# Summerschool Vortreffen

#3 Vernetzte Dinge

Johannes Deger
*kiz Universität Ulm*

Simon Lüke
*Studierendenwerk Ulm*

# Hardware kommuniziert - BUS

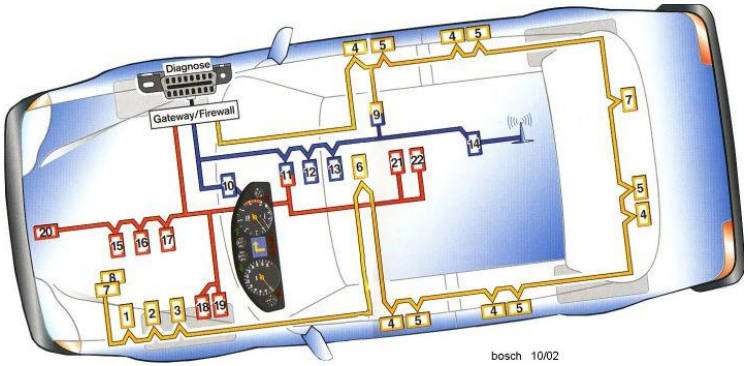
*Ein Bus ist ein System zur **Datenübertragung** zwischen mehreren Teilnehmern über einen **gemeinsamen Übertragungsweg**.*

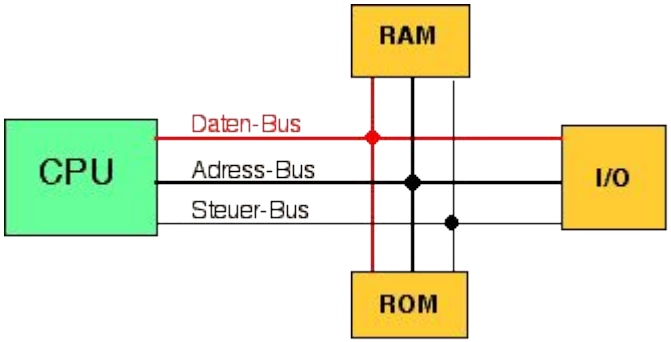
**Verwendung:**

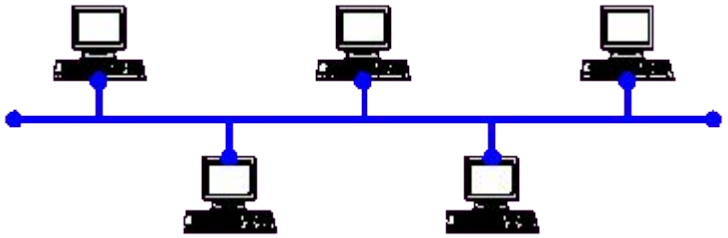
Vernetzung von Geräten!

**Nutzen:**

Erweiterung von Grundfunktionalitäten



# Ein BUS für alle? - Vielleicht zu voll

Alle Teilnehmer teilen sich einen BUS...

... alle Teilnehmer können sich gegenseitig stören.

**Problem:**

Es darf immer nur einer sprechen. Wie teile ich diesem mit, wann er das darf?

**Lösung:**

*Entweder:* **Scheduling** - jeder darf zu einem bestimmten Zeitpunkt.

Beispiel: Token-Ring (gaaanz alt).

*Oder:* **Busmastering** - der Master sagt, wer sprechen darf.

Beispiel: PCI (DMA), SPI, IDE....

*Oder*: **CSMA** - hören ob frei, dann losreden.

Beispiel: .... (ihr seid gefragt)

# Taktung eines Busses

**Problem:**

Wann kann ich auf die Leitung schauen?

**Lösung:**

Taktung des Busses.

**Zwei Arten:**

*synchron:*

Taktleitung an alle Teilnehmer (SPI)

*asynchron:*

Takt wird aus Signal gewonnen (RS232)

