



ESP32 Arduino Workshop

Helmut Tschemernjak

Arduino-Hannover

www.arduino-hannover.de

Installation von Arduino

■ Arduino installieren

Hier laden: <https://www.arduino.cc/en/Main/Software>



■ Software für ESP32

<https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/windows.md>

Git installieren:

<https://git-scm.com/download/win> (alle Defaults übernehmen)

■ Clone vom `arduino-esp32.git`

Git Repo gemäß Anleitung laden

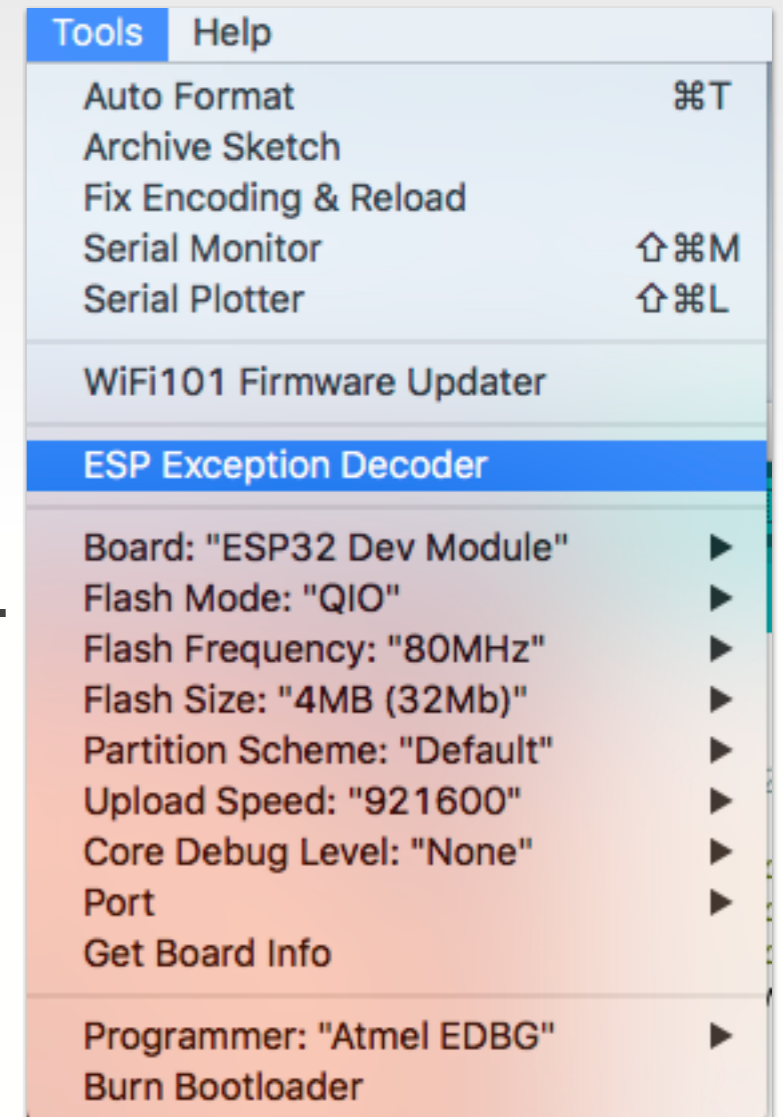
get.exe ausführen (Tools werden geladen)

„CP210x“-Treiber von Silicon Labs installieren (Windows)

ESP Exception Decoder

■ Zeigt Zeile vom Crash an

Einfach die Backtrace-Zeile vom Serial Monitor in das Fenster „ESP Exception Decoder“ kopieren.
Funktioniert für ESP und ESP32



Decoding 4 results

```
0x40080f3a: startTimer() at /Users/.../arduino-esp32.cpp line 357
0x40081051: Timeout::restart() at /Users/./arduino-mbed.cpp line 300
0x400d120b: Timeout::attach_us(Callback, long) at arduino-mbed.h line 386
0x40081801: __timerISR at esp32-hal-timer.c line 173
```

Arduino

Arduino IDE

C/C++ Code

Standard C / C++ Compiler

Upload Tools (openocd, bootloader, ISP, ...)



Programm

Standard C / C++ Runtime

Arduino Hardware Treiber & OS

(Treiber: GPIO, SPI, I2C, Analog, UART) (OS: digitalWrite(), delay() ...)

MCU

mbed-os

mbed IDE

C/C++ Code

Standard C / C++ Compiler

Upload Tools (openocd, bootloader, ISP, ...)



Programm

Standard C / C++ Runtime

mbed-os Hardware Treiber & OS

(Treiber: GPIO, SPI, I2C, Analog, UART) (OS: DigitalOut, Timer, ...)

