



# **WLAN mit ESP-NOW**

Uwe Berger  
bergeruw@gmx.net

---



# ~~WLAN~~ Funk mit ESP-NOW

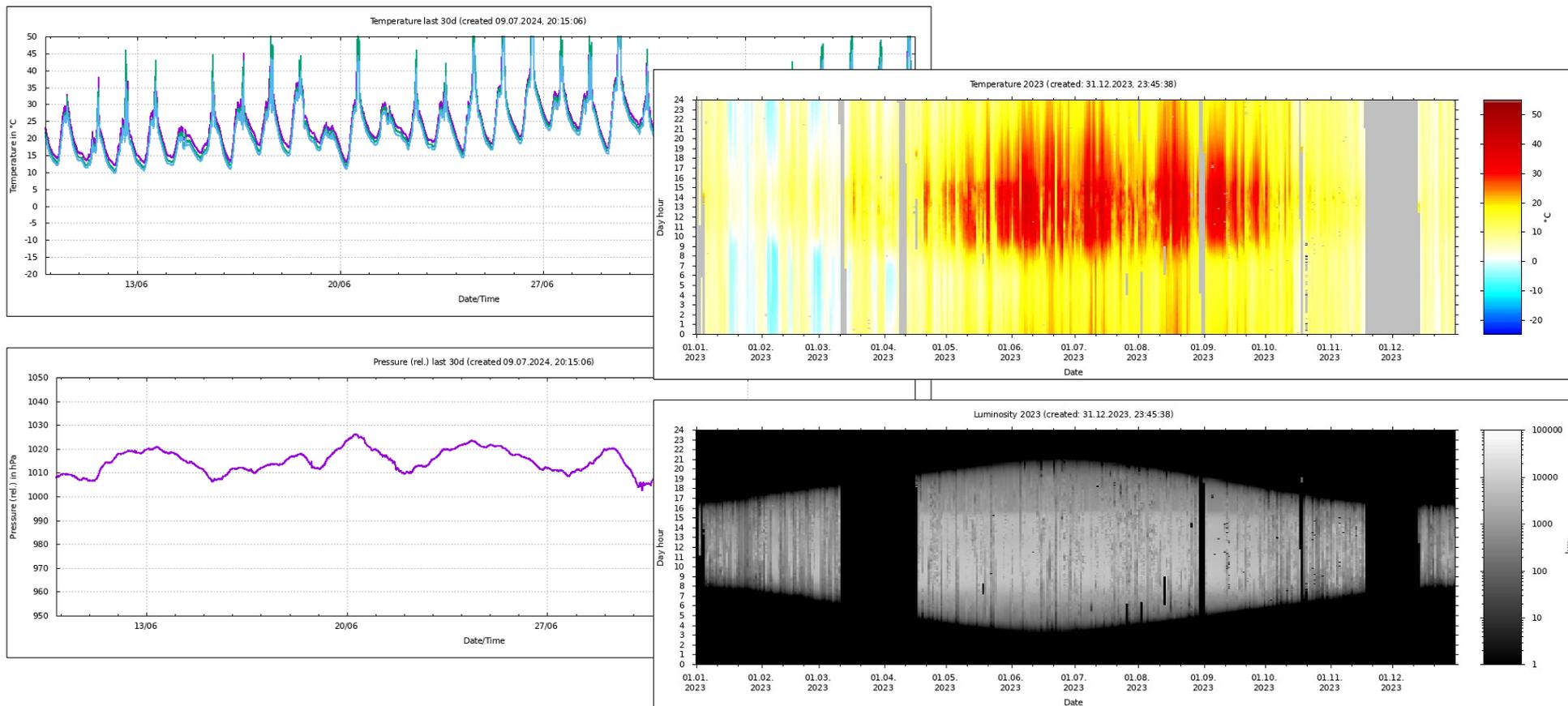
Uwe Berger  
bergeruw@gmx.net

---

# Uwe Berger

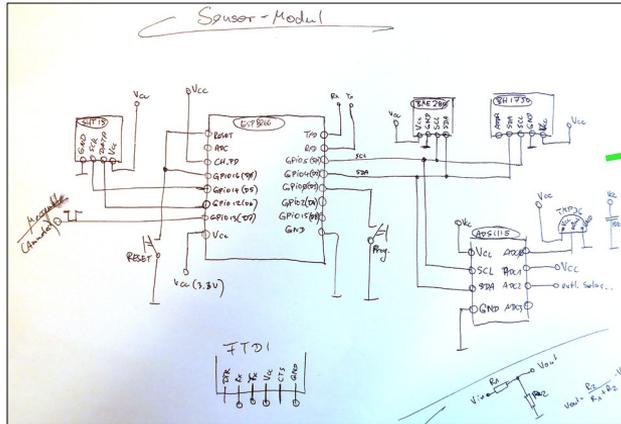


# Motivation



# Motivation

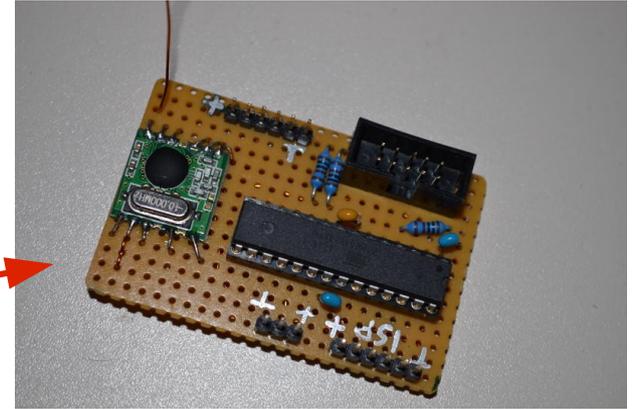
...hmmm..., da gibt es keine Steckdose in der Nähe?!?!



