Arduino und Amateurfunk (Fortsetzung)

Workshop USKA-Sektion Solothurn am 2.Sept 2023 .

Thema: Mit ESP32
oder ESP8266 und
MQTT Daten
übertragen

HB9BFD Roland

Einleitung

- Unterrichtsziele:

  « …. nötige Kompetenz für die Programmierung von ESP's erwerben. Programme vom Internet anpassen und verwenden können.»

  « ….ESP8266 und/oder ESP32 programmieren können.»

Wieso ESPs von der Firma Expressif?

- Günstig (China –Clone für Fr. 2.- erhältlich)
- Grosse Community im Internet
- Viele Programme und Libraries (Bibliotheken)
- Grosse Anzahl von Sensoren und Shields
- Schnelle Realisierung von Ideen
- Leicht erlernbare Programmiersprache C++
- Viele Projekte bereit zum Anpassen auf meine Bedürfnisse

Inhalt der Präsentation

- Arduino Grundlagen
  - Unterschiede SPS – MikroController
  - Funktion, Hardware
  - Software
  - Grundlagen der Programmierung
  - Übungsbeispiel Blinklampe (bis hierher im 1.Kurs 2021!)

- Bussysteme mit Arduino
  - Verwendung von Bibliotheken
  - «Bus Systeme» mit Arduino: Seriell, USB, I^2C, 1-Wire
  - Übungsbeispiel LCD Anzeige mit Arduino

Inhalt der Präsentation

- Sensorik
  - Sensorik mit Arduino
  - Temperaturmessung
  - Übungsbeispiel Dallas Chip
- ESP32 oder ESP8266
  - Vorbereitungen in IDE, Unterschiede,
- Kommunikation
  - Kommunikation mit MQTT Broker
  - Übermitteln von Messwerten an MQTT Broker
- Anwendungs-Beispiele im Amateurfunk

Arduino Grundlagen

Arduino Grundlagen

- Inhalt:
  - Unterschiede SPS – MikroController (Arduino HW)
  - Funktion, Hardware
  - Software
  - Grundlagen der Programmierung
  - Übungsbeispiel Blinklampe

Arduino Grundlagen

- Lernziel:
  - Du bist in der Lage zu entscheiden, welche Lösung für dein Problem am meisten Sinn macht (Kosten, Performance, Sicherheit, Ersatzteile, Dokumentierbarkeit, Lifetime, ….)
  - Umgang mit den Arduino Beispielen vom Internet und der Hardware (Aliexpress, Bangood, DealExtreme,…..).
  - Blinklampe programmiert mit dem ESP MCU-Board
  - Daten an MQTT-Broker senden

Funktion, Hardware (Arduinos)

- Open-Source Soft- und Hardware
- Einfaches E/A Board meist basierend auf Atmel-AVR oder ESPs von der Firma Espressif
- Spannungsversorgung meist via USB
- Intern entweder 5VDC oder 3.3VDC
  Vorsicht ESP's vertragen keine 5V an den Eingängen!!
- Programmierung über USB (nackte Chips/Boards via RS232 und/oder USB-Wandler)
- Shields sind sogenannte Erweiterungen, die direkt aufgesteckt werden.

Software Arduino IDE

- Integrierter Bootloader: Programmierung direkt über serielle Schnittstelle, respektive USB möglich
- Integrierte Entwicklungsumgebung (IDE), kostenlos
- Läuft auf Windows, Linux und macOS (bei mir leider nicht!)
- Arduino-IDE hat Code Editor und Compiler eingebunden
- Zusätzliche Bibliotheken vereinfachen die Programmierung in C und C++ stark

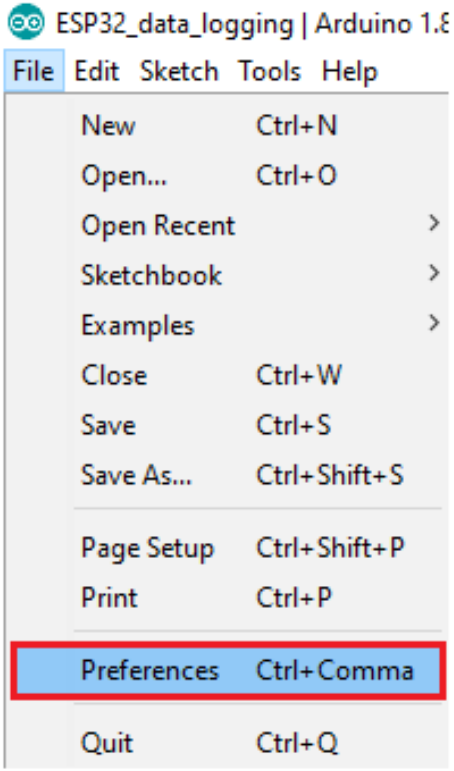
Der **Bootloader** ist ein kleines Programm, welches dem Atmel ATMega328P-pu sagt, dass er ein **Arduino** (genauer gesagt ein **Arduino**-Uno) werden soll. ... Der **Bootloader** sorgt auch dafür, dass der Atmel ATMega328P-pu nach dem Einschalten prüft ob ein Programm auf ihm gespeichert ist.

Software Arduino IDE: Programmiersoftware

- Die IDE (**I**ntegrated **D**evelopment **E**nvironment) habt ihr bereits zuhause auf eurem PC installiert! (?)
- Sonst kann sie von: www.arduino.cc geladen werden. Ich verwende die englische Version (1.8.57.0), was immer das bedeutet....Ihr könnt problemlos auch die deutsche Version nutzen oder die Version 2.2.
- Bei mir hat die IDE Probleme gemacht, wenn ich die Programme (=Sketches) auf dem Netzwerk speichern wollte, also verwende ich den C:-Drive.