MCU

Programmieren

■ **C/C++ Programmieren lernen**

Windows, Mac, Linux, Arduino, mbed, PI, identisch immer C/C++

■ **Embedded Spezialitäten**

GPIO, SPI, I2C, UART usw. ist auch immer gleich

APIs zwischen Arduino und mbed sind natürlich unterschiedlich

■ **Debugger**

GDB, OpenOCD ist auch wieder identisch

■ **Tipp von mir**

Einfach mal ein kleines Programm unter Windows in C/C++ schreiben, da gibt es gute Entwicklungsumgebungen mit Debugger. Die getesteten Funktionen dann einfach in Arduino übernehmen.

Arduino OS versus mbed-os

	Arduino	mbed-os
Pin 2 LED an:	<code>pinMode(2, OUTPUT); digitalWrite(2, HIGH);</code>	<code>DigitalOut led(2); led = 1;</code>
Pin 2 abfragen:	<code>if (digitalRead(2) == HIGH) ...</code>	<code>if (led) ...</code>
Sekunde warten:	<code>delay(1000);</code>	<code>wait_ms(1000);</code>
Interrupt Pin 3:	<code>pinMode(3, INPUT); attachInterrupt(3, &func, FALLING);</code>	<code>InterruptIn intr(3); intr.fall(callback(&func));</code>
Sleep:	🤔	<code>sleep(); oder deepsleep();</code>
(Abgelaufene Zeit) Timer:	🤔 Reboot bei millis() D21, 49 Tage Probleme bei micros() D21, ESP32	<code>Timer t; t.start(); t.read_ms(); // oder us(), sec()</code>
Timeout in 200 ms:	🤔	<code>Timeout t; t.attach(callback(&func), 200);</code>

Neue Funktionen per Arduino-mbed-APIs

- **sleep();**

Legt den Prozessor bis zum nächsten Interrupt schlafen

- **deepsleep();**

Schlafmodus mit geringstem Energieverbrauch

- **pause(millis);** (nutzt ESP lightsleep, 0,8 mA, alles bleibt erhalten)

- **Timeout t;**

Beliebige Timer, welche eine Funktion nach einer gewissen Zeit aufrufen

- **Weitere mbed-kompatible APIs**

DigitalIn, DigitalOut, DigitalInOut, SPI, InterruptIn, Timer

Weitere schöne Funktionen

■ **RTCInit(__DATE__, __TIME__);**

Initialisiert die RTC Uhrzeit beim Einspielen eines neuen Programms

■ **dprintf("Hello World");**

Ein „print“-Befehl mit Zeitstempel, Ausgabe auf den Serial Monitor mit automatischem Zeilenende:

```
18:13:51.494086 ESP32: Rev: 1, 240 MHz, IDF(v3.1)
```

```
18:13:51.522698 Power: 3.30V (ADC: 3335 Vref: 1.121)
```

rprintf(): ohne Zeitstempel, ohne Newline

■ **dump("Message",void *buffer, int length);**

```
18:14:01.784660 dump("Message", 0x3ffdb8ca, 32 bytes)
```

```
3ffdb8ca: 54 68 65 20 73 65 72 76 65 72 20 66 65 65 6c 73 The serv er feels
```