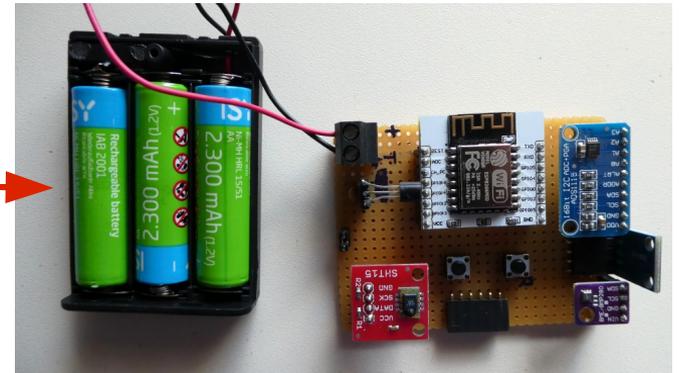
# Motivation

3 Evolutionsstufen einer Wetterstation:

1. ATmega328 als MC; Funkbrücke via RFM12-Modul (ISM-Band; 433MHz)



2. ESP8266 als MC; Funkbrücke via WLAN (MQTT)



3. ...darüber reden wir heute!

# Was erzähle ich heute?

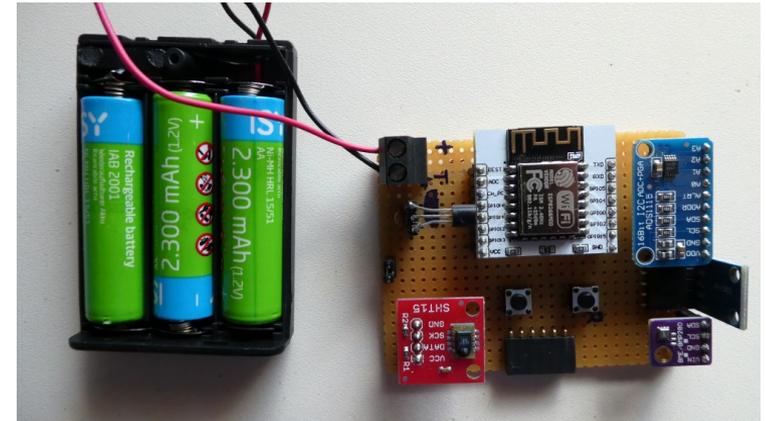
- Review: 2. Evolutionsstufe → Datenübertragung über „normales“ WLAN
- ESP-NOW
  - Grundsätzliches, Voraussetzungen, Randbedingungen
  - Implementierungsbeispiele
  - Konkrete Ergebnisse

## 2. Evolutionsstufe

→ <https://github.com/boerge42/weatherstations/tree/master/esp8266>

- Wetterstation 2 (Aufbau: 2020):

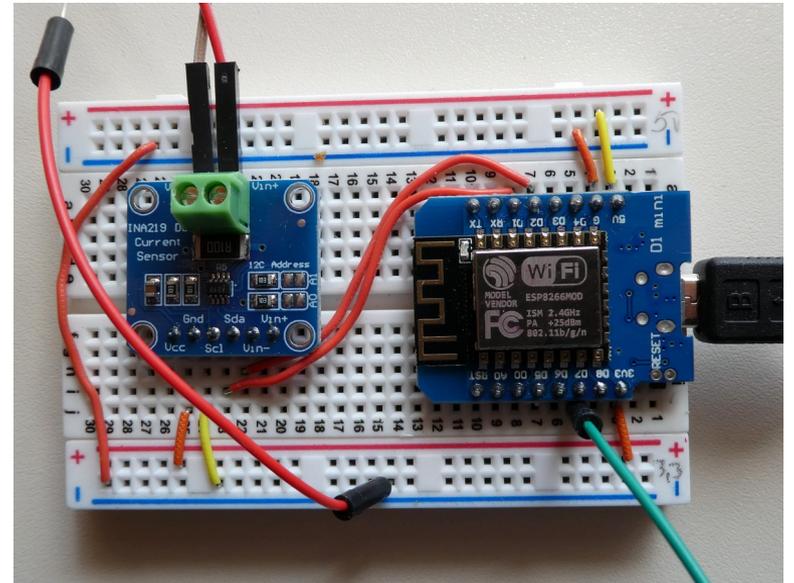
- ESP8266
- Funkbrücke via WLAN (MQTT)
- Diverse Sensoren
- Stromversorgung via Akkus und Mini-Solarpaneel
- ESP-Firmware:  
...DeepSleep → Messen/Senden → DeepSleep...