Software Arduino: ESP32/ESP8266 Add-on

- Arduino IDE, **File> Preferences**

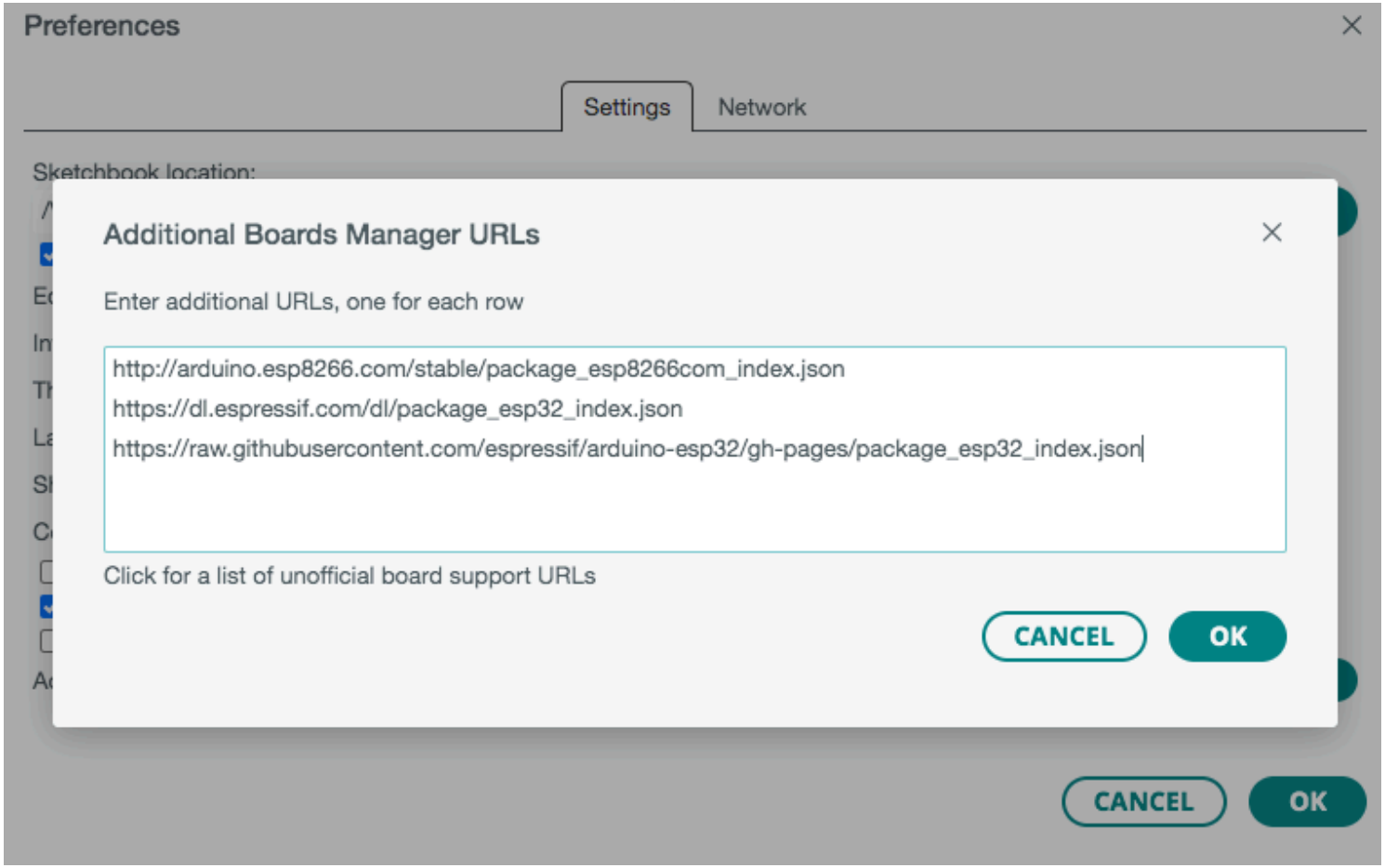


Sieht in Version 2 etwas anders aus!

- Für ESP32 und/oder ESP8266 diese Zeile einfügen:
- http://arduino.esp8266.com/stable/package_esp8266com_index.json, https://dl.espressif.com/dl/package_esp32_index.json
- Damit sollten unter >>Tools>>BoardsManager neue Einträge für ESP-Boards erscheinen

Software Arduino IDE: Präferenzen einstellen

- Für Espressif-Chips unbedingt in Preferences zusätzliche URLs einfügen…. Bedeutet online sein !!



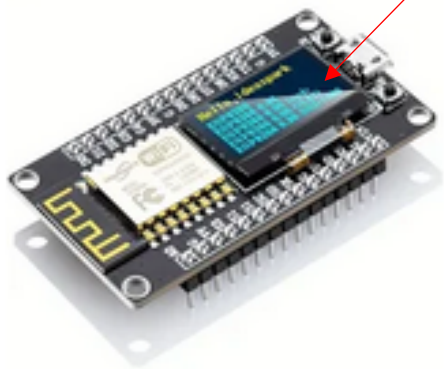
Software Arduino IDE : Board auswählen

- Hier hatte ich am meisten Mühe, das richtige Board zu finden, denn ich kaufte Chinaschrott, meist ohne weitere Bezeichnungen der Boards.
- Wir verwenden ESP8266-12E NodeMCU V1.0

Software Arduino IDE : Board auswählen

Mit externer Antennenbuchse

Mit OLED Display



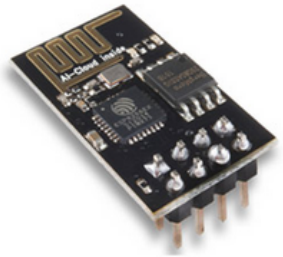
Praktisch
nackter Chip

ESPs

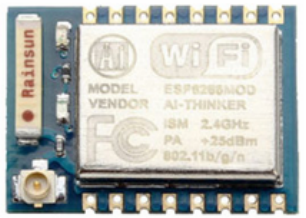
DOIT DEVKIT V1

ESP32 DevKit

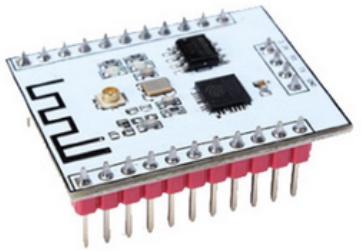
ESP-32S NodeMCU



ESP-01



ESP-07



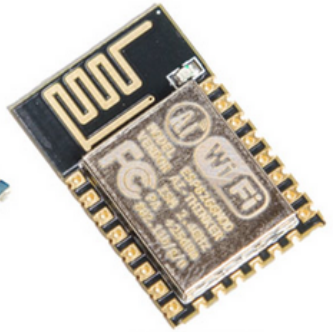
ESP201

WEMOS LOLIN32

"WeMos" OLED

HUZZAH32

ESP-12



ESP-12E

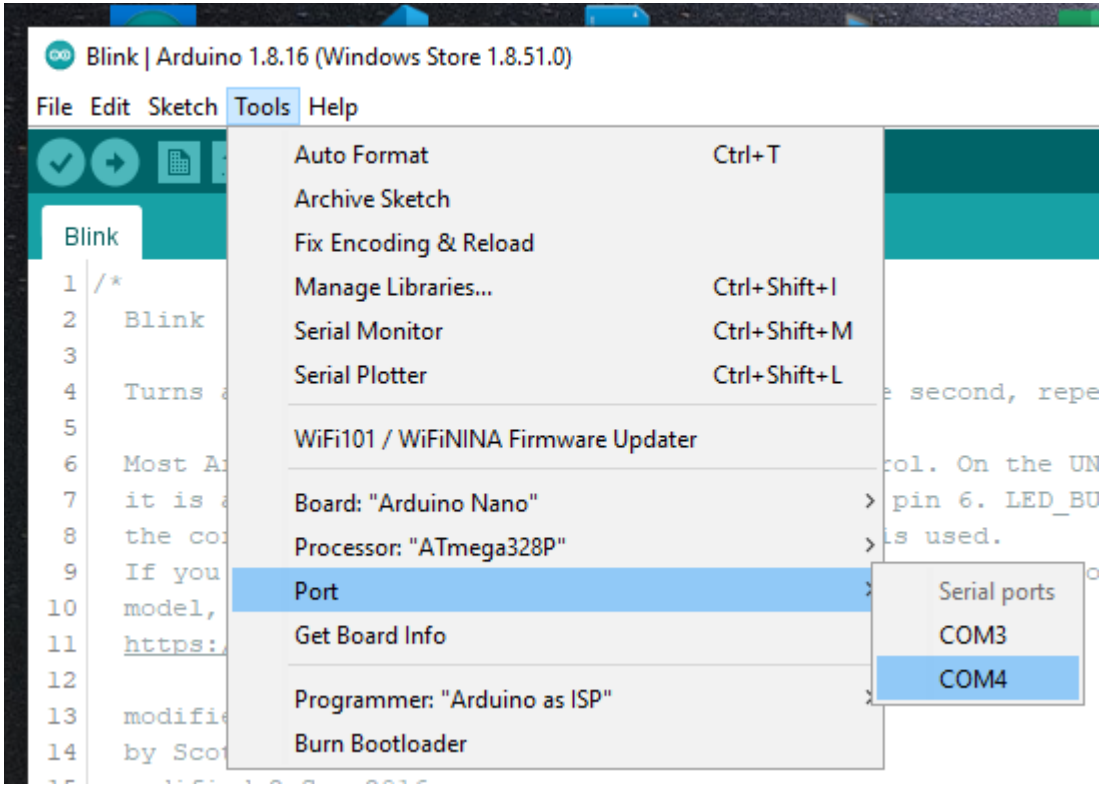
ESP-12F

Welchen Typ ESP soll ich verwenden?

Software Arduino IDE: Treiber für USB/Seriell-Chip

- Oftmals fehlt der entsprechende Treiber für die USB-Schnittstelle, sodass die Software keinen seriellen (COM) Port sieht.
- CH3406, 341 oder FTDI 210x oder….googeln und hoffentlich ohne einen Virus zu laden
- Wichtig: Es gibt USB-Kabel, die nur zum Laden geeignet sind, aber nicht zum Daten übertragen. Falls ihr so ein Kabel erwischt, so habt ihr Pech und es wird in der IDE Software eventuell kein Port angezeigt.

