIONS_VISIBLE false #define MENU_INDEX_TOOLS_VISIBLE false */ #define MAIN_PAGE_SHOW_SYSINFO_BUTTON true #define MAIN_PAGE_SHOW_WiFi_SETUP_BUTTON true #define MAIN_PAGE_SHOW_BASIC_INFO_NOT_LOGGED_IN false #define MAIN_PAGE_SHOW_NODE_LIST_BUILD true #define MAIN_PAGE_SHOW_NODE_LIST_TYPE true #define SETUP_PAGE_SHOW_CONFIG_BUTTON true *#define FEATURE_AUTO_DARK_MODE 0* 0 = Disable auto-dark mode *#define FEATURE_EXTENDED_TASK_VALUE_TYPES 0* 0 = Disable extra task value types like 64 bit ints, double, etc. in Dummy tasks *#define FEATURE_USE_DOUBLE_AS_ESPEASY_RULES_FLOAT_TYPE 0* 0 = switch to float as floating point type for rules/formula processing. *#define WEBPAGE_TEMPLATE_HIDE_HELP_BUTTON #define SHOW_SYSINFO_JSON 1* Enables the sysinfo_json page (by default is enabled when WEBSERVER_NEW_UI is enabled too) /* ## ##*

```
CSS / template
```

*## ## */* /* #define WEBPAGE_TEMPLATE_DEFAULT_HEADER "<header class='headermenu'><h1>ESP Easy Mega: title</h1>
" #define WEBPAGE_TEMPLATE_DEFAULT_FOOTER "<footer>
<h6>Powered by Let's Control It community</h6></footer></body></html>" #define WEBPAGE_TEMPLATE_AP_HEADER "<body><header class='apheader'><h1>Welcome to ESP Easy Mega AP</h1>" #define WEBPAGE_TEMPLATE_HIDE_HELP_BUTTON */ *Embed Custom CSS in Custom.h: /* #define WEBSERVER_EMBED_CUSTOM_CSS static const char DATA_ESPEASY_DEFAULT_MIN_CSS[] PROGMEM = { ... ,0}; */ /* ## ## Special settings (rendering settings incompatible with other builds) ## ## */* #define FEATURE_NON_STANDARD_24_TASKS 1 /* ##

```
##############################################
```