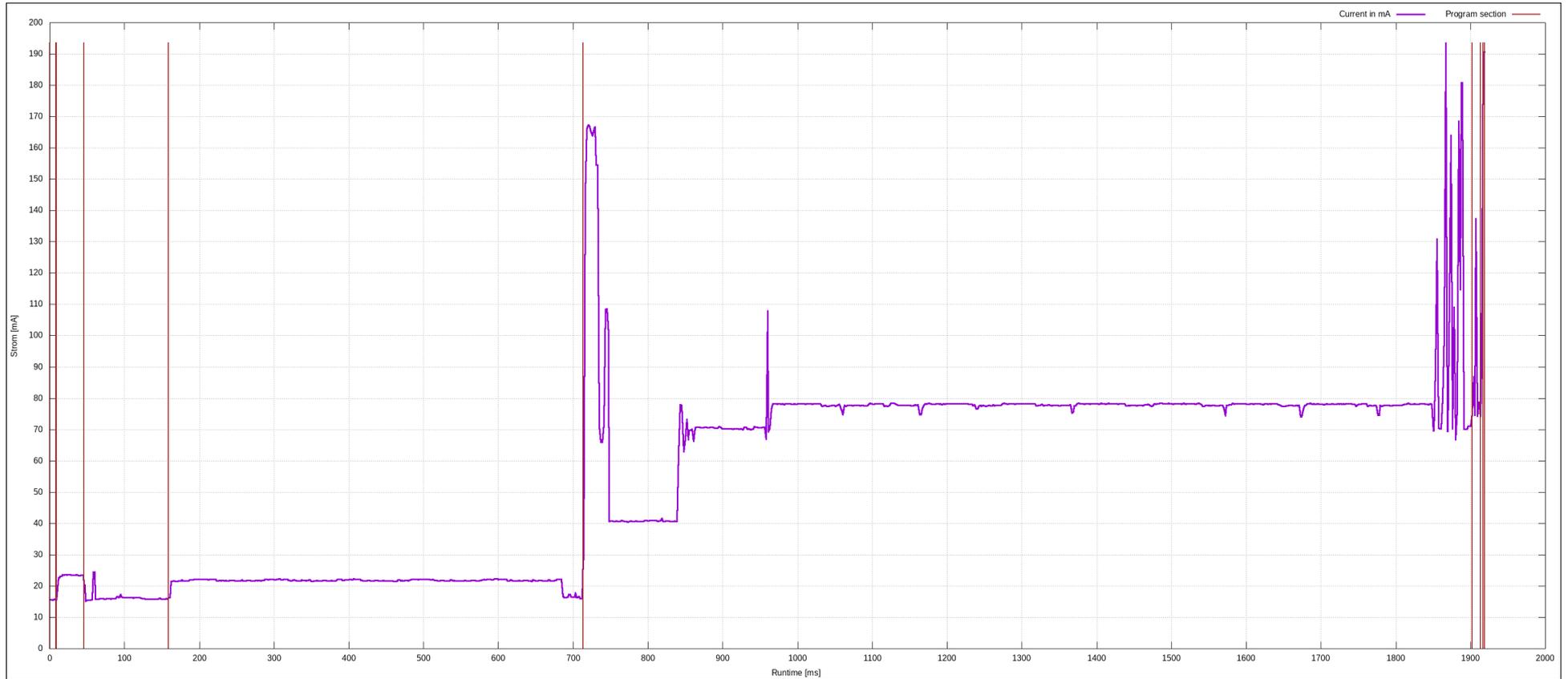
## 2. Evolutionsstufe (Stromverbrauch)

Messung Strom über die Zeit  
(mein Setup):

- Strom: INA219
- Zeit: interessante Zeitpunkte via „GPIO-Toggle“ in Firmware des Messobjektes; Auflösung ca. 1ms
- Ansteuerung/Auswertung/Be-  
rechnung etc. z.B. via Raspberry



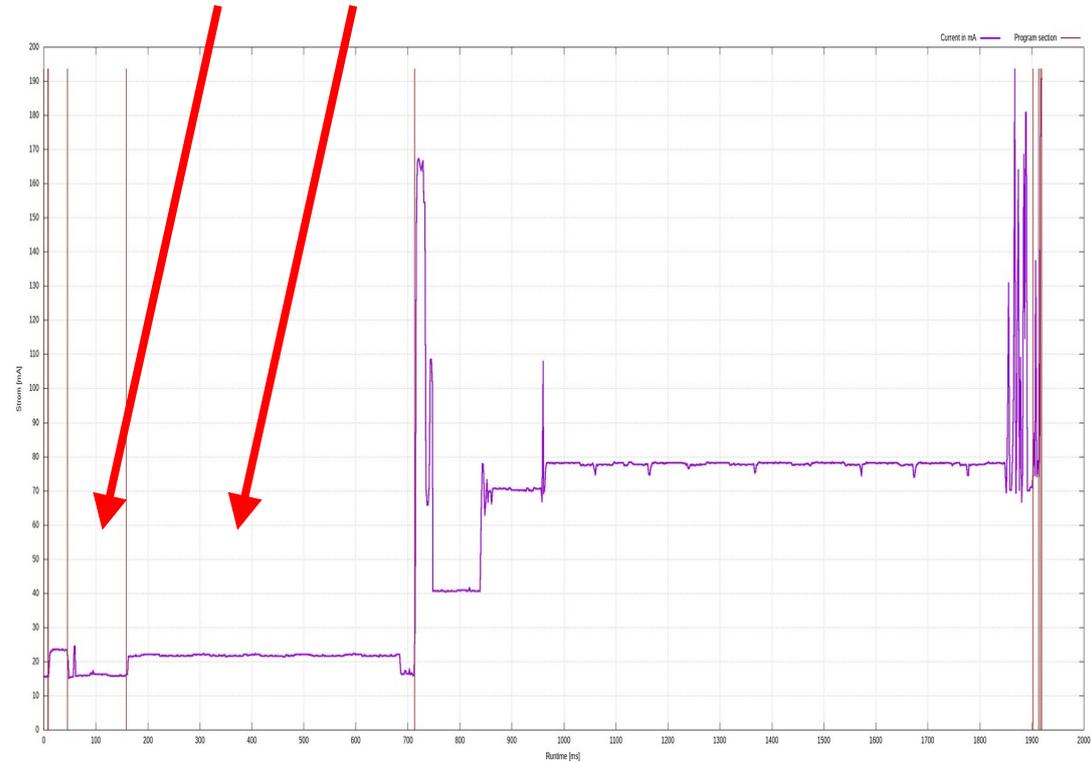
# 2. Evolutionsstufe (Stromverbrauch)



# 2. Evolutionsstufe (Kurvendiskussion)

## Abschnitt: Sensoren

- ca. 665ms;
- ca. 15 - 22mA
- Abhängig von verwendeten Sensoren und den jeweiligen Bibliotheken...