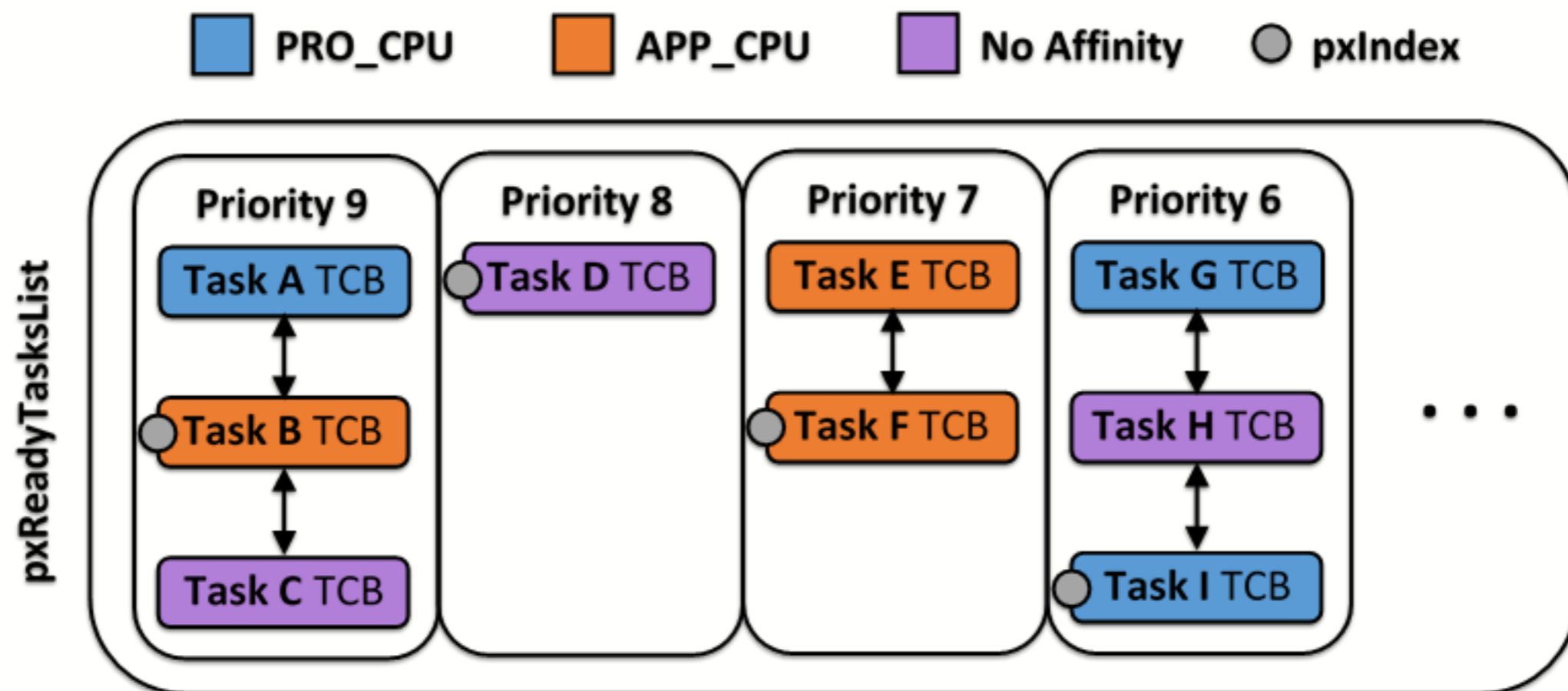
```
3ffdb8da: 20 76 65 72 79 20 67 6f 6f 64 20 74 6f 64 61 79 very go od today
```

Beispiel	Funktion
Blinky:	Periodisches Blinken einer LED
BlinkyEnhanced:	Timergesteuertes Blinken einer LED
BlinkyEnhanced_C++:	Timergesteuertes Blinken einer LED (in C++)
CPUBench:	Misst die Float- und Integer-CPU-Leistung
ESP32AsyncHTTPClient:	Einfaches HTTP-Clientprogramm. (Async HTTP GET/POST)
ESP32BlinkyDeepSleep:	Beispiel (geringer Energiebedarf); blinkt alle 10 Sek.
ESP32FrequencyThrottle:	MCU-Prozessortaktung auf 160, 80, 40, oder 2 MHz herabsetzen
ESP32MQTTClient:	Stellt eine WiFi-Verbindung zu einem MQTT-Broker her
ESP32RadioShuttleMQTT:	RadioShuttle MQTT-Gateway – Empfängt Daten von Knoten und leitet diese an einen MQTT-Broker weiter
HelloWorld:	Einfaches Ausgabebeispiel; erläutert „printf“-Optionen
PropertyEditor:	Programm zum Setzen von Einstellungen (Properties)
PropertyTest:	Properties, z. B. Strings und Nummernwerte lesen/speichern
TimerTest:	Beispiel, welches mehrere asynchrone Timer aufruft
RadioTest:	Einfacher RadioShuttle Server und Client
PMSensorRadio:	LoRa-Umweltsensor; misst periodisch die Staubkonzentration
RTC3231Calibration:	Kalibrierungswerkzeug für die RTC des „ECO Power“-Boards

ESP32: Mehrere Tasks per RTOS

Amazon FreeRTOS

Zweiter Prozessor



ESP32: Mehrere Tasks per RTOS

■ Vorteile

Verschiedene Tasks können die beiden Cores ausnutzen

Expressif nutzt das intern für WiFi usw.

Einfach mehrere Tasks aktivieren, à la `loop2()`, `loop3`, `loop4()`

■ Nachteile

Nicht auf anderen Arduino Plattformen nutzbar (nur ESP32)

Arduino ist nicht multitaskingfähig, Beispiel:

- zwei Tasks nutzen gleichzeitig `Serial.print`-> Crash
- zwei Talks nutzen gleichzeitig die `String()` Klasse -> Crash (später)
- zwei Talks nutzen gleichzeitig GPIO (`pinMode`, `digitalRead`, ...) -> Fehler
- Arduino/C/C++ ist nicht „thread-safe“, daher vorsichtig verwenden

Permanentspeicher für Einstellungen

■ **OTP-Speicher**

kann nur einmalig programmiert werden

■ **Flash-Speicher**

speichert Daten, die erhalten bleiben sollen (non-volatile)