Software Arduino IDE: Programmier-Ports



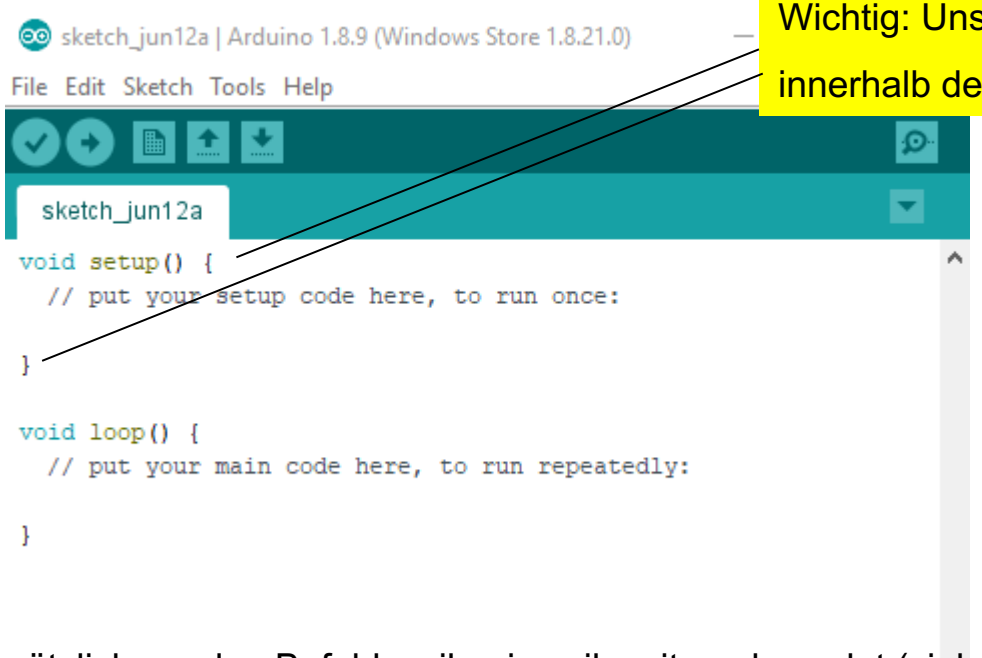
Port einstellen!

Falls es nicht Ping macht beim Anschliessen des Arduinos, dann fehlt vielleicht der Treiber für den USB Chip auf dem Board und die Schnittstelle wird nicht als USB/Com erkannt. Abhilfe→ richtigen Treiber im Internet dazu suchen und installieren.

CH340 Treiber für Windows: **https://download.bastelgarage.ch/Produkte/CH341SER.ZIP**

Software Arduino DIE. Grundgerüst für Programm

- Wenn man Arduino IDE aufruft, dann kriegt man zuerst ein Grundgerüst für die Programmierung (oder das zuletzt bearbeitete Programm erscheint).



Wichtig: Unsere Befehle werden

innerhalb der Klammern {} geschrieben!

Zusätzlich werden Befehlszeilen jeweils mit «;» beendet (siehe folgendes Beispiel)! Vergiss ich selbst immer wieder!

Software Arduino IDE

- Für die Programmierung kennt man grundsätzlich zwei Strukturfunktionen:
  - **setup()** wird beim Start des Programmes einmalig aufgerufen
  - **loop()** wird kontinuierlich durchlaufen, ähnlich wie bei einer SPS (bei Siemens typischerweise OB1).
  - Zusätzlich **definiert** man am Anfang des Programmes die Ein-Ausgänge, Konstante Werte, Instruktionen für den Compiler etc und ruft hier Bibliotheken auf.

Software Arduino IDE: Syntax

- ; damit werden einzelne Instruktionen getrennt
- {} unterscheiden der Programmabschnitte
- // für einzeiligen Kommentar
- /*……*/ für mehrzeiligen Kommentar
- () übergeben von Parametern bei Funktionen
- #define zum Definieren von Anfangswerten
- #include zum Einbinden von Bibliotheken

Grundbefehle der Programmierung (Functions)

- **Digital I/O**
  - digitalRead()
    digitalWrite()
    pinMode()
- **Analog I/O**
  - analogRead()
    analogReference()
    analogWrite()
- **Zero, Due & MKR Family**
  - analogReadResolution()
    analogWriteResolution()
- **Time**
  - delay()
    delayMicroseconds()
    micros()
    millis()
- ….
- ….. see Internet
- **Communication**
  - Serial
    Stream

Grundlagen der Programmierung

- Arduino Datentypen und Konstanten

Constants

Floating Point Constants
Integer Constants
HIGH | LOW
INPUT | OUTPUT | INPUT_PULLUP
LED_BUILTIN
true | false

Conversion

byte()
char()
float()
int()
long()
word()

Data Types

String()
array
bool
boolean
byte
char
double
float
int
long
short
size_t
string
unsigned char
unsigned int
unsigned long
void
word

Variable Scope & Qualifiers

const
scope
static
volatile

Utilities

PROGMEM
sizeof()

Grundlagen der Programmierung

- Variablen
  - Eine Variable ist ein kleiner Speicher, in den eine Information einer bestimmten Form passt. Die Form wird durch den sogenannten Variablentyp bestimmt.

Variablentyp	Bedeutung	Beschreibung
int	ganze Zahlen	-32'768 bis 32'767
long	ganze Zahlen	-2'147'483'648 bis 2'147'483'647
float	Fließkommazahl	gebrochene Zahlen
char	Character	Alphanumerische Zeichen (Buchstaben, Zahlen, Sonderzeichen)

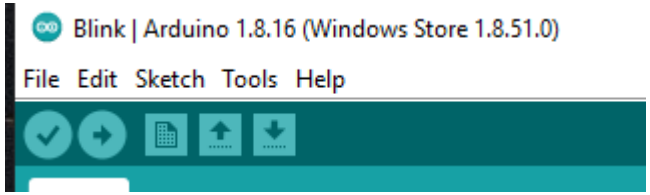
Übung: Blinkende LED Was braucht ihr dazu?

ESP Platine (ESP32 oder ESP8266)

Betriebsspannung : 5 V via **Mikro-USB,** intern wird daraus 3.3V gemacht**,**
Vorsicht: die Eingänge vertragen nur 3.3V

USB-Programmier-Kabel zum PC, Software IDE auf PC

Auf dem PC ist IDE geladen



Übung: Blinkende LED

Dieses Board hat, wie die meisten Boards eine eingebaute LED. Diese wird mit LED_BUILTIN angesprochen