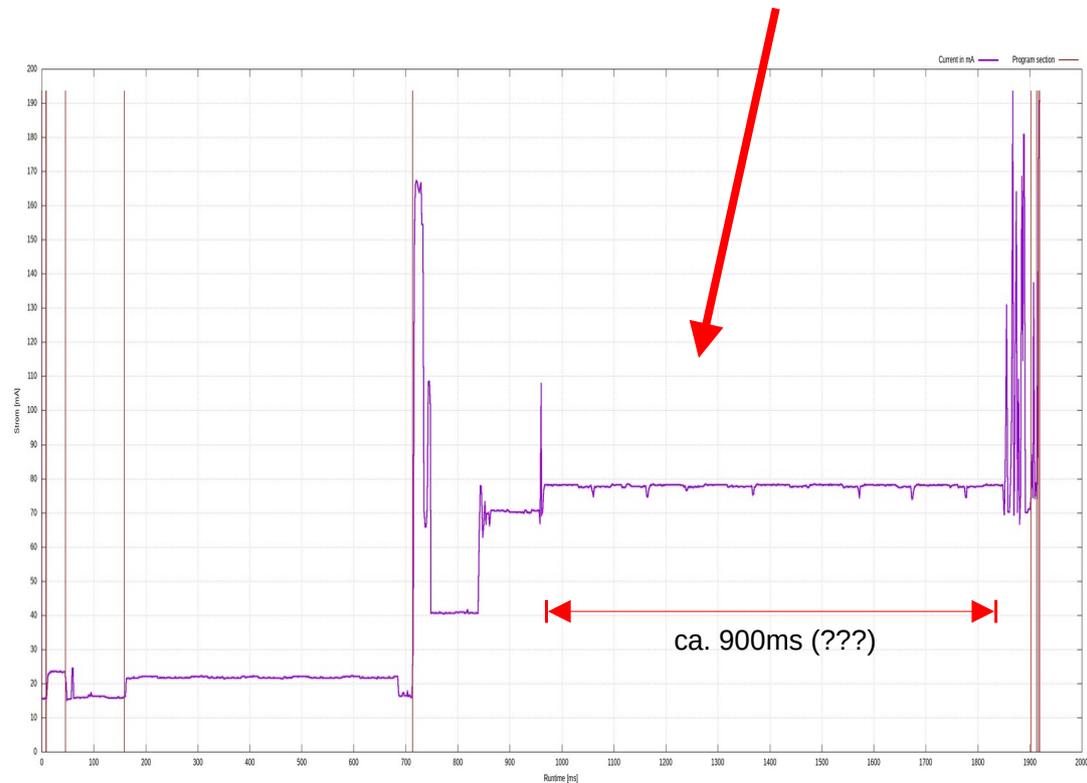
# Ausflug: OSI-Schichtenmodell

7.	Anwendungsschicht	Bereitstellung von Funktionen für die jeweilige Anwendung, Datenein-/ausgabe	FTP, DNS, DHCP, HTTP, SMTP, MQTT, ...
6.	Darstellungsschicht	Übersetzung systemabhängige Datendarstellung in unabhängiges Format	
5.	Sitzungsschicht	Verbindungsauf-/abbau, Steuerung des Datenaustausches (wer, wann)	
4.	Transportschicht	Transport der Nachrichten zwischen den Kommunikationspartnern	TCP, UDP, SCTP, SPX, ...
3.	Vermittlungsschicht	Routing der Datenpakete zum nächsten Knoten	IP, IPsec, IPX, ICMP, ...
2.	Sicherungsschicht	Segmentierung des Bitstroms in Blöcke; Prüfsummen pro Block	IEEE 802.3, MAC, IEEE 803.11, ...
1.	Bitübertragungsschicht	physikalische Bitübertragung, je nach Medium	1000Base-T, Token Ring, V.24, RS232, X.21, ...