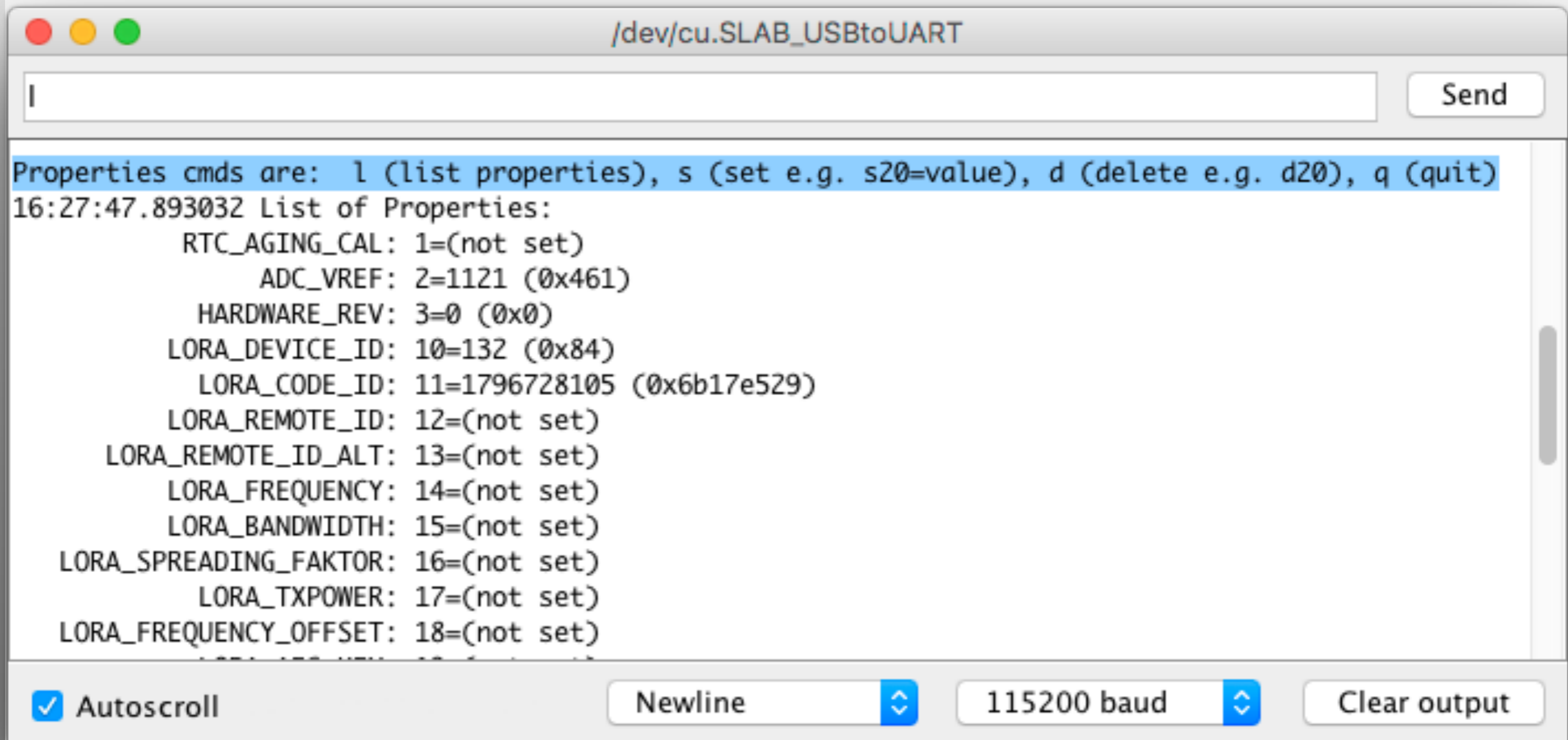
■ **RAM-Speicher**

hier können Property-Daten, während das Programm läuft, gespeichert werden

Das ist interessant für die WiFi-SSID, Kennwort, MQTT-Server-URL, ADC_VREF, LoRa-Server-ID usw. Einmal gesetzt bleibt es erhalten.

Properties

Arduino Property Editor



The screenshot shows the Arduino Property Editor window. The title bar indicates the connection path: `/dev/cu.SLAB_USBtoUART`. A text input field is present with a "Send" button. The main area displays the following text:

```
Properties cmds are: l (list properties), s (set e.g. s20=value), d (delete e.g. d20), q (quit)
16:27:47.893032 List of Properties:
    RTC_AGING_CAL: 1=(not set)
    ADC_VREF: 2=1121 (0x461)
    HARDWARE_REV: 3=0 (0x0)
    LORA_DEVICE_ID: 10=132 (0x84)
    LORA_CODE_ID: 11=1796728105 (0x6b17e529)
    LORA_REMOTE_ID: 12=(not set)
    LORA_REMOTE_ID_ALT: 13=(not set)
    LORA_FREQUENCY: 14=(not set)
    LORA_BANDWIDTH: 15=(not set)
    LORA_SPREADING_FAKTOR: 16=(not set)
    LORA_TXPOWER: 17=(not set)
    LORA_FREQUENCY_OFFSET: 18=(not set)
```

At the bottom, there are several controls: a checked "Autoscroll" checkbox, a "Newline" button with a dropdown arrow, a "115200 baud" button with a dropdown arrow, and a "Clear output" button.