Übung: Blinkende LED

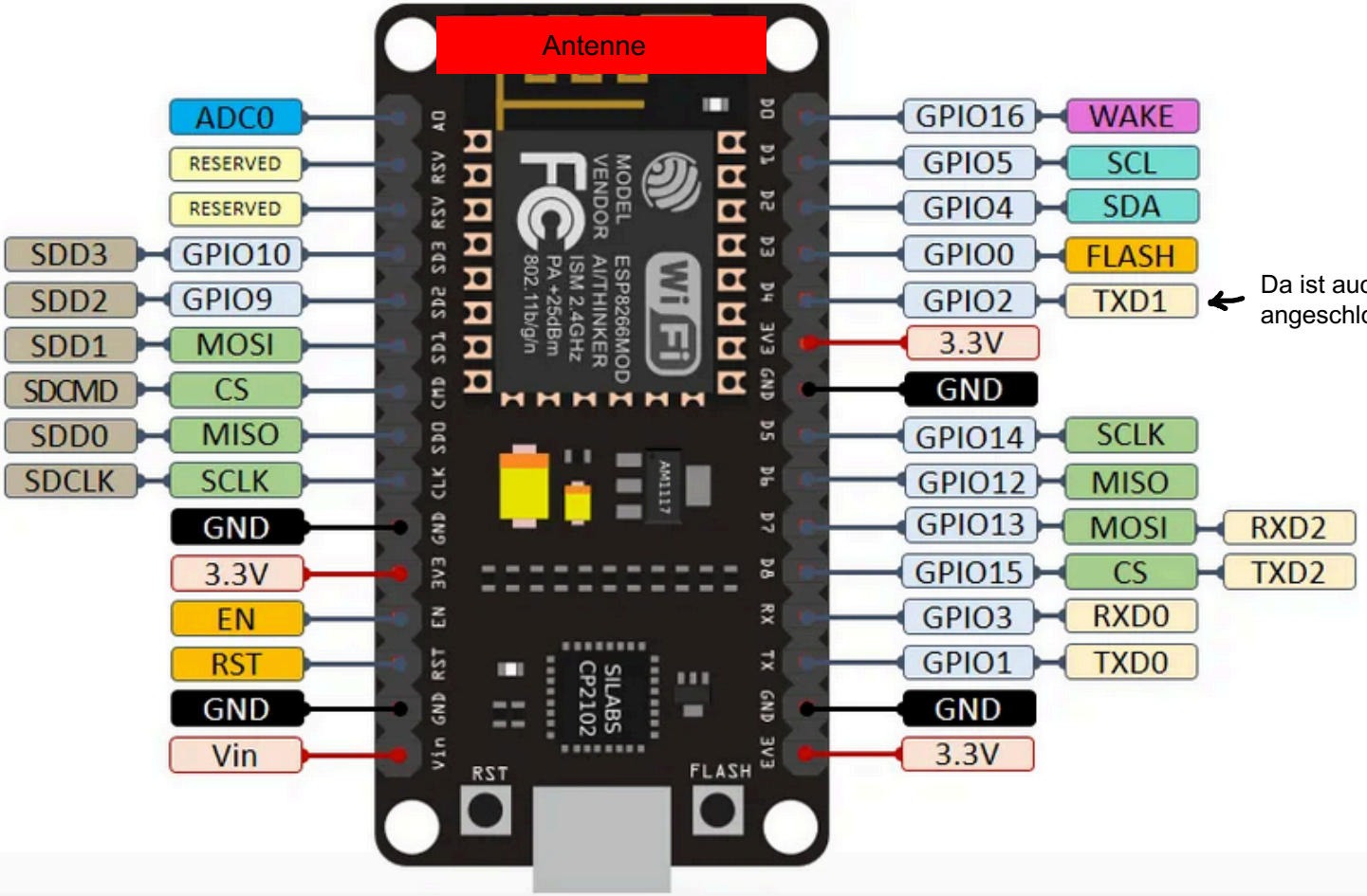
- Mit diesem Beispiel-Programm wird die interne LED zum Ausgang deklariert und nachher jeweils mit HIGH und LOW beschaltet!

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

LED_BUILTIN = GPIO2 = D4

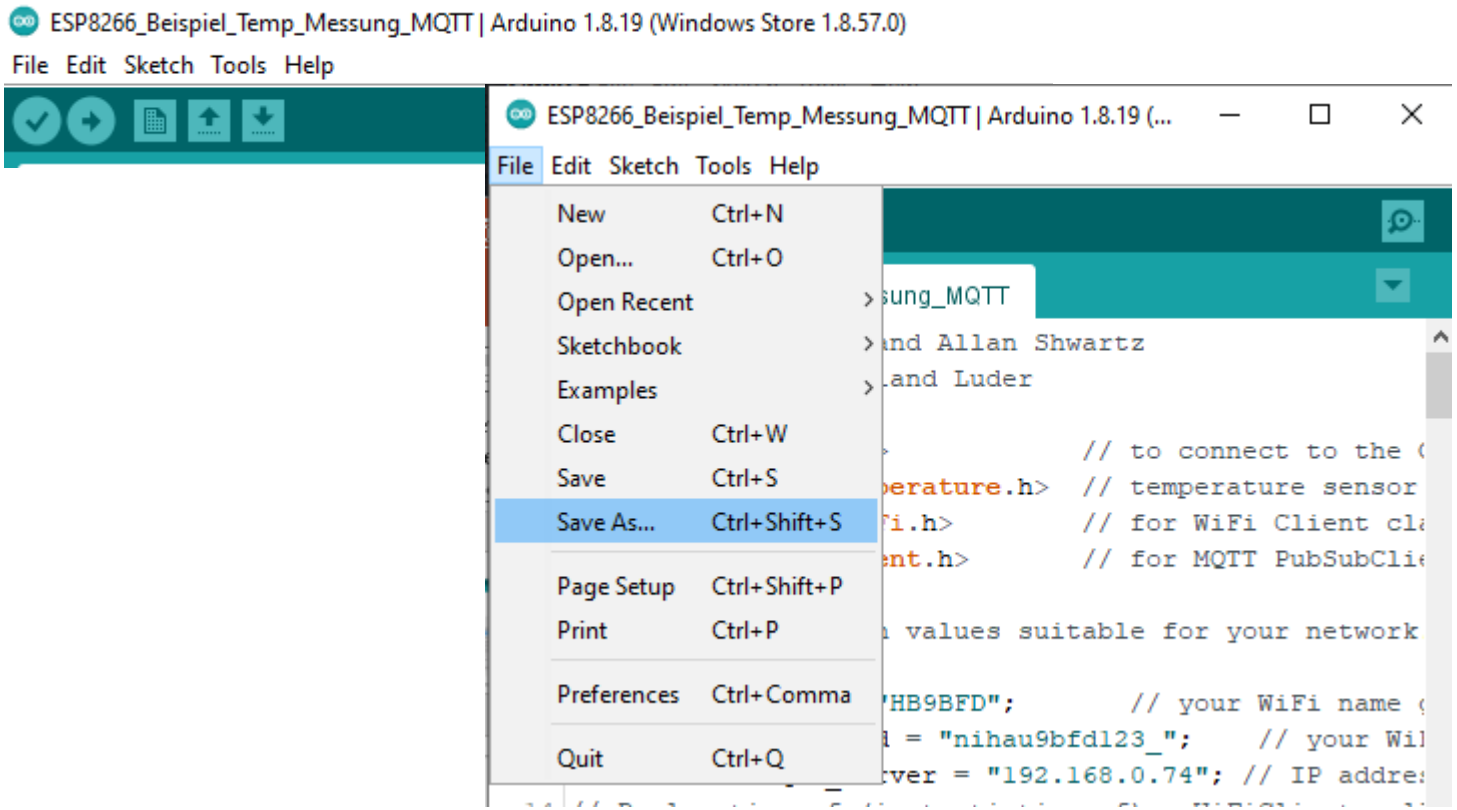
Grundsätzlich kann jeder Digital-Pin als Ein- oder Ausgang verwendet werden. Ein paar Pins sind jedoch reserviert für die Programmierung (RX/TX ….etc).

Übung: Blinkende LED: Wie heissen die Pins?



Programmiert werden die Pins über die GPIO-Nummer und nicht via Bezeichnung auf der Platine, denkt daran! Kann je nach Programmumgebung jedoch anders sein.

IDE Programmiersoftware



Nicht vergessen, das Programm unter einem Namen zu speichern mit "save as". Nachher genügt "save".

Übung: Blinkende LED

- Da ich den Ausgang 13 (D7) als LED benutzen möchte, kann ich ihn auch als Variable ersetzen:
  - int ledPin=13;
- Nun kann ich im Programm "ledPin" verwenden.
- Die Variablenzuweisung erfolgt vor "void setup" und ist nachher überall gültig.
- Muss nun natürlich eine externe LED mit Vorwiderstand 470 OHM gegen Masse an D7 anschliessen.