# 2. Evolutionsstufe (Kurvendiskussion)

Abschnitt: an AP anmelden

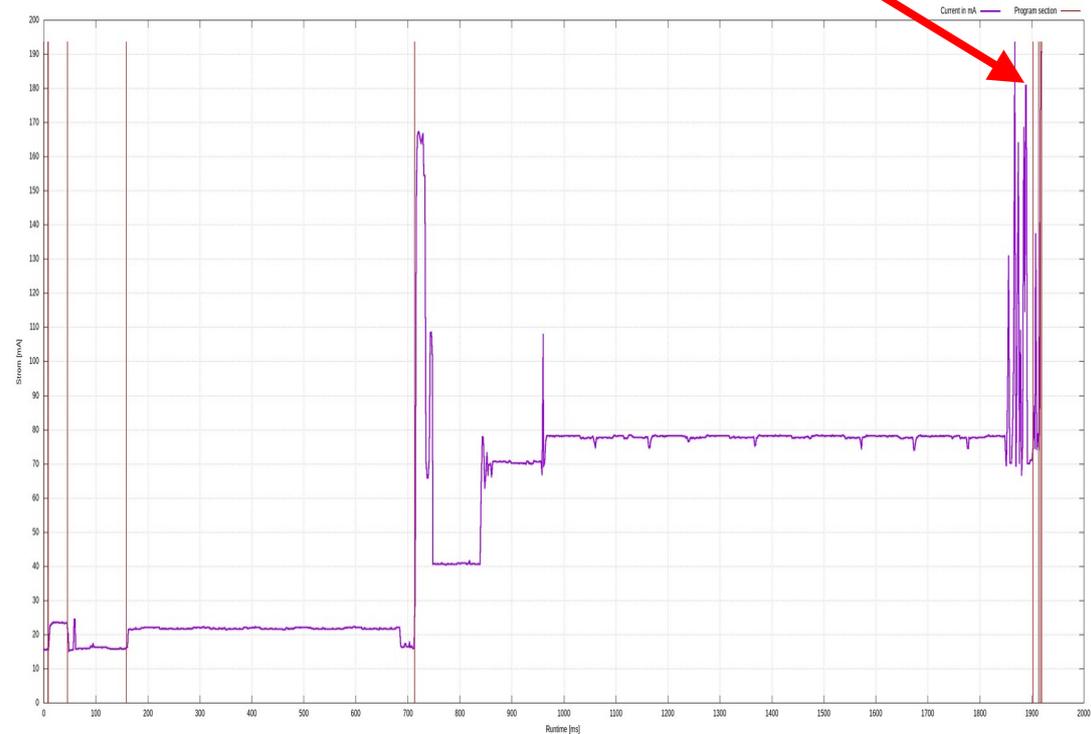
- ca. 1200ms (!)
- > 80mA
- schon optimiert (Versuch mit letztem AP...)
- DNS/DHCP
- ...was macht der ESP in den ca. 900ms?



# 2. Evolutionsstufe (Kurvendiskussion)

## Abschnitt: MQTT senden

- ca. 21ms
- > 130mA
- an Broker anmelden,  
Daten senden
- QoS-Level...



## 2. Evolutionsstufe (Kurvendiskussion)

Erkenntnisse:

- Sensoren → wenn Optimum gefunden, dann feste Größe
- Wozu brauchen wir das ganze WiFi- und TCP/IP-Geraffel?
  - ca. 1220ms bei relativ hohem Stromverbrauch, trotz Optimierungen!
- **ESP-NOW** → Nutzung der vorhandenen ESP-Funkhardware, ohne komplizierte/zeitaufwendige Protokolle und Abhängigkeiten von weiterer Infrastruktur!

# ESP-NOW: Was ist das?

- Entwickler: Espressif Systems
- Proprietäres drahtloses Kommunikationsprotokoll für ESP8266 und ESP32
- Peer-to-Peer-Kommunikationsprotokoll
- Ermöglicht eine stromsparende, schnelle, direkte Kommunikation zwischen ESP-Geräten ohne weitere Infrastruktur

# ESP-NOW: OSI-Schichtenmodell

7.	Anwendungsschicht	<b>ESP-NOW</b>	
6.	Darstellungsschicht		
5.	Sitzungsschicht		
4.	Transportschicht		
3.	Vermittlungsschicht		
2.	Sicherungsschicht	Segmentierung des Bitstroms in Blöcke; Prüfsummen pro Block	IEEE 802.3, MAC, IEEE 803.11, ...
1.	Bitübertragungsschicht	physikalische Bitübertragung, je nach Medium	1000Base-T, Token Ring, V.24, RS232, X.21, ...

# ESP-NOW: Kennzahlen

- Datenrate: 1Mbit/s
- Reichweite: je nach Umgebung und Antenne; typisch 200m im Freien
- Latenz: sehr gering (1-5ms)
- Stromverbrauch: sehr gering
- Maximal 250 Byte Nutzdaten pro Datenpaket
- Maximal 20 Kommunikationspartner

# ESP-NOW: Netzwerktopologie

Möglichkeiten:

- Unidirektional (Gerät entweder Sender oder Empfänger):
  - Ein Sender, ein Empfänger
  - Ein Sender, mehrere Empfänger
  - Mehrere Sender, ein Empfänger
- Bidirektional (Gerät gleichzeitig Sender/Empfänger)
  - Zwischen zwei Geräten
  - Zwischen mehreren Geräten

# ESP-NOW anwenden

ESP-NOW Implementierung, Bibliotheken/Module u.a. für:

- C (Espressif-SDK)
- Arduino (C++)
- Micropython
- Lua

# ESP-NOW: Implementierungsbeispiele

- Unidirektional (ein Sender, ein Empfänger)
- Beispiele verwenden jeweils ESP8266 als Sender/Empfänger
- Vollständige Beispiele:
  - <https://github.com/boerge42/ESP-NOW-Experimente>
- Ausführlichere API-Referenz z.B.:
  - [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp\\_now.html#](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_now.html#)
  - [https://wolles-elektronikkiste.de/esp-now#function\\_overview](https://wolles-elektronikkiste.de/esp-now#function_overview)

# ESP-NOW: Sender

```
#include <ESP8266WiFi.h>    // notwendige Bibliotheken...
#include <espnw.h>

uint8_t rx_mac_addr[6] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC};    // MAC-Adresse des ESP-NOW-Empfängers
#define ESP_NOW_CHANNEL    1    // WiFi-Kanal

void OnDataSend(uint8_t * mac, uint8_t status) {...}    // Callback bei Senden
. . .

WiFi.mode(WIFI_STA);    // WiFi-Device in Station-Mode

if (espnw_init() != 0) {...}    // Init ESP-NOW

espnw_set_self_role(ESP_NOW_ROLE_CONTROLLER);    // ESP-NOW-Rolle setzen (Sender)

espnw_register_send_cb(OnDataSend);    // Callback-Funktion, wenn Daten gesendet wurden

espnw_add_peer(rx_mac_addr, ESP_NOW_ROLE_SLAVE, ESP_NOW_CHANNEL, NULL, 0);    // RX-MAC, Kanal setzen

espnw_send(rx_mac_addr, (uint8_t *) &sensor_values, sizeof(sensor_values));    // Daten senden
```