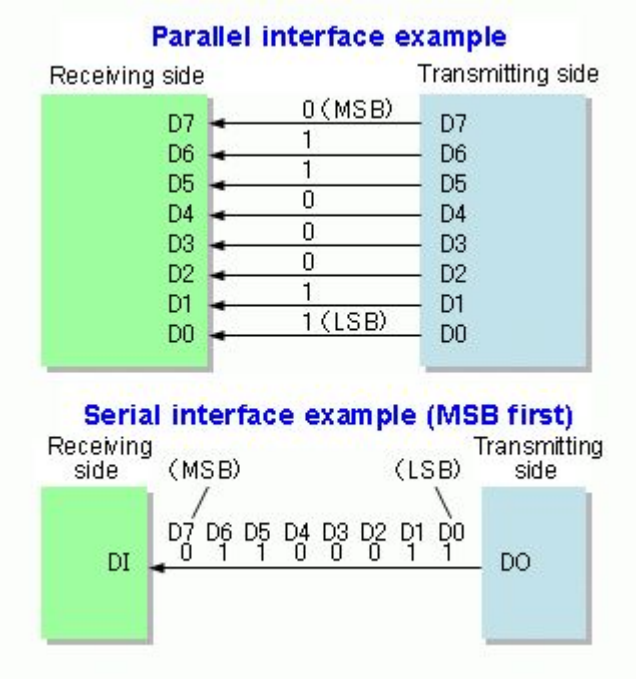
# Datenübertragung auf BUS

**Seriell:**

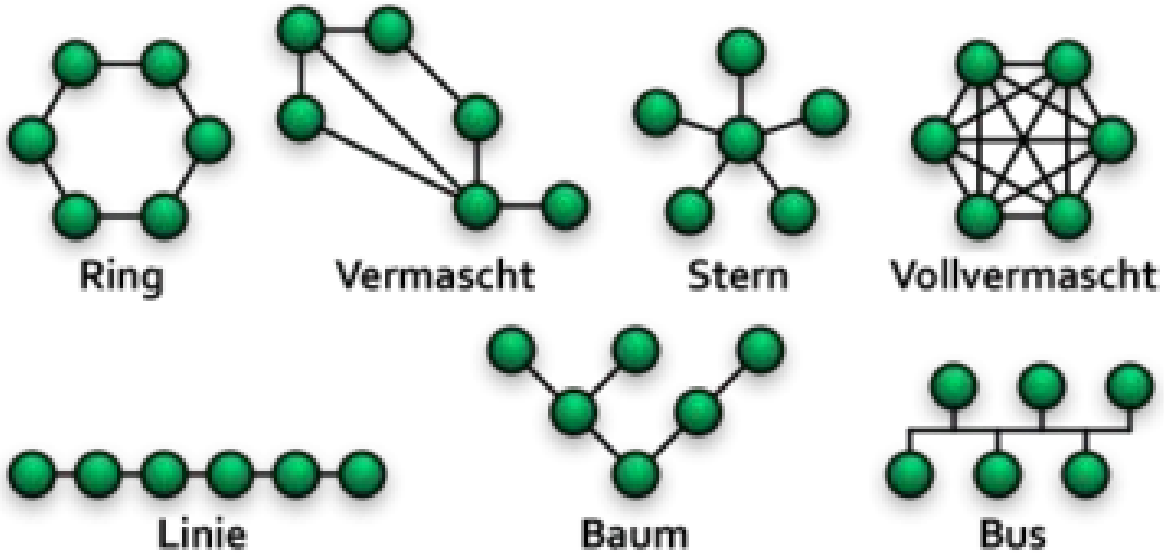
Bits werden nacheinander über eine Leitung geschickt.

**Parallel:**

Immer synchron! Bits werden über n-Leitungen zeitgleich zur Verfügung gestellt.