Aufgaben – Allgemein

■ **Blinky laufen lassen**

Vor dem Programmieren: User-Taste halten, einmal “Reset” drücken

Nach dem Programmieren: Reset drücken

■ **BlinkyEnhanced**

Erst mal laufen lassen, erste LED blinkt jede Sekunde

Zweite LED dazu programmieren, soll 5 mal pro Sekunde blinken

■ **HelloWorld**

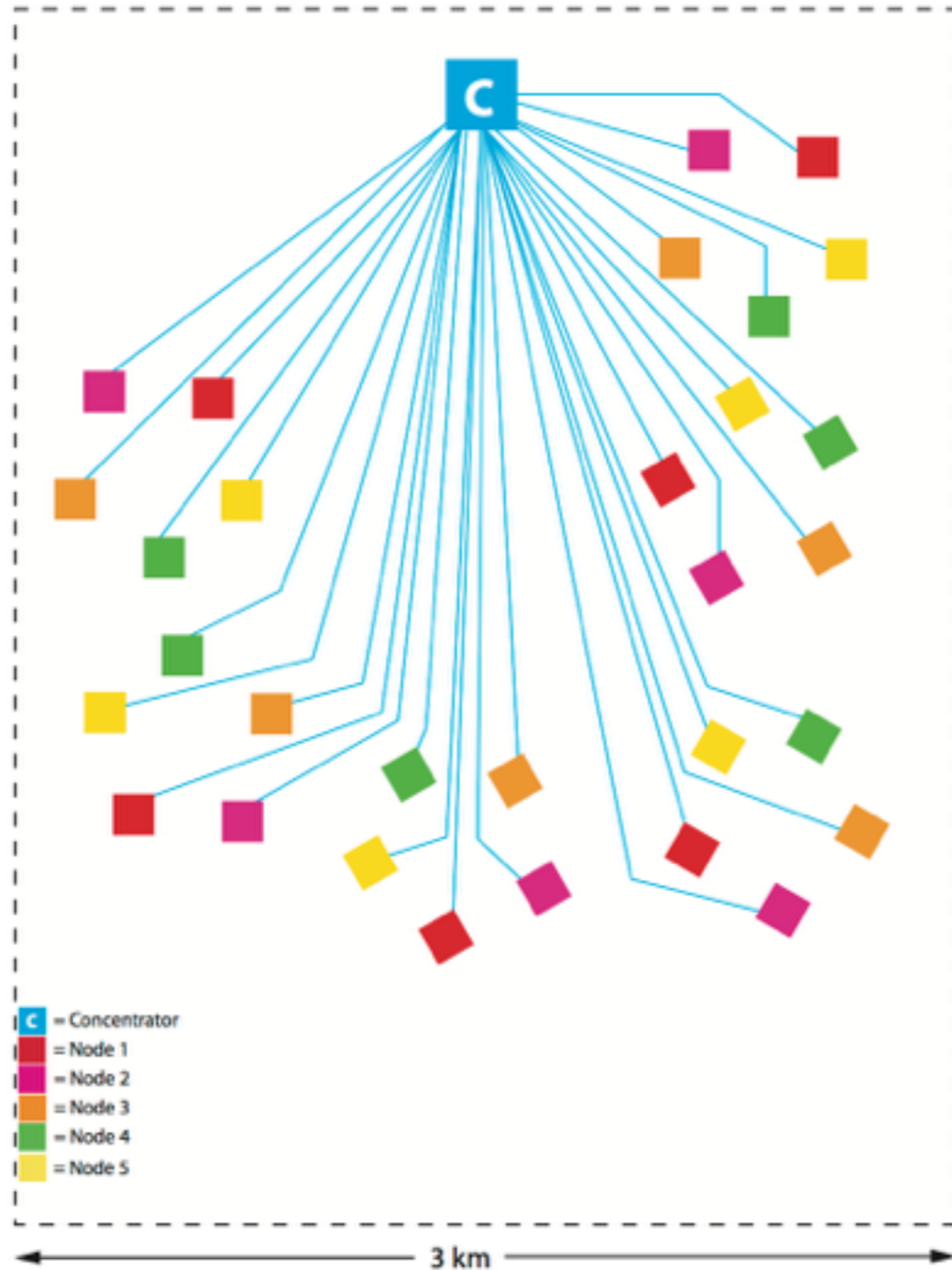
Ausgabebeispiele von „dprintf“ ansehen und verstehen