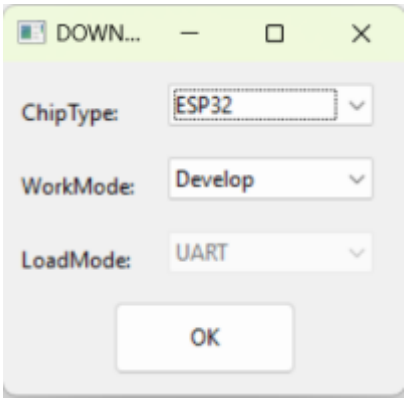
```
 Your Own selection of plugins and controllers
```

```
##################################################################
########################################### */ #define
CONTROLLER_SET_NONE #define NOTIFIER_SET_NONE #define PLUGIN_SET_NONE /*
##################################################################
########################################### ##########
Plugins
##################################################################
########################################### */ #define
FEATURE_SERVO 1 Uncomment and set to 0 to explicitly disable SERVO support #define
USES_P001 Switch #define USES_P002 ADC #define USES_P003 Pulse #define USES_P004 1-
Wire Temperature (Dallas/Maxim DS18B20) #define USES_P005 DHT11/12/22
SONOFF2301/7021/MS01 #define USES_P006 BMP085/180 #define USES_P007 PCF8591
#define USES_P008 Wiegand (RFID) #define USES_P009 MCP23017 #define USES_P010
BH1750 #define USES_P011 ProMini Extender #define USES_P012 LCD2004 #define USES_P013
HC-SR04/RCW-0001 #define USES_P014 SI70xx/HTU21D #define USES_P015 TSL2561 #define
USES_P017 PN532 #define USES_P018 GP2Y10 #define USES_P019 PCF8574 #define
USES_P020 Ser2Net #define USES_P021 Level Control #define USES_P022 PCA9685 #define
USES_P023 OLED SSD1306 #define USES_P024 MLX90614 #define USES_P025 ADS1x15
#define USES_P026 SysInfo #define USES_P027 INA219 #define USES_P028 BMx280 #define
USES_P029 Domoticz MQTT Helper #define USES_P031 SHT1x #define USES_P032 MS5611
(GY-63) #define USES_P033 Dummy Device #define USES_P034 DHT12 #define USES_P036
OLED SSD1306/SH1106 Framed #define P036_FEATURE_DISPLAY_PREVIEW 1 Enable Preview
feature, shows on-display content on Devices overview page #define
P036_FEATURE_ALIGN_PREVIEW 1 Enable center/right-align feature when preview is enabled
(auto-disabled for 1M builds) #define P036_ENABLE_TICKER 1 Enable ticker function #define
USES_P037 MQTT Import #define P037_MAPPING_SUPPORT 1 Enable Value mapping support
#define P037_FILTER_SUPPORT 1 Enable filtering support #define P037_JSON_SUPPORT 1
Enable Json support #define USES_P038 NeoPixel #define P038_FEATURE_NEOPIXELFOR 1
Enable NeoPixelFor/NeoPixelForHSV commands (default enabled for ESP32) #define USES_P039
Thermocouple #define USES_P040 RFID - ID12LA/RDM6300 #define USES_P041 NeoPixel (Word
Clock) #define USES_P042 NeoPixel (Candle) #define USES_P043 ClkOutput #define USES_P044
P1 Wifi Gateway #define USES_P045 MPU6050 #define USES_P046 Ventus W266 #define
USES_P047 Soil moisture sensor #define USES_P048 Motoshield v2 #define USES_P049 MH-Z19
#define USES_P050 TCS34725 RGB Color Sensor with IR filter and White LED #define
USES_P051 AM2320 #define USES_P052 SenseAir #define USES_P053 PMSx003 / PMSx003ST
#define USES_P054 DMX512 #define USES_P055 Chiming #define USES_P056 SDS011/018/198
#define USES_P057 HT16K33_LED #define USES_P058 HT16K33_KeyPad #define USES_P059
Rotary Encoder #define USES_P060 MCP3221 #define USES_P061 PCF8574 / MCP23017 /
PCA8575 #define USES_P062 MPR121 #define USES_P063 TTP229 #define USES_P064
APDS9960 Gesture #define USES_P065 DRF0299 #define USES_P066 VEML6040 #define
USES_P067 HX711_Load_Cell #define USES_P068 SHT3x #define USES_P069 LM75A #define
USES_P070 NeoPixel_Clock #define USES_P071 Kamstrup401 #define USES_P072
HDC1000/HDC1008/HDC1010/HDC1050/HDC1080 #define USES_P073 7-segment display
#define USES_P074 TSL2591 #define USES_P075 Nextion #define USES_P076
HLW8012/BL0937 (Shelly Plug S, Sonoff POW R1, Huafan SS, KMC 70011, Aplic WDP303075,
SK03 Outdoor, BlitzWolf SHP, Teckin, Teckin US, Gosund SP1 v23) #define USES_P077 CSE7766
(Sonoff S31, Sonoff POW R2, Sonoff POW R3xx(D), Sonoff Dual R3) #define USES_P078 Eastron
```