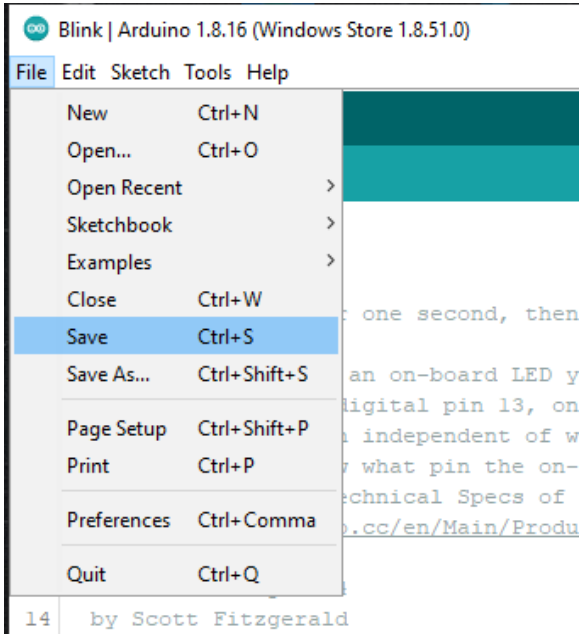
Übung: Blinkende LED

- LED_BUILTIN ersetzen durch Pin 13

```
int led_Pin 13;
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(led_Pin, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(led_Pin, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(led_Pin, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

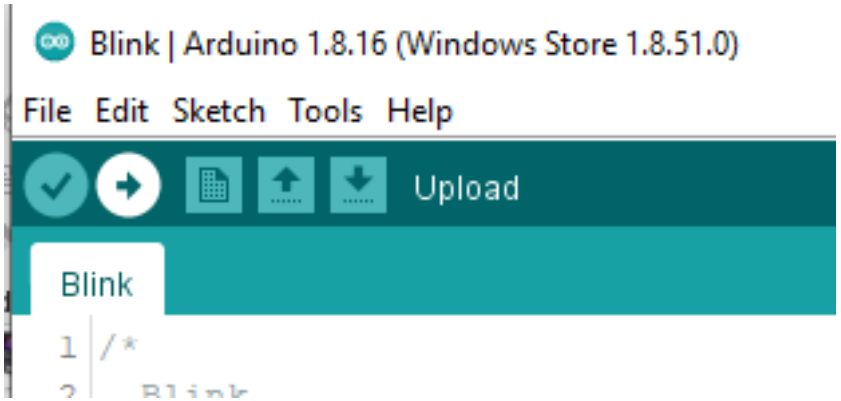
Übung: Blinkende LED

- Programm eintippen, abspeichern (ev. anschliessend Arduino IDE schliessen und Programm "Blinken" wieder laden.



- Du kannst mit Copy/Paste arbeiten um Programme zu editieren! → einfach probieren!

Übung: Blinkende LED (weitere Versuche)



Wenn man "Upload" anklickt, so wird das Programm übersetzt (compiliert) und wenn alles gut ist, auf den Chip geladen….man sieht dies an den flakernden LED's.

Übung: Blinkende LED (weitere Versuche)

- Bitte die Ein-/Ausschaltzeiten ändern und Programm wieder übersetzen und auf Arduino laden.

ESP8266 oder ESP32

ESP8266

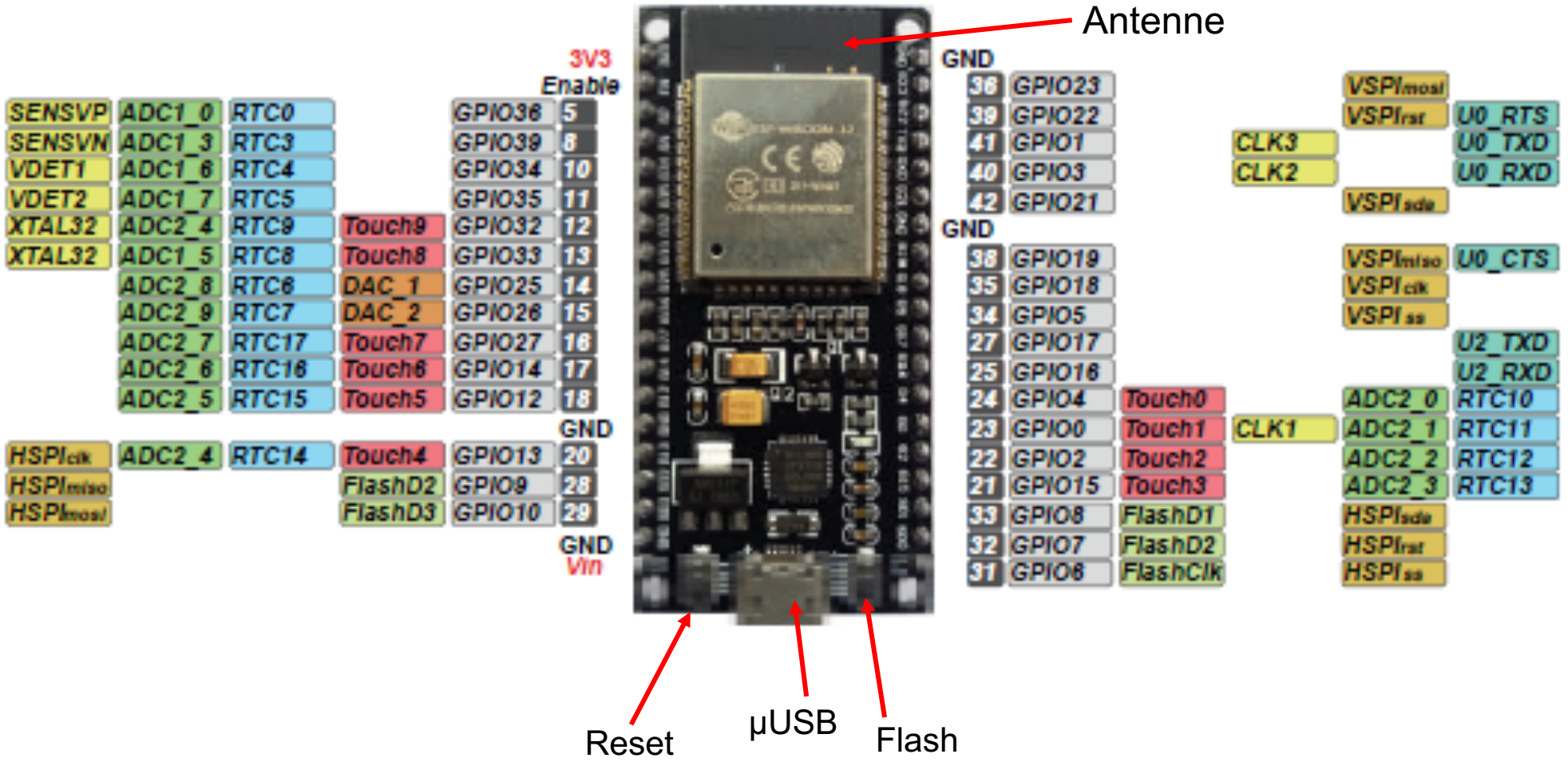
- 1 Kern
- 80 Mhz
- 160 KB
- 17 GPIOs
- 2 SPI/1 I2C
- 1 ADC
- 10 Bit
- 80...170 mA (im Deep Sleep nur 0.5uA)

ESP32

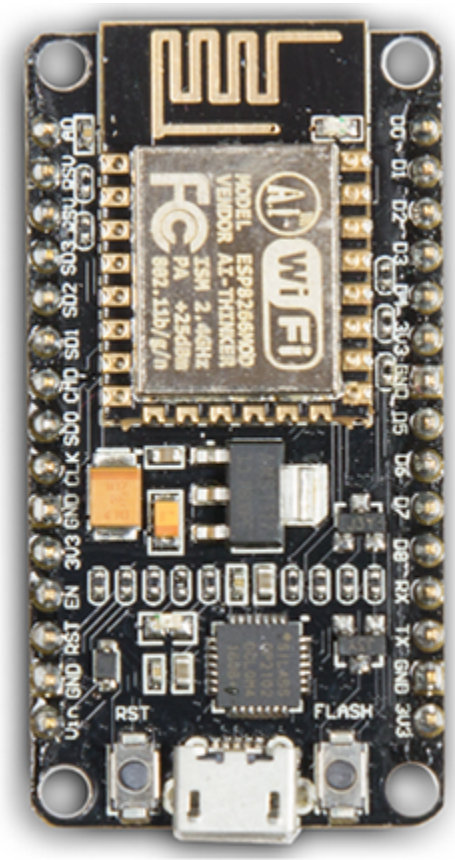
- 2 Kerne
- 160 Mhz
- 520 KB
- 36 GPIOs
- 4 SPI / 2 I2C
- 16 ADC
- 12 Bit
- 260 mA (WLAN) ! Ein Stromfresser also!
- BlueTooth, ESP32-ESP32

Wichtig: 3.3V Ein-/Ausgänge, WLAN integriert,
kann mit OTA (=**O**ver **T**he **A**ir) programmiert werden!!!

ESP32 Board



ESP8266 Board



ESP Boards

- Wozu sind die Buttons RST und Flash?