# ESP-NOW: Empfänger

```
// entsprechende Includes...

uint8_t rx_mac_addr[6] = {0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC}; // MAC-Adresse des ESP-NOW-Empfängers
#define ESP_NOW_CHANNEL 1 // WiFi-Kanal

void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len) {...} // Callback bei Empfang

. . .

WiFi.mode(WIFI_STA); // Station-Mode

wifi_set_macaddr(STATION_IF, rx_mac_addr); // MAC-Adresse setzen

wifi_set_channel(ESP_NOW_CHANNEL); // WiFi-Kanal setzen

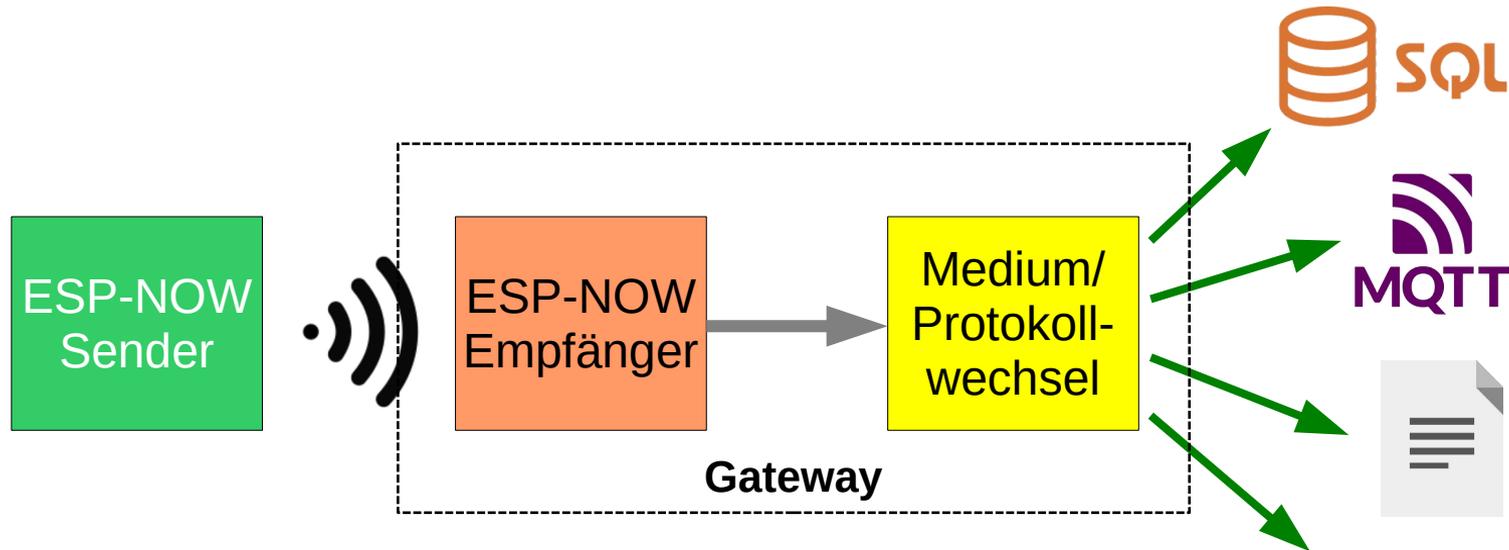
if (esp_now_init() != 0) {...} // Init ESP-NOW

esp_now_set_self_role(ESP_NOW_ROLE_SLAVE); // ESP-NOW-Rolle setzen (Empfänger)

esp_now_register_recv_cb(OnDataRecv); // Callback-Funktion registrieren
```

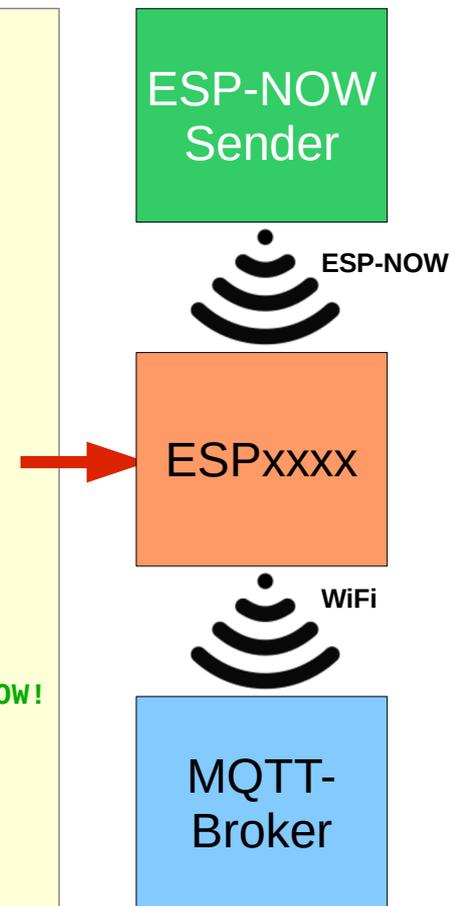
# ESP-NOW: Gateway

- Gateway: ein Netzwerkknoten, welcher Netzwerke mit unterschiedlichen Übertragungsmedien/-protokollen verbindet