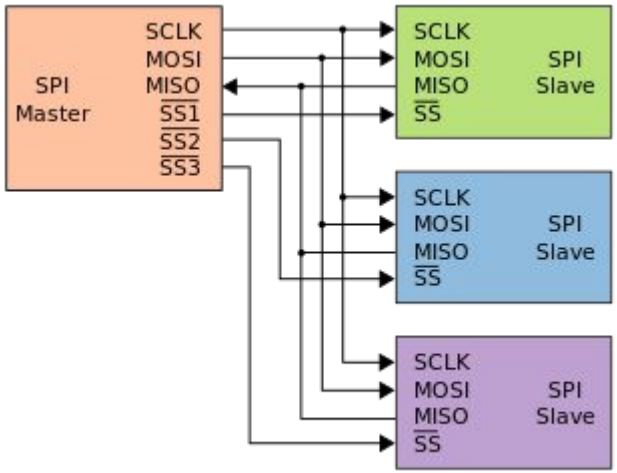
# Busverkabelung

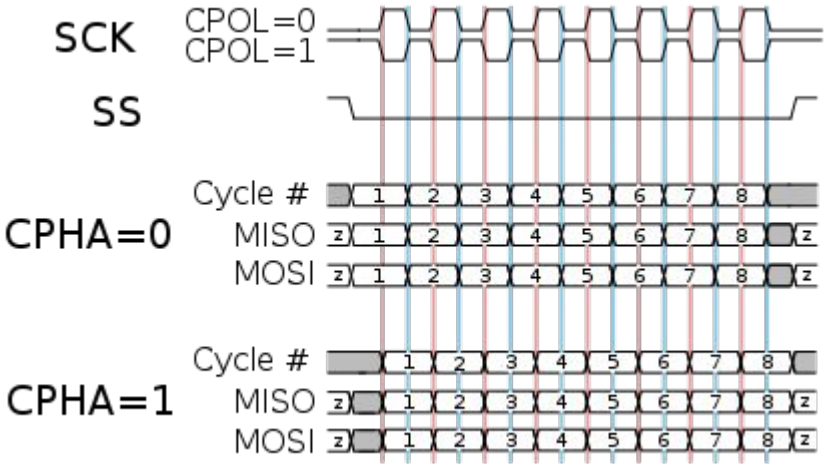


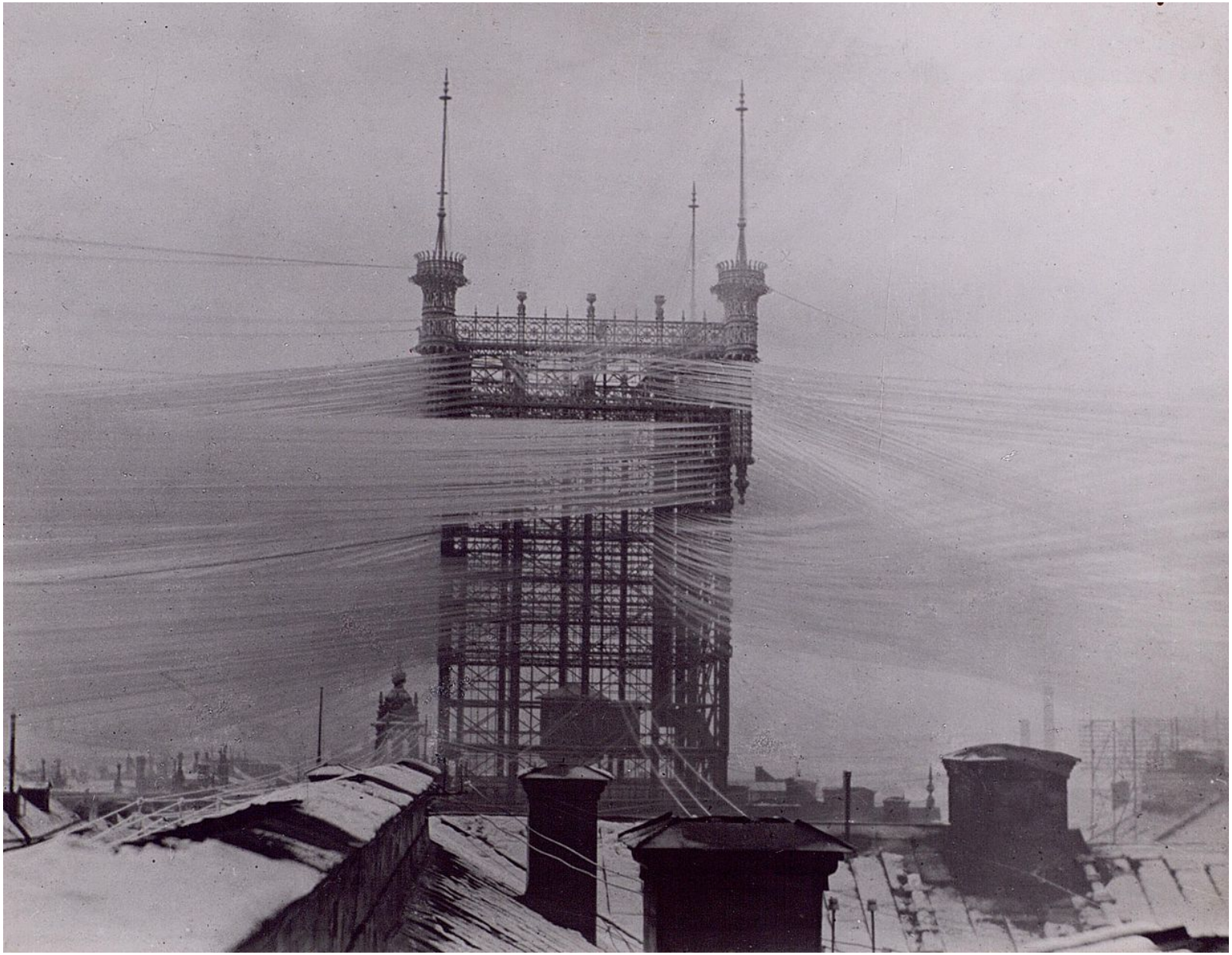
# Beispiel: SPI

SPI — *Serial Peripheral Interface:*

- serieller BUS
- synchroner BUS
- Master-/Slave-BUS
- Taktflankengesteuert





*Telefontornet, Stockholm,1887-1913, https://commons.wikimedia.org/wiki/File:Telefontornet6838150900.jpg*

# Zahlenrepräsentation

# Zahlendarstellung im Computer

Zahlen werden *binär* dargestellt...