- Zum Laden von Programmen steckst du den NodeMCU an den PC an. Anschliessend musst du beide Taster auf dem Board drücken (RST und FLASH). Dann zuerst RST loslassen und anschliessend erst FLASH. Nun solltest du in der Arduino IDE das COM Port anwählen können und das Programm auf den Controller kopieren können.
- **Muss nicht sein**, je nach Modell werden die Buttons durch die IDE-Software automatisch richtig gesteuert !!

Fehlende Bibliotheken in Arduino

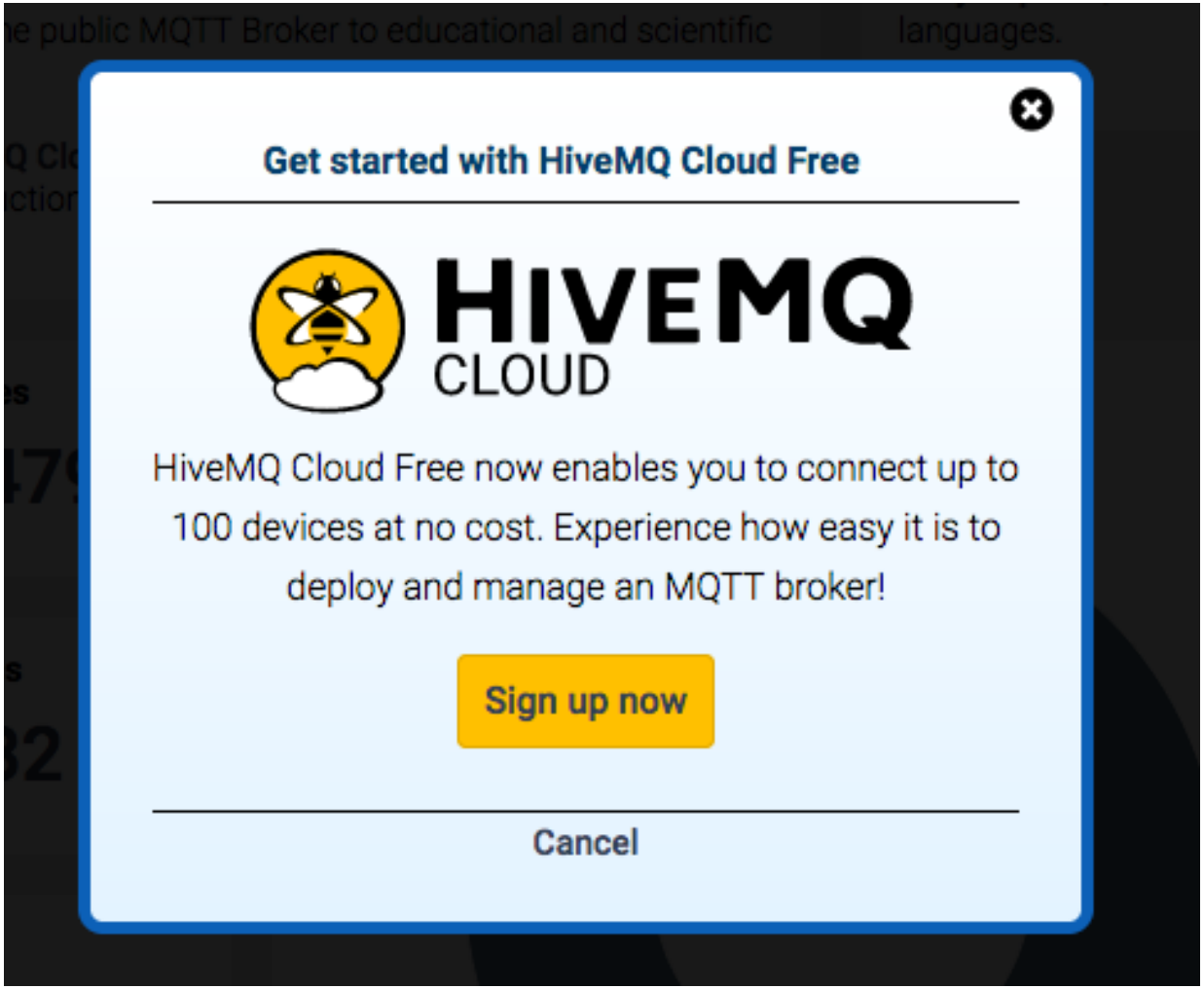
- Arduino hat nur einen Standardsatz von Bibliotheken, fehlende können mit «Manage Library» oder direkt vom Internet als ZIP-Datei runtergeladen werden.
- Dazu wird *Sketch -> Include Library-> Add .ZIP Library* aufgerufen. Die ZIP Datei wird dann automatisch sauber ins Verzeichnis eingebunden und steht nun zur Verfügung.

Kommunikation via Mosquitto Broker

- MQTT

Kommunikation via HIVEMQ Free Cloud Broker

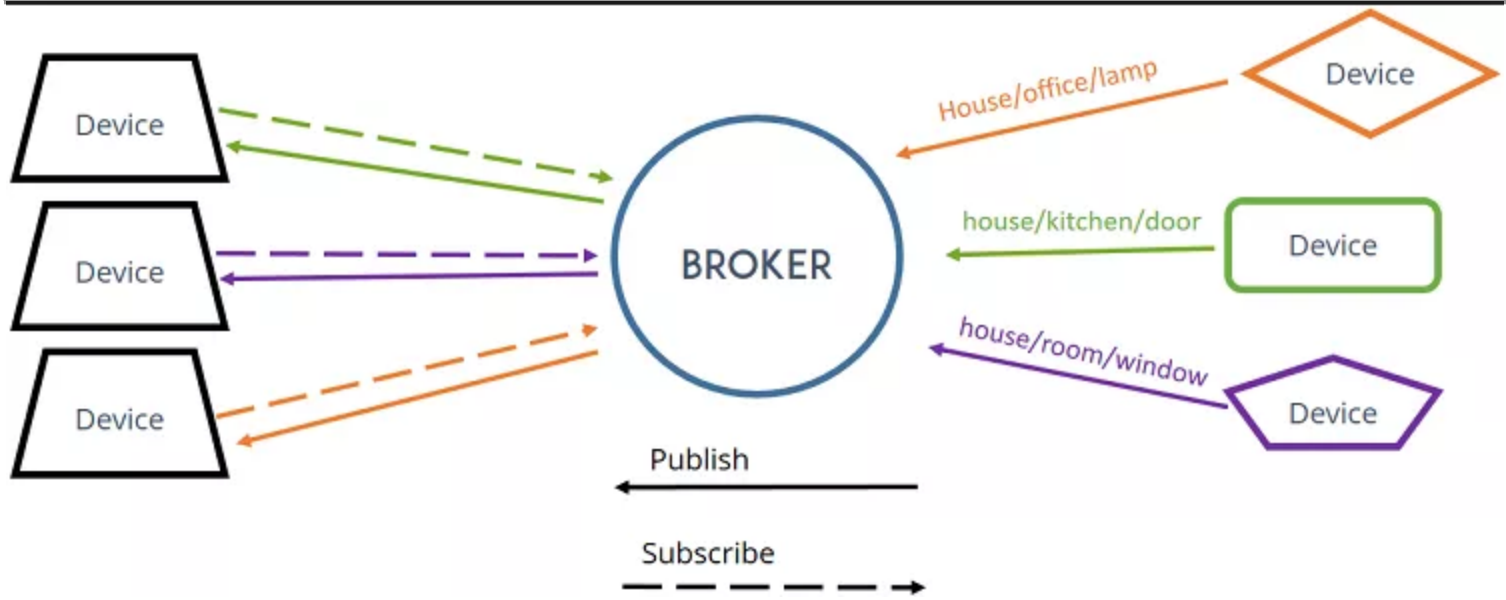
- Wir können dazu HIVEMQ (= Cloud Server) verwenden!



Kommunikation via Mosquitto Broker

- Lernziel:
  - Du erkennst die Möglichkeiten von IoT (Internet of Things).
  - Kannst eigene Daten auf dein Handy übertragen.
  - Bibliotheken runterladen und einbinden
  - WLAN konfigurieren
  - Analogeingang verwenden

Kommunikation via Mosquitto Broker



MQTT (kopiert aus Wikipedia)

Message Queuing Telemetry Transport (MQTT) ist ein offenes Nachrichtenprotokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht, trotz hoher Verzögerungen oder beschränkter Netzwerke.

Entsprechende Geräte reichen von Sensoren und Aktoren, Mobiltelefonen, Eingebetteten Systemen in Fahrzeugen oder Laptops bis zu voll entwickelten Rechnern.

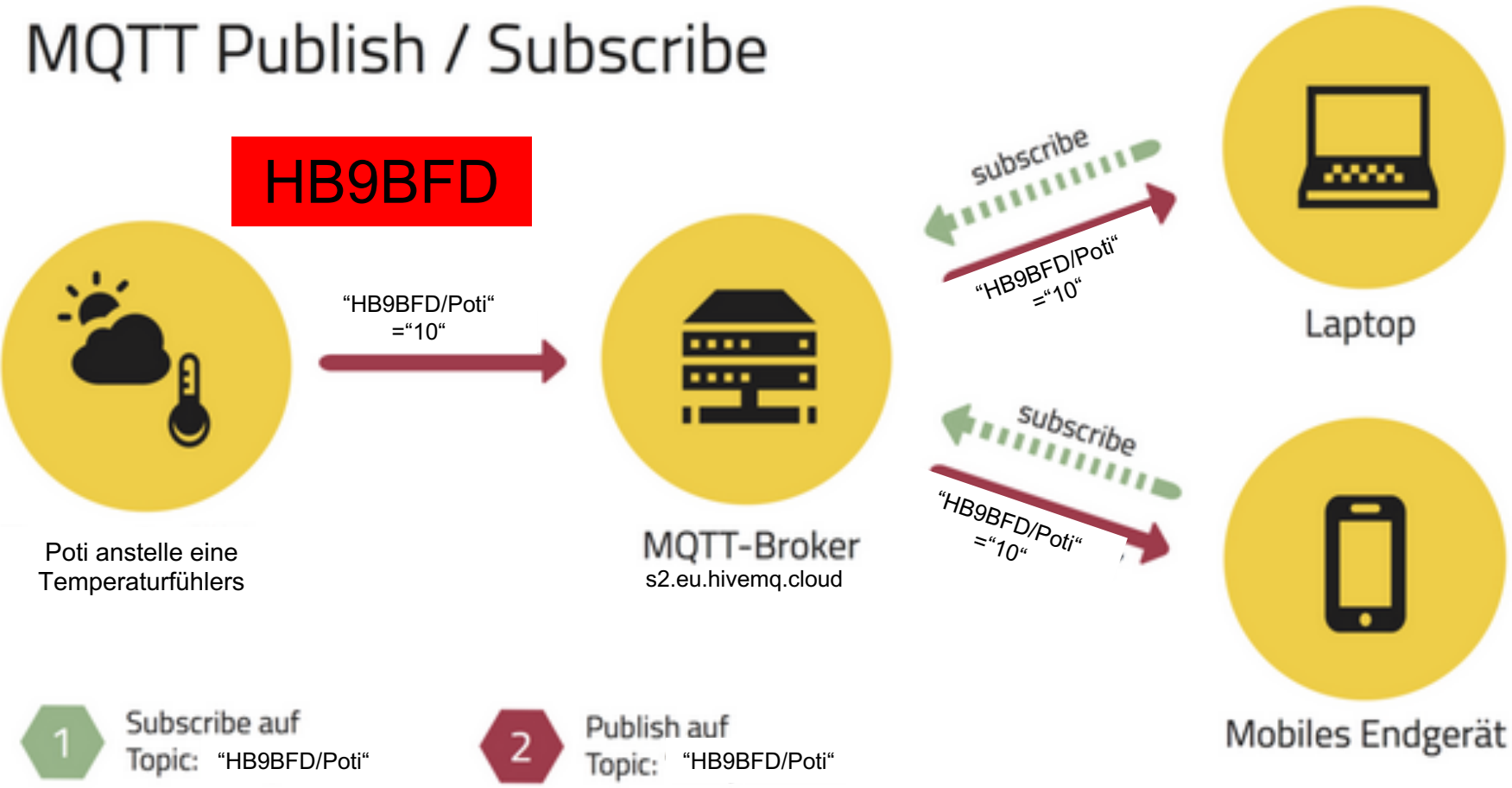