SDMxxx Modbus #define USES_P079 *Wemos / Lolin Motorshield* #define USES_P080 *iButton Sensor DS1990A #define USES_P081* Cron *#define USES_P082* GPS #define USES_P083 *SGP30 TVOC #define USES_P084* VEML6070 *#define USES_P085* AcuDC24x *#define USES_P086* Receiving values according Homie convention. Works together with C014 Homie controller *#define USES_P087* Serial Proxy *#define USES_P088* HeatpumpIR *#define USES_P089* Ping #define USES_P090 *CCS811 TVOC* #define USES_P091 *Serial MCU controlled switch* #define USES_P092 *DLbus* #define USES_P093 *Mitsubishi Heat Pump* #define USES_P094 *CUL Reader* #define USES_P095 *ILI934x / ILI948x* #define USES_P096 *eInk* #define USES_P097 *ESP32 Touch* #define USES_P098 *PWM Motor* #define USES_P099 *XPT2046 touchscreen* #define USES_P100 *DS2423 counter* #define USES_P101 *Wake On Lan* #define USES_P102 *PZEM-004Tv30-Multiple #define USES_P103* Atlas Scientific EZO Sensors (pH, ORP, EZO, DO) *#define USES_P104* MAX7219 dot matrix #define USES_P105 *AHT10/AHT2x #define USES_P106* BME68x *#define USES_P107* SI1145 *#define USES_P108* DDS238-x ZN Modbus energy meters *#define USES_P109* ThermoOLED *#define USES_P110* VL53L0X Time of Flight sensor *#define USES_P111* MFRC522 RFID reader *#define USES_P112* AS7265x *#define USES_P113* VL53L1X ToF #define USES_P114 *VEML6075* #define USES_P115 *MAX1704x* #define USES_P116 *ST77xx #define USES_P117* SCD30 #define USES_P118 *Itho* #define USES_P119 *ITG3205 Gyro* #define USES_P120 *ADXL345 I2C Acceleration / Gravity* #define USES_P121 *HMC5883L* #define USES_P122 *SHT2x* #define USES_P123 *I2C Touchscreens* #define USES_P124 *I2C Multi Relay* #define USES_P125 *ADXL345 SPI Acceleration / Gravity* #define USES_P126 *74HC595 Shift register #define USES_P127* CDM7160 *#define USES_P128* NeoPixel (BusFX) *#define P128_USES_GRB* Default *#define P128_USES_GRBW* Select 1 option, only first one enabled from this list will be used *#define P128_USES_RGB* #define P128_USES_RGBW *#define P128_USES_BRG* #define P128_USES_BGR *#define P128_USES_RBG* #define P128_ENABLE_FAKETV 1 *Enable(1)/Disable(0) FakeTV effect, disabled by default on ESP8266 (.bin size issue), enabled by default on ESP32* #define USES_P129 *74HC165 Input shiftregisters* #define USES_P131 *NeoPixel Matrix* #define USES_P132 *INA3221 #define USES_P133* LTR390 UV *#define USES_P134* A02YYUW #define USES_P135 *SCD4x* #define P135_FEATURE_RESET_COMMANDS 1 *Enable/Disable quite spacious (~950 bytes) 'selftest' and 'factoryreset' subcommands* #define USES_P137 *AXP192* #define USES_P138 *IP5306* #define USES_P141 *PCD8544 Nokia 5110 LCD #define USES_P142* Position - AS5600 *#define USES_P143* I2C Rotary encoders *#define P143_FEATURE_INCLUDE_M5STACK 0* Enabled by default, can be turned off here *#define P143_FEATURE_INCLUDE_DFROBOT 0* Enabled by default, can be turned off here *#define P143_FEATURE_COUNTER_COLORMAPPING 0* Enabled by default, can be turned off here *#define USES_P144* PM1006(K) (Vindriktning) #define USES_P145 *MQxxx (MQ135 CO2, MQ3 Alcohol) #define USES_P146* Cache Reader #define USES_P147 *SGP4x* #define P147_FEATURE_GASINDEXALGORITHM 0 *Enabled by default, can be turned off here* #define USES_P148 *POWR3xxD/THR3xxD #define USES_P150* TMP117 Temperature #define USES_P151 *Honeywell Pressure #define USES_P152* ESP32 DAC #define USES_P153 *SHT4x #define USES_P154* BMP3xx I2C *#define USES_P159* Presence - LD2410 Radar detection *#define USES_P162* Output - MCP42xxx Digipot #define USES_P163 *Environment - RadSens I2C radiation counter #define USES_P164* Gases - ENS16x TVOC/eCO2 *#define USES_P166* Output - GP8403 Dual channel DAC (Digital Analog Converter) #define USES_P167 *Environment - Sensirion SEN5x / Ikea Vindstyrka #define USES_P168* Light - VEML6030/VEML7700 #define USES_P169 *Environment - AS3935 Lightning Detector #define USES_P170* Input - I2C Liquid level sensor #define USES_P172 *BMP3xx SPI. #define USES_P173* Environment - SHTC3 /*
##
Controllers