... d.h zur Basis 2 mit den *Zeichen* 0 und 1

**Überlegung:**

**Darstellung Basis 10**: Zeichen 0...9 repräsentieren “Ziffern”.

Beispiel:

$$(20123)_{10} = 1 \cdot 10^4 + 0 \cdot 10^3 + 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0$$

**Darstellung in Basis 2:**

$$(1101)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^0 = 16 + 8 + 0 + 1 = (25)_{10}$$

# Weitere Zahlensysteme

**Hexadezimal - Basis 16:**

Die Zeichen 0...9 und A...F stellen die Ziffern dar.

Dabei: $(0)_{16} = (0)_{10}$

$(F)_{16} = (15)_{10}$

**Verwendung:** Gut zur Darstellung von Bytes (später mehr)

**Wichtig:** Oft auch als 0x(Zahl) gekennzeichnet.

**Oktal - Basis 8:**

Die Zeichen 0...7 stellen die Ziffern dar.

Dabei: Gleiche Wertigkeit wie im Dezimalen.

# Umrechnung

**Anything -> 10er-System**

Einfaches Aufaddieren der Wertigkeiten:

$$(1101)_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^0 = 16 + 8 + 0 + 1 = (25)_{10}$$

$$(BEEF)_{16} = B \cdot 16^3 + E \cdot 16^2 + E \cdot 16^1 + F \cdot 16^0$$
$$= 11 \cdot 4096 + 14 \cdot 256 + 14 \cdot 16 + 15 \cdot 16 = (48879)_{10}$$

**10er-System -> Binär**

Hornerschema, siehe Tafel.

# Hex-System

Besonders gut geeignet, um Bytes kurz darzustellen.
Beispiel:

$$(11110010)_2 = 0\text{x}F2$$

Umrechnung besonders einfach in wenigen Schritten.

1. Prüfen, ob Anzahl der Ziffern durch 4 teilbar
2. Wenn nein, ergänze genügend führende Nullen
3. Gruppiere Ziffern *von hinten an* zu 4er-Blöcken
4. Rechne 4er-Blöcke einzeln um.

**Beispiele:**

$$(1011101011101)_2 = (1 \quad 0111 \quad 0101 \quad 1101)_2 = (\underbrace{0001}_{(1)_{16}} \quad \underbrace{0111}_{(7)_{16}} \quad \underbrace{0101}_{(5)_{16}} \quad \underbrace{1101}_{(D)_{16}})_2 = 0\text{x}175\text{D}$$

$$(11110010)_2 = (\underbrace{1111}_{(F)_{16}} \quad \underbrace{0010}_{(2)_{16}})_2 = 0\text{xF2}$$

# Negative Zahlen (I)

Bisher: Nur *positive* Zahlen darstellbar.

Überlegung: Wie können *negative* Zahlen dargestellt werden?

**Ansatz 1:**

Vorzeichen speichern als Bit.

Beispiel: $(0111)_2 = 7$

$(1111)_2 = -7$

Nachteile: - Rechenwerk in der CPU wird sehr kompliziert.

- Doppelte Null: $(0000)_2 = +0; \quad (1000)_2 = -0$

**Ansatz 2:**

# Negative Zahlen (II)

**Bester Ansatz: Zweierkomplement**

*Vorüberlegung:*

Wir nutzen den Overflow aus!

Siehe Grafik.