# ESP-NOW: Gateway → gleicher ESP

```
. . .  
  
//*****  
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len)  
{  
    memcpy(&sensor_values, incomingData, sizeof(sensor_values)); // empfangene Daten  
    . . . // aufbereiten  
    mqtt_publish_values(); // via MQTT versenden  
}  
  
. . .  
// gleicher Code wie ESP-NOW-Empfaenger  
. . .  
  
if (WiFi.status() != WL_CONNECTED) {wifi_connect();} // an WLAN anmelden;  
// mit gleichem Chanel, wie ESP-NOW!  
if (!mqtt_client.connected()) {mqtt_connect();} // an MQTT-Broker anmelden  
  
esp_now_register_recv_cb(OnDataRecv); // Callback-Funktion registrieren
```



# ESP-NOW: Gateway → seriell Tx

```
. . .  
//*****  
void OnDataRecv(uint8_t * mac, uint8_t *incomingData, uint8_t len)  
{  
    char json[550];  
  
    memcpy(&sensor_values, incomingData, sizeof(sensor_values)); // empfangene Daten  
                                                                // aufbereiten  
  
    . . .  
    Serial.write(json, strlen(json)); // Daten über serielle  
    Serial.write("\n"); // Schnittstelle senden  
}  
  
. . .  
Serial.begin(115200); // Serielle Schnittstelle initialisieren  
. . .  
esp_now_register_recv_cb(OnDataRecv); // Callback-Funktion registrieren
```

ESP-NOW  
Sender



ESPxxxx

seriell

...

# ESP-NOW: Gateway → seriell Rx

```
#!/usr/bin/python3
.
.
.
ser = serial.Serial(SERIAL_PORT, SERIAL_RATE)           # serielle Schnittstelle

mqtt_client = mqtt.Client()                             # MQTT
mqtt_client.username_pw_set(username=MQTT_USER, password=MQTT_PWD)
mqtt_client.connect(MQTT_BROKER, MQTT_PORT)
mqtt_client.loop_start()

while True:
    try:
        rx_buf = ser.readline().decode('utf-8')         # Daten empfangen

        rx_payload = json.loads(rx_buf)                 # Daten aufbereiten
        topic = rx_payload["topic"]["TX_NAME"]
        payload = rx_payload["payload"]
        payload = json.dumps(payload)

        mqtt_client.publish(topic, payload, qos=0)     # MQTT versenden
    except:
        pass
```

ESP-NOW  
Sender



ESPxxxx

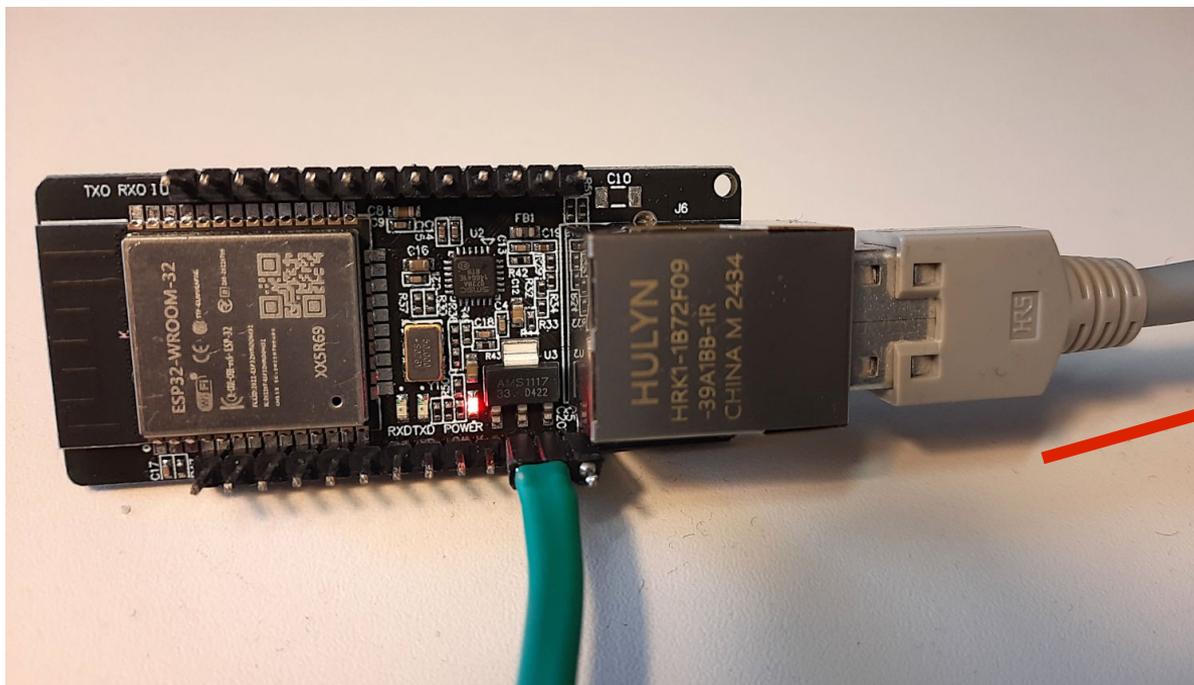
seriell

z.B.  
Raspberry

# ESP-NOW: Gateway mit WT32-ETH01

Implementierungsbeispiel:

→ <https://github.com/boerge42/ESP-NOW-Experimente/...>



ESP-NOW  
Sender



ESP32



ETH01

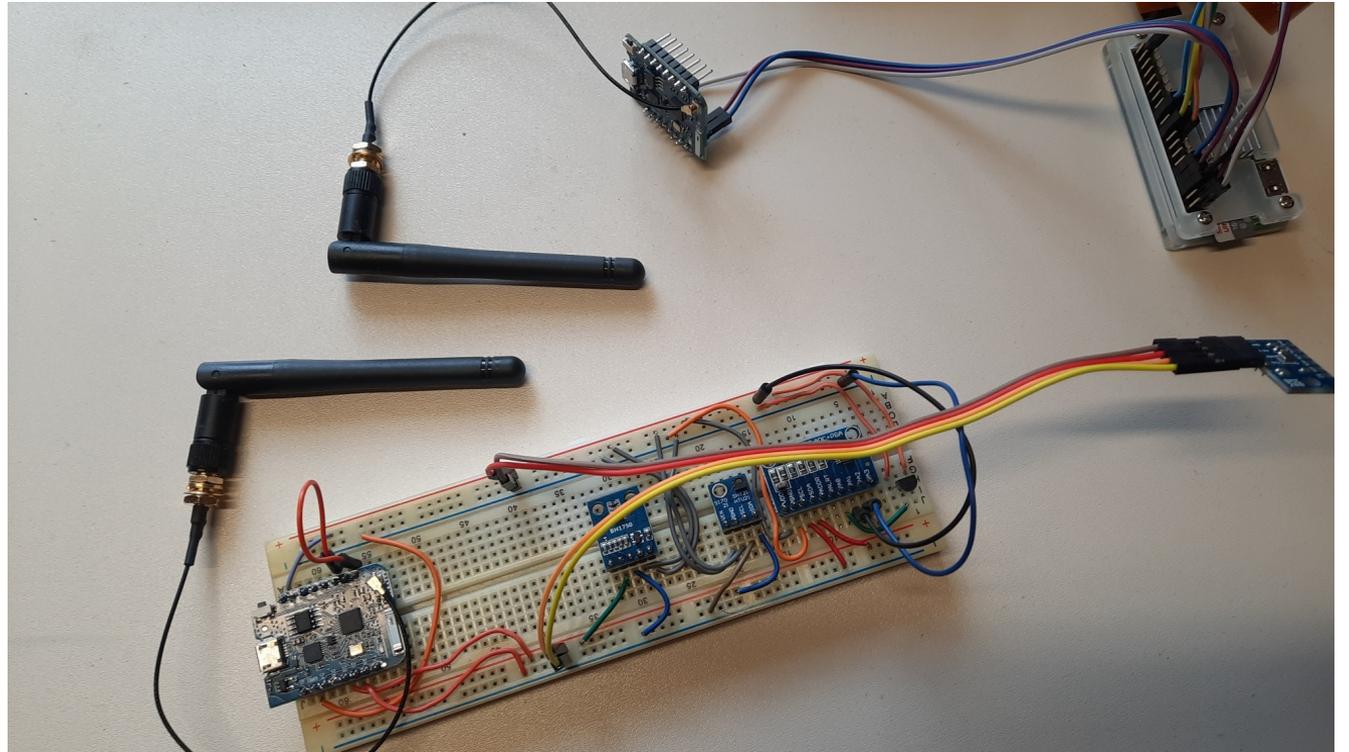
MQTT via  
Ethernet

# ESP-NOW: Verschlüsselung

- Verschlüsselte Datenkommunikation ist möglich
  - AES-128-CCM (Verschlüsselung und Integrität)
  - Schlüsselaustausch erfolgt während des Pairing-Prozesses
  - Entsprechende Algorithmen sind in den Expressif-Bibliotheken enthalten
- Beispiele:
  - <https://randomnerdtutorials.com/esp32-esp-now-encrypted-messages/>
  - <https://microcontrollerslab.com/esp32-esp-now-encrypted-messages/>

# 3. Evolutionsstufe

...zurück zum Sensormodul mit ESP-NOW als Datenübertragungsmedium...



# ESP-NOW (Sensormodul)

## Messergebnisse:

- Laufzeit: DeepSleep → ... → DeepSleep → ca. 150ms
  - ...davon ca. 1ms (!) Datensenden
- Strommessung habe ich mir deshalb gespart :-)

## Interpretation:

- Es kommt zeitlich nur noch die Abfrage Sensoren zum tragen → Datenblätter genau studieren!
- **Verkürzung der Wachzeit des ESP um Faktor 13(!)**

# ESP-NOW: ... nochmal Pro und Kontra

## Pro

- Geringer Stromverbrauch
- Einfache Einrichtung (kein Router oder AP notw.)
- Niedrige Latenzzeiten
- Flexibel einsetzbar

## Kontra

- Kleine Datenpaketgröße
- Paarungsprozess über MAC-Adresse
- Teilweise beschränkte Reichweiten

# ESP-NOW: Anwendungsgebiete

- Überall da, wo es auf geringe Latenzen und geringen Stromverbrauch ankommt, z.B.:
  - Heimautomation
  - Sensornetzwerke
  - Wearables
  - Spielzeug, Gadgets

# Weiterführende Informationen

- 1. Evolutionsstufe:
  - <https://chemnitzer.linux-tage.de/2014/de/vortraege/detail/153/>
  - <https://github.com/boerge42/weatherstations/tree/master/avr>
- 2. Evolutionsstufe:
  - <https://programm.froscon.org/2021/events/2638.html>
  - <https://github.com/boerge42/weatherstations/tree/master/esp8266>
- 3. (und aktuelle) Evolutionsstufe:
  - <https://github.com/boerge42/ESP-NOW-Experimente>
- ...weiterführende Informationen zu ESP-NOW findet sicherlich jeder mit der Suchmaschine seiner Wahl...



# Fragen?

...ansonsten Danke &  
Ende!