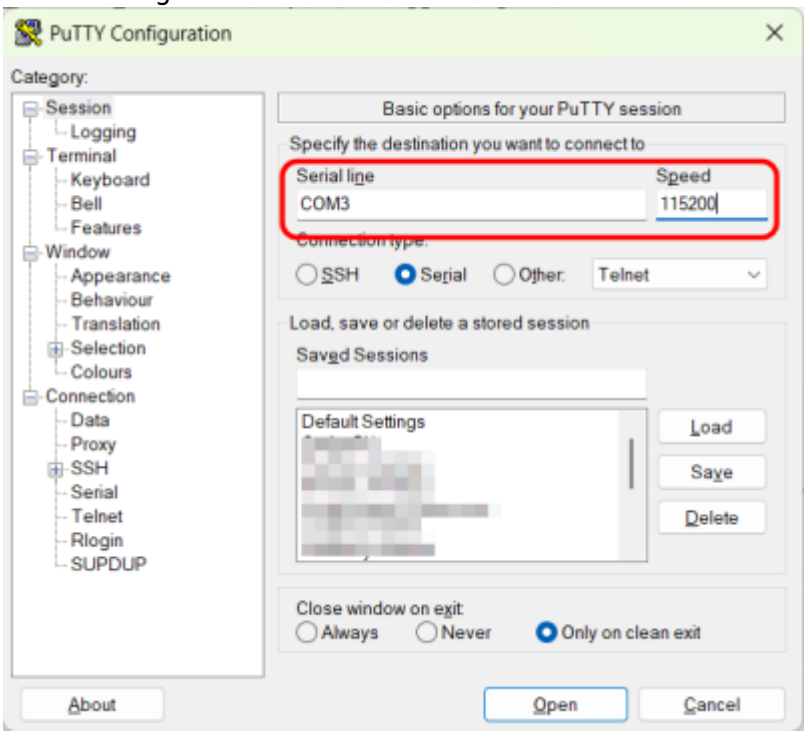
*## */ #define USES_C001 *Domoticz HTTP #define USES_C002* Domoticz MQTT #define USES_C003 *Nodo telnet #define USES_C004* ThingSpeak #define USES_C005 *Home Assistant (openHAB) MQTT #define USES_C006* PiDome MQTT #define USES_C007 *Emoncms #define USES_C008* Generic HTTP #define USES_C009 *FHEM HTTP #define USES_C010* Generic UDP #define USES_C011 *Generic HTTP Advanced #define USES_C012* Blynk HTTP #define USES_C013 *ESPEasy P2P network #define USES_C014* homie 3 & 4dev MQTT #define USES_C015 *Blynk #define USES_C016* Cache controller #define USES_C017 *Zabbix #define USES_C018* TTN/RN2483 /* ## ########## Notifiers ## */ #define USES_N001 *Email #define USES_N002* Buzzer #endif *ESPEASY_CUSTOM_H </code> This is example text. ++ Title | This table | is only shown | when you unfold the block |*

++ 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

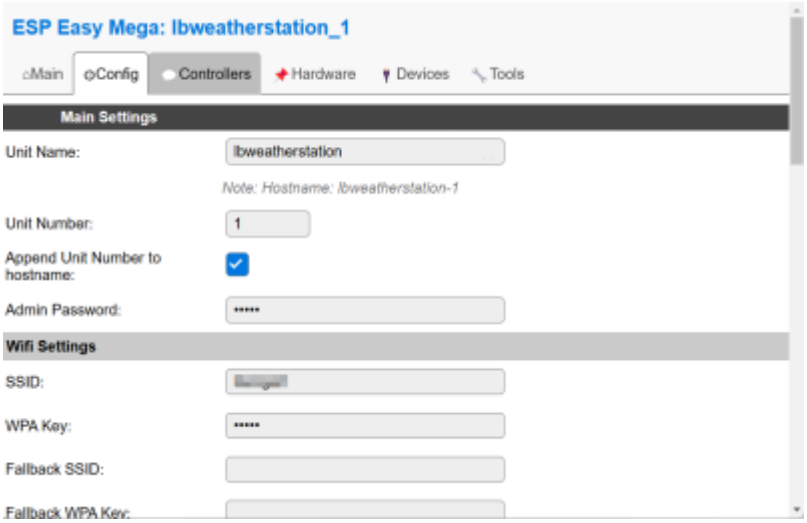


gelistet ist, wählt ESP32) und setzt die Software auf `Develop`: Oben wählt ihr als erstes das Firmware-File aus, welches ihr flashen wollt. Nutzt ihr die fertig kompilierten Firmwares des ESPEasy Projekts, geht ihr dazu in das `bin`-Verzeichnis der entpackten ESPEasy-Firmware und sucht Euch die `collection_G` 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.