*Vorteile:*

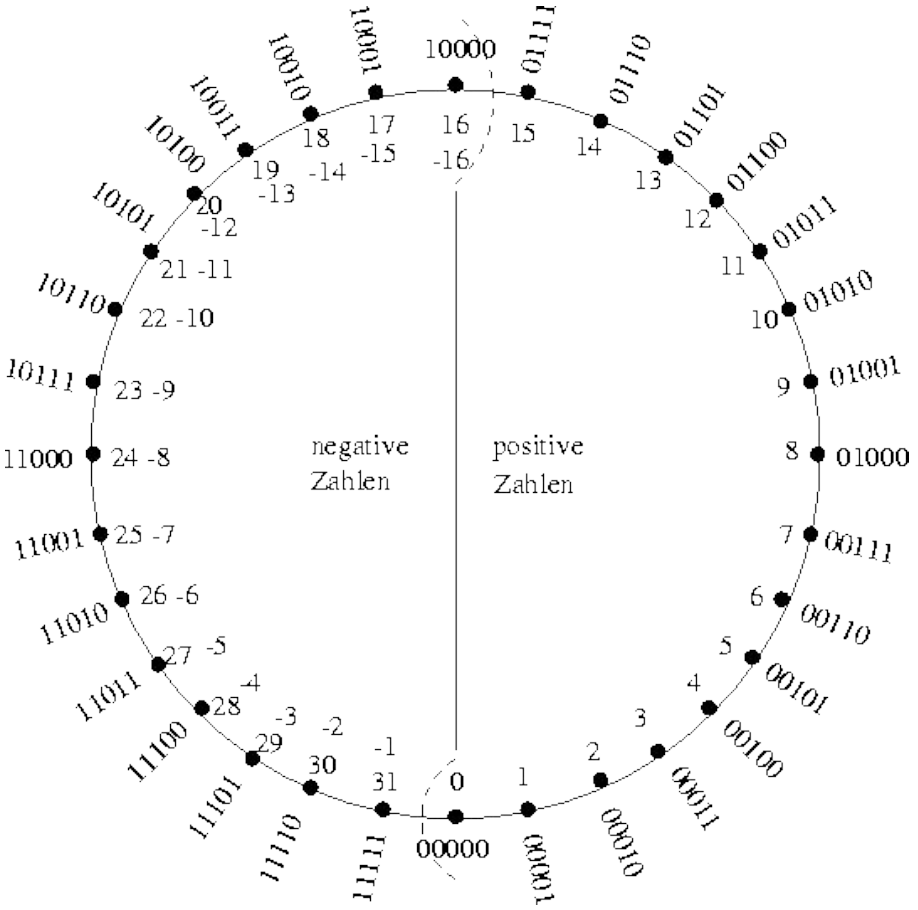
Keine doppelte Null

Benötigt NUR Addierwerk

**ACHTUNG:**

**Werterbereich wird eingeschränkt!**

**Bei 8 bit:** -128 - +127



# Einschub: Informationsdarstellung

*Problem:* Darstellung im Computer entspricht nicht mehr “eigentlicher” Bedeutung!

*Dazu:*

Bits und Bytes speichern Informationen immer als ‘1’ und ‘0’.

Interpretation der Daten hat nichts mit Darstellung zu tun!

Beispiel: 0x41

*Interpretation:*

Zahl: 65

Binär: $(0100 \quad 0010)_2$

ASCII: A

Interpretation ist das Entscheidende!

# Berechnung Zweierkomplement

**Berechnung erfolgt in Schritten anhand Beispiel $(-15)_{10}$ im 8-bit ZK**

**Schritt 1:**

Umrechnen der *positiven* Zahl ins Zweiersystem (Hornerschema): $(15)_{10} = (1111)_2$

**Schritt 2:**

Ergänzen der Zahl mit führenden Nullen (bis 8-bit): $(0000 \quad 1111)_2$

**Schritt 3:**

Invertieren der Zahl mit NOT: $(1111 \quad 0000)_2$

**Schritt 4:**

Eins dazuzählen: $(1111 \quad 0000)_2 + (1)_2 = (1111 \quad 0001)_2$

# Rechnen mit dem ZK

Beispiel: $(-15) + 7 = 8$

**Schritt 1:**

Beide Zahlen umrechnen.

$7 = (0000 \quad 0111)_2$ (kein ZK, da positiv)

$-15 = (1111 \quad 0001)_2$ (ZK, da negativ)

**Schritt 2:**

Schriftliche Addition: Siehe Tafel

Ergebnis: $(1111 \quad 1000)_2$

# Zurückrechnen

**Schritt 1:**

Checke höchstes Bit. Wenn 0: -> Zahl positiv, normal zurückrechnen

Wenn 1: -> Zahl negativ, Zweierkomplement zurückrechnen

**Schritt 2:**

Invertieren $\mathrm{NOT}(1111 \quad 1000)_2 = (0000 \quad 0111)_2$

**Schritt 3:**

1 addieren: $(0000 \quad 0111)_2 + (0000 \quad 0001)_2 = (0000 \quad 1000)_2$

**Schritt 4:**

Normal Umrechnen: $(0000 \quad 1000)_2 = (8)_{10}$

**Schritt 5:**

Da vorher 1 am Anfang, Minuszeichen beachten! $(1111 \quad 1000)_2 = (-8)_{10}$

# Informationskodierung: ASCII

Darstellung von Zeichen möglich?

JA - ASCII-Code

# Große Frage: Warum?!