■ **PropertyEditor**

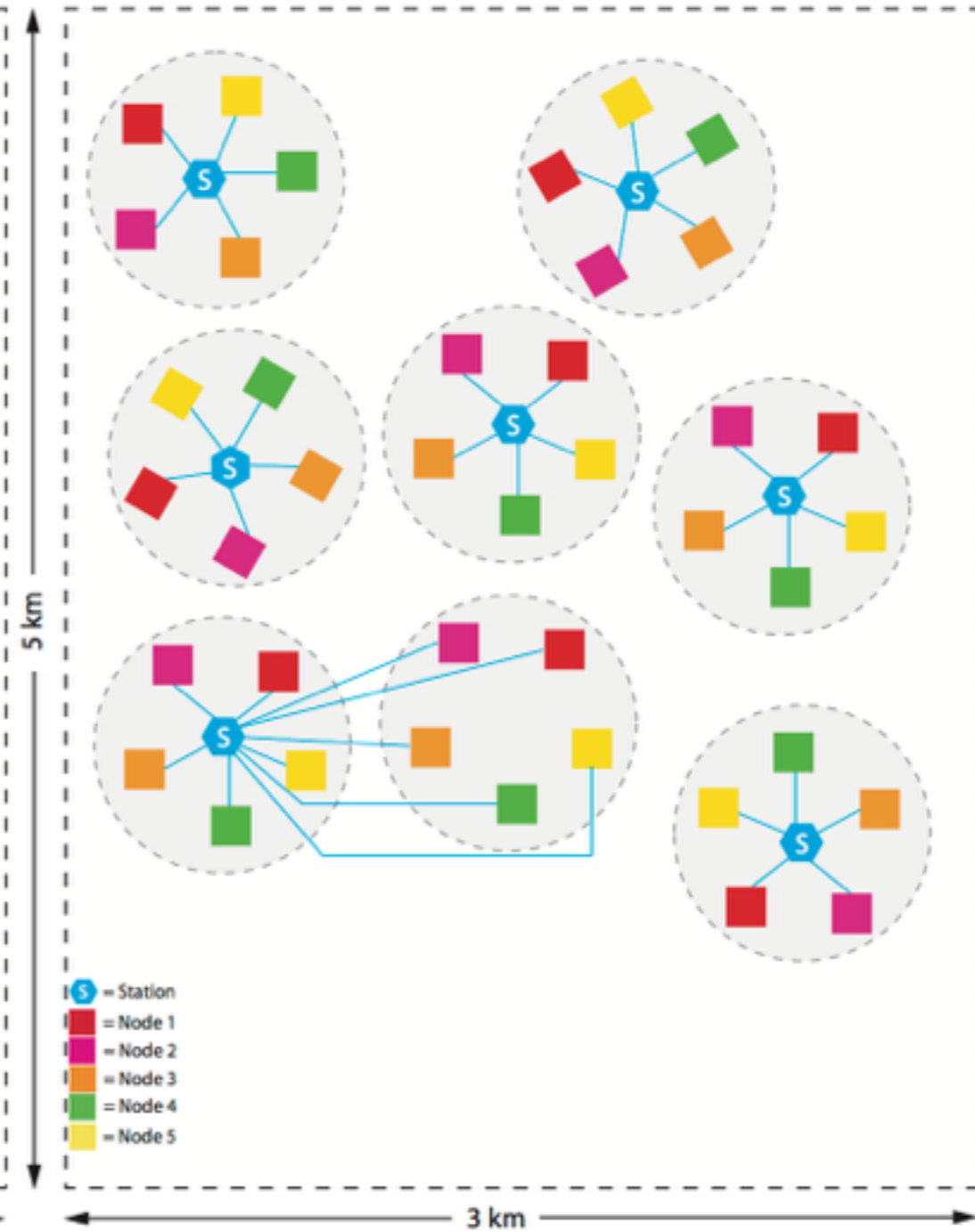
WiFi einstellen: SSID: Computerwurgstaat Kennwort: no_na...