Das MQTT-Protokoll ist auch unter älteren Namen wie „WebSphere MQTT“ (WMQTT), „SCADA-Protokoll“ oder „MQ Integrator SCADA Device Protocol“ (MQIsdp) bekannt.

Die Internet Assigned Numbers Authority (IANA) reserviert für MQTT die Ports 1883 und 8883. MQTT-Nachrichten können mit dem TLS-Protokoll verschlüsselt werden.

MQTT (kopiert aus Wikipedia)

Interessant ist, dass ein MQTT-Server („Broker“) die gesamte Datenlage seiner Kommunikationspartner hält, und so als Zustands-Datenbank benutzt werden kann.
So ist es möglich, kleine unperformante MQTT-Geräte mit einem MQTT-Broker zu verbinden, wobei die Geräte Daten einsammeln und/oder Befehle entgegennehmen, während ein komplexes Lagebild nur auf dem MQTT-Broker entsteht und hier oder durch einen leistungsfähigen Kommunikationspartner ausgewertet werden kann.
Stelleingriffe können so von einer oder mehreren leistungsfähigen Instanzen an den MQTT-Broker übermittelt und auf die einzelnen Geräte verbreitet werden.
Dadurch eignet sich MQTT sehr gut für Automatisierungslösungen und findet im Bereich IoT durch die einfache Verwendung große Verbreitung.

MQTT Mosquitto Broker Funktion



MQTT Beispiel Programm

s2.eu.hivemq.cloud

Arduino Configuration

- Download and install CH340G USB driver
- Install ESP8266 module
- Install PubSubClient library (by Nick O'Leary)
- Sketch -> Include Library -> Manage Libraries… -> Type PubSub in Search field -> Install

MQTT Beispiel Programm für Poti Pos senden

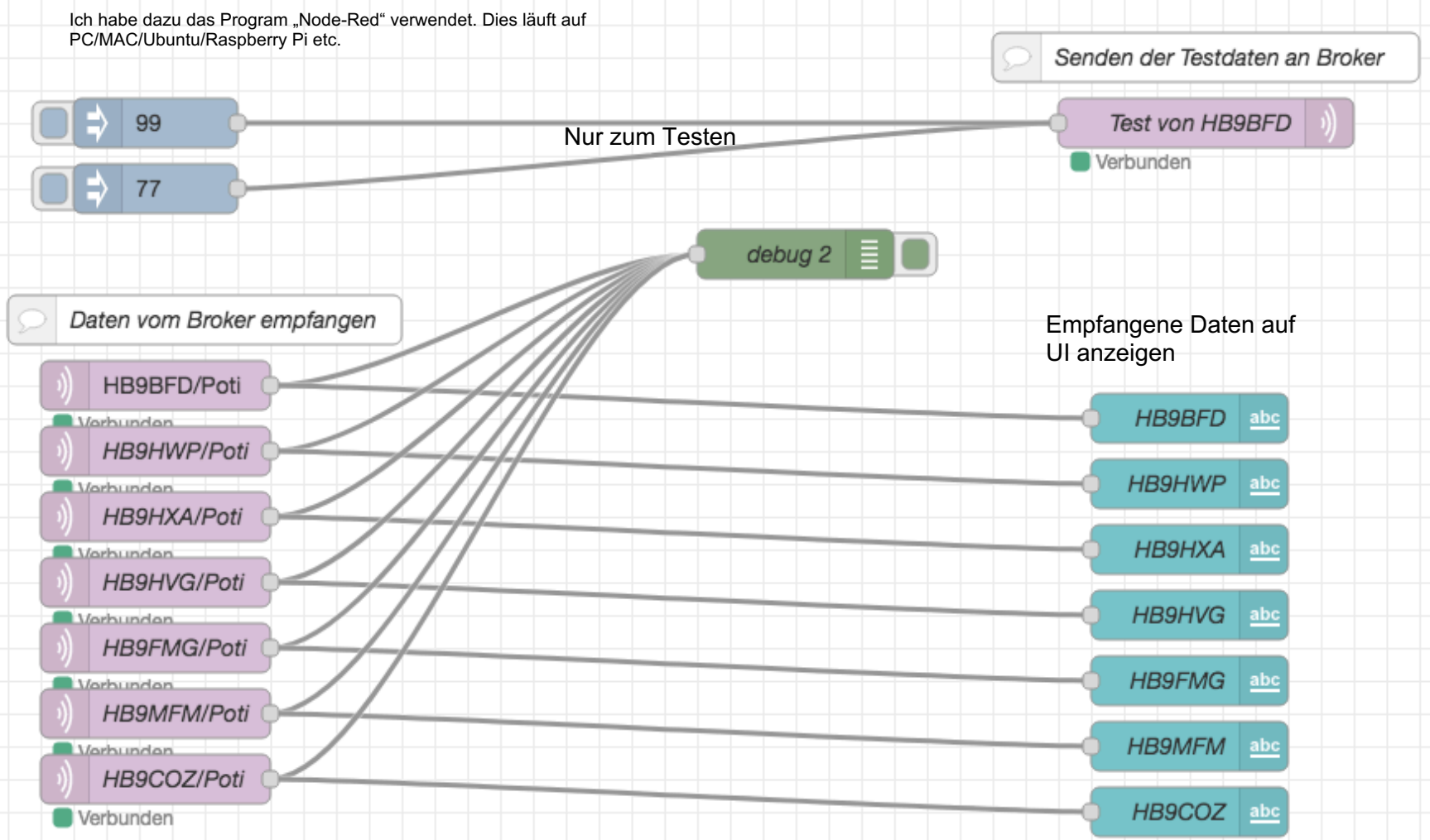
```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
const int analogInPin = A0; // ESP8266 Analog Pin ADC0 = A0