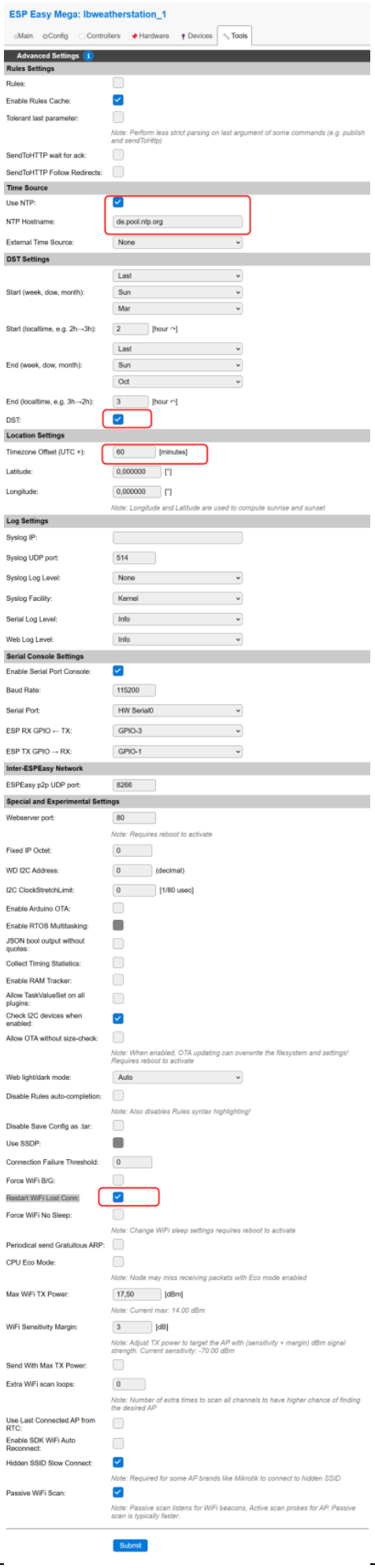


Nun verbindet ihr den ESP32 wieder mit dem Rechner, wartet 1-2 Sekunden, bis der ESP erkannt wurde und klickt in PuTTY dann auf Open. Ihr solltet nun in PuTTY die Bootmeldungen des ESP32 sehen und er sollte das WLAN zur Erstkonfiguration aufmachen - die Meldungen sollten in etwa so aussehen: Der ESP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



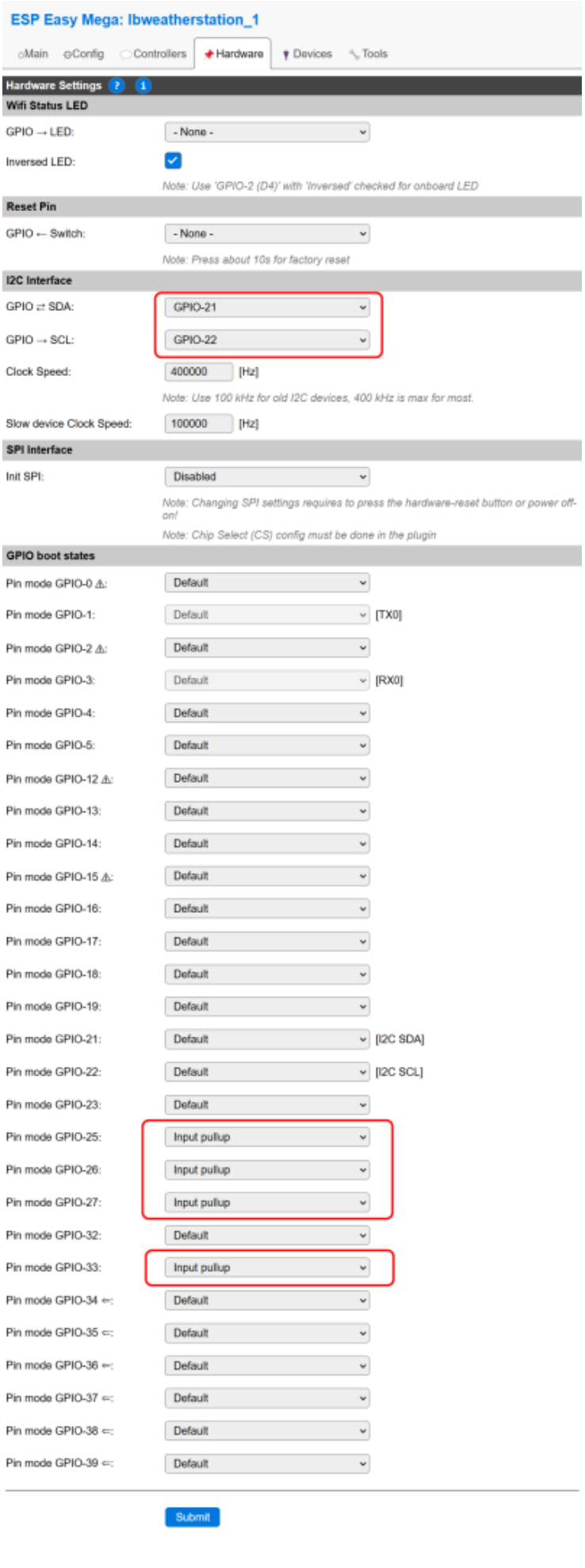
Im Tab `Tools` öffnet ihr nun noch unter System die `Advanced` Options. Unter `Time Source` aktiviert ihr NTP und nutzt den NTP-Server `de.pool.ntp.org`. Prüft die `DST Settings` (Sommer-/Winterzeit) und die `Location Settings` (hier die `Timezone Offset (UTC +)`). Unter `Special and Experimental Settings` 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

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



Damit ist die

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