■ **ESP32AsyncHTTPClient**

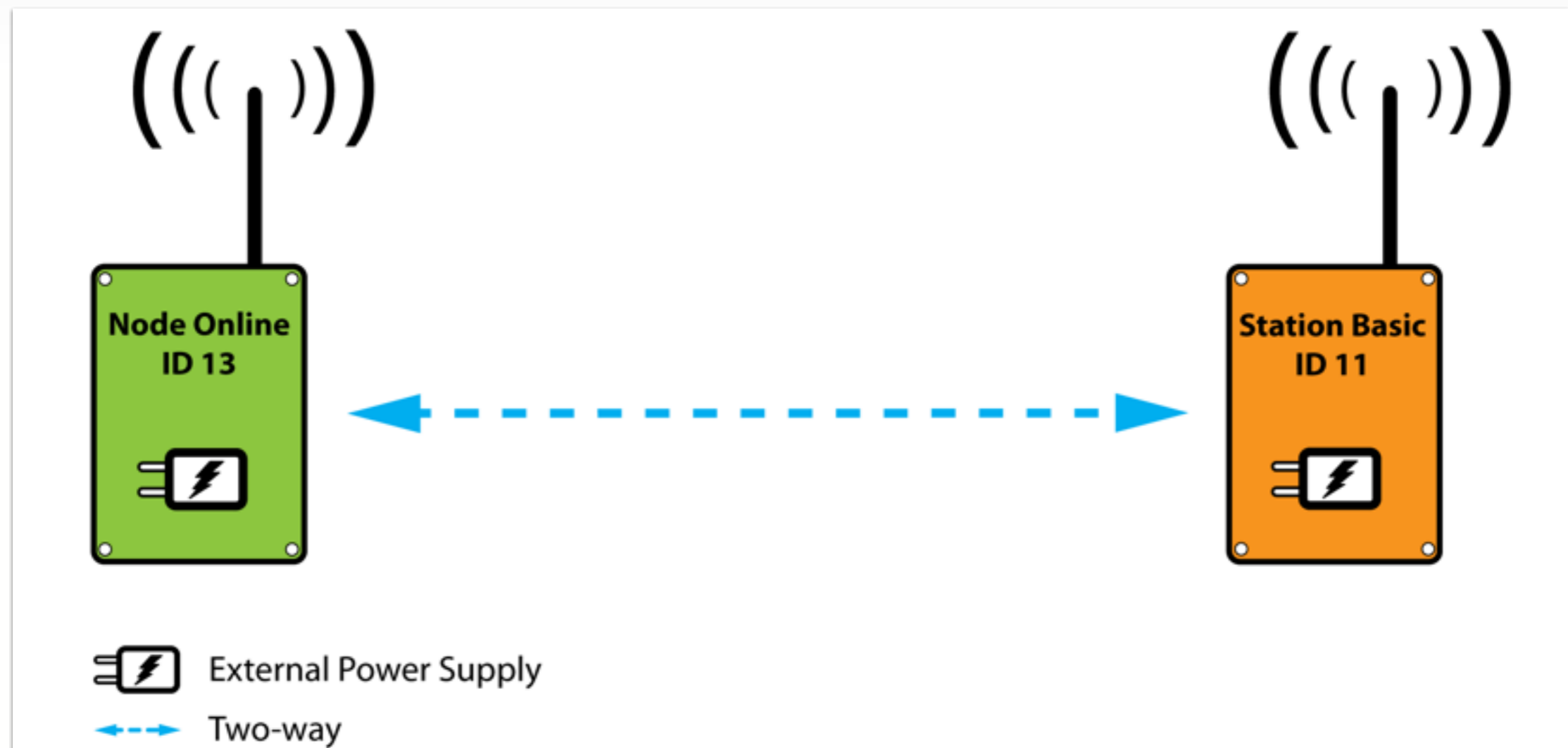
LoRaWAN™



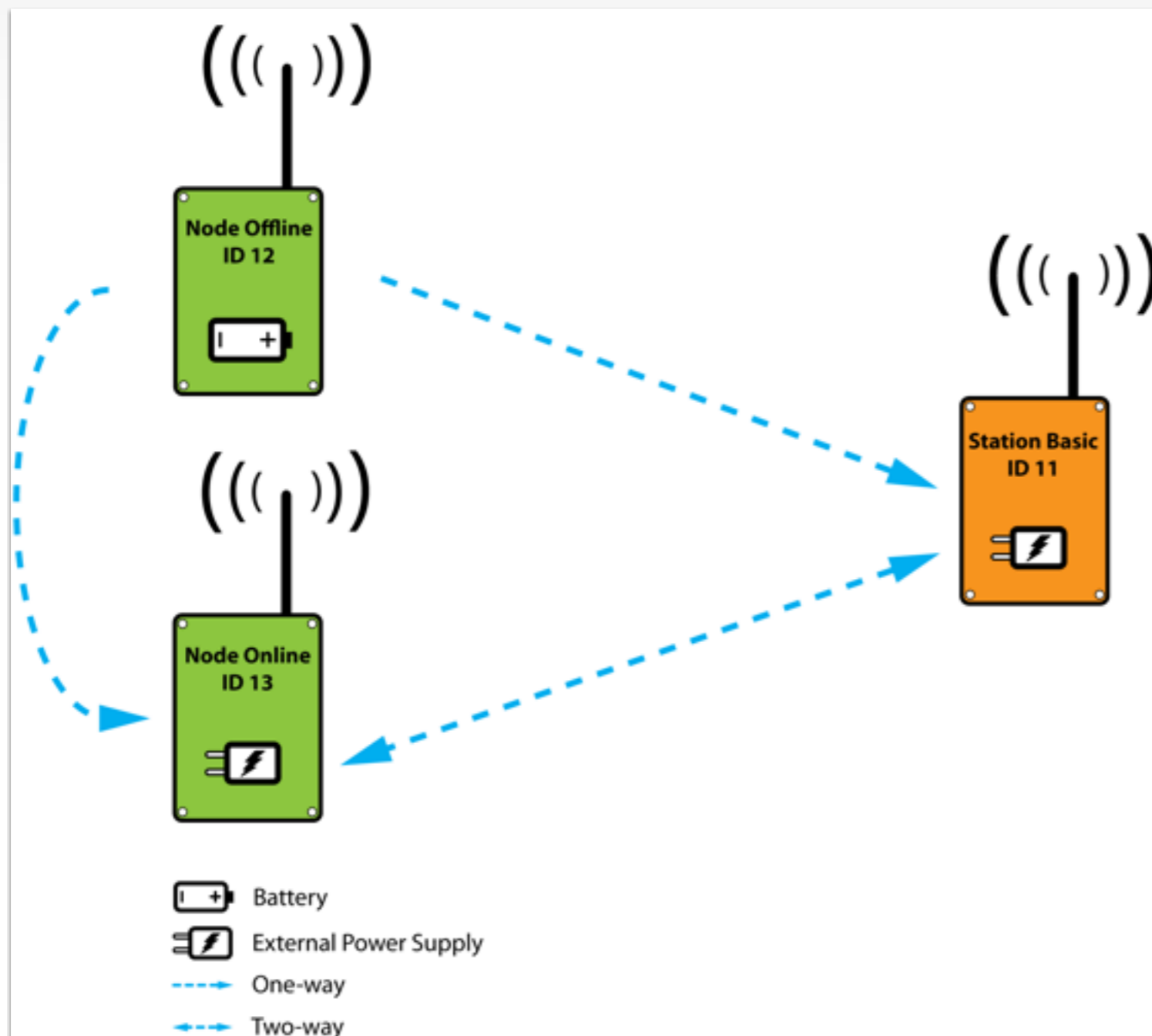
RadioShuttle



Einfaches Netz mit Station und Knoten



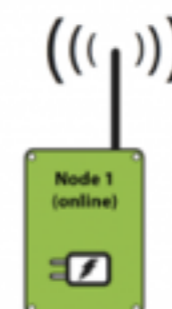
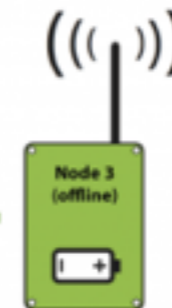
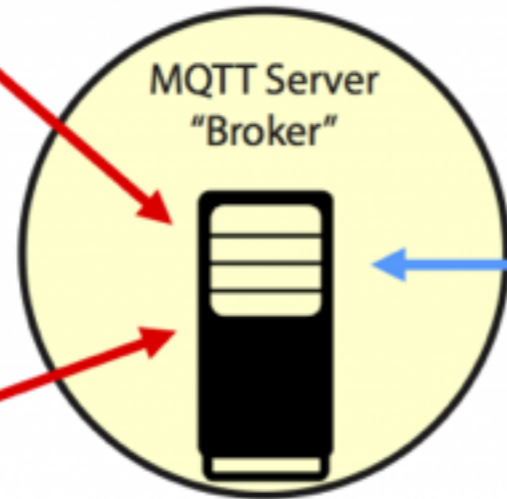
Netz mit Station und 2 Knoten (online/offline)







MQTT Gateway

MQTT Internet

LoRa RadioShuttle



-  Two-way MQTT
-  Two-way MQTT / LoRa Gateway
-  One-way LoRa
-  Two-way LoRa

Aufgaben – ECO Power RadioShuttle

- **Beispiel „RadioShuttle->RadioTest“**

Zeile: „#define USE_DEMOBOARD_PAIR“ auskommentieren

- **Programm laufen lassen**

- **User-Taster drücken**

Sollte jetzt funktionieren

- **Der Server hat die ID: 1 und steht bei mir!**

Erklärung der Beispiele

Aufgaben – MQTT

■ **ESP32MQTTClient**

MQTT-Broker:

Per TCP/IP ohne Verschlüsselung

<mqtt://loratest:test1234@mqtt.arduino-hannover.de:1883>

Per TCP/IP mit SSL-Verschlüsselung

<mqtts://loratest:test1234@mqtt.arduino-hannover.de:8883>

■ **ESP32RadioShuttleMQTT**

Beispiel: RadioShuttle - MQTT-Gateway



Vielen Dank!