#define wifi_ssid " HB9BA "
#define wifi_password "xxxxx"

#define mqtt_server "b48168793db644848e07472b4fb7ea01.s2.eu.hivemq.cloud"

// MQTT Cloud address
#define Topic "HB9BFD/Poti"

WiFiClient espClient;
PubSubClient client(espClient);

void setup() {
Serial.begin(115200);
setup_wifi();
client.setServer(mqtt_server, 1883);
}

void setup_wifi() {
delay(10);
WiFi.begin(wifi_ssid, wifi_password);
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
}

void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
if (client.connect("nodeMcuHB9BFD")) {
Serial.println("connected");
} else {
Serial.print("failed, rc=");
Serial.print(client.state());
Serial.println(" try again in 5 seconds");
delay(5000);
}
}
}
```

Nur Grundgerüst,
gültiges Programm
siehe Beilage

```
bool checkBound(float newValue, float prevValue, float maxDiff) {
return newValue < prevValue - maxDiff || newValue > prevValue + maxDiff;
}

long lastMsg = 0;
float potiPos = 0.0; // value read from the poti
float diff = 10.0;

void loop() {
if (!client.connected()) {
reconnect();
}
client.loop();

long now = millis();
if (now - lastMsg > 30000) {
// Wait a few seconds between measurements
lastMsg = now;
potiPos = analogRead(analogInPin);

if (checkBound(potiPos, prevPos, diff)) {
prevPos = potiPos;
Serial.print("Neue Position:");
Serial.println(String(potPos).c_str());
client.publish(Topic, String(potiPos).c_str(), true);
}
}
}
```

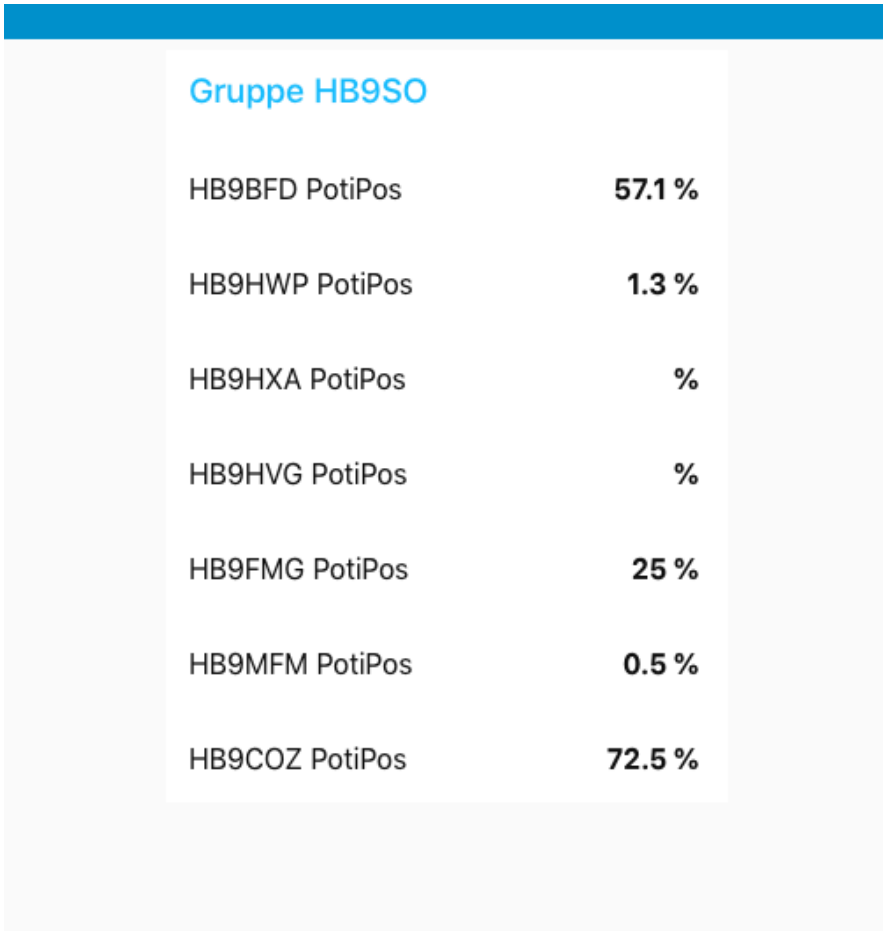
Resultat auf Node-Red sichtbar machen

Ich habe dazu das Program „Node-Red“ verwendet. Dies läuft auf PC/MAC/Ubuntu/Raspberry Pi etc.



Unter…….“IP-Adresse“:1880

Anzeige auf Mobil Telefon



Unter…….“IP-Adresse“:1880/ui

Anwendungen im Amateurfunk

- Für was ist der Arduino weniger geeignet?
  - Wenn grosse Datenmengen verarbeitet werden müssen.
  - Ansteuern von Graphikdisplays
  - FT8-Signale dekodieren

Welche Anwendungen kennen wir?

- AD9850 ansteuern (VFO etc.)
- Programmierbarer Oszi Si570 0.5….160Mhz von Axe Schultze, DK4AQ
- ARDUINO©- Mikrocontroller im Amateurfunk von Mathias Dahlke, DJ9MD
- ADF4351-Baustein von SV1AFN http://www.kh-gps.de/adf4351.htm
- Stationsuhr von Ingo, DL6IS https://p34.meindarc.de/?p=1182
- Morsekeyer von Ingo, DL6IS https://p34.meindarc.de/?p=1198
- SWR Analyser von Daniel, DL2AB https://wiki.funkfreun.de/projekte/swr-analyser
- CAT- Interface http://www.kh-gps.de/Arduino_8f.pd
- Rotor- Interface, K3NG https://blog.radioartisan.com/yaesu-rotator-computer-serial-interface
- µBitX Transceiver mit Nextion©
- APRS-Terminal von Jürgen, DL8MA http://www.dl8ma.de/Arduino/APRS-Terminal/
- Remote Power/SWR Meter von Michael Clemens, DK1MI.radio → HB9MFM
- …..zig viele Beispiele im Internet

Quellen

Folie	Quelle
1,27,28,38,39, 53	Aliexpress China diverse Websites
29	RN-wissen.de, Earthbondhon.com
25,26,31,32,36 ,42,44,45,46	Printscreens von Arduino IDE
58	Gelfiebermücke, Wikipedia.org
59	HiveMQ
41	Bastelgarage.ch, PIN Definition
64	IoT Tutorial #18 Raspberry Pi by Akshay Daga Feb 06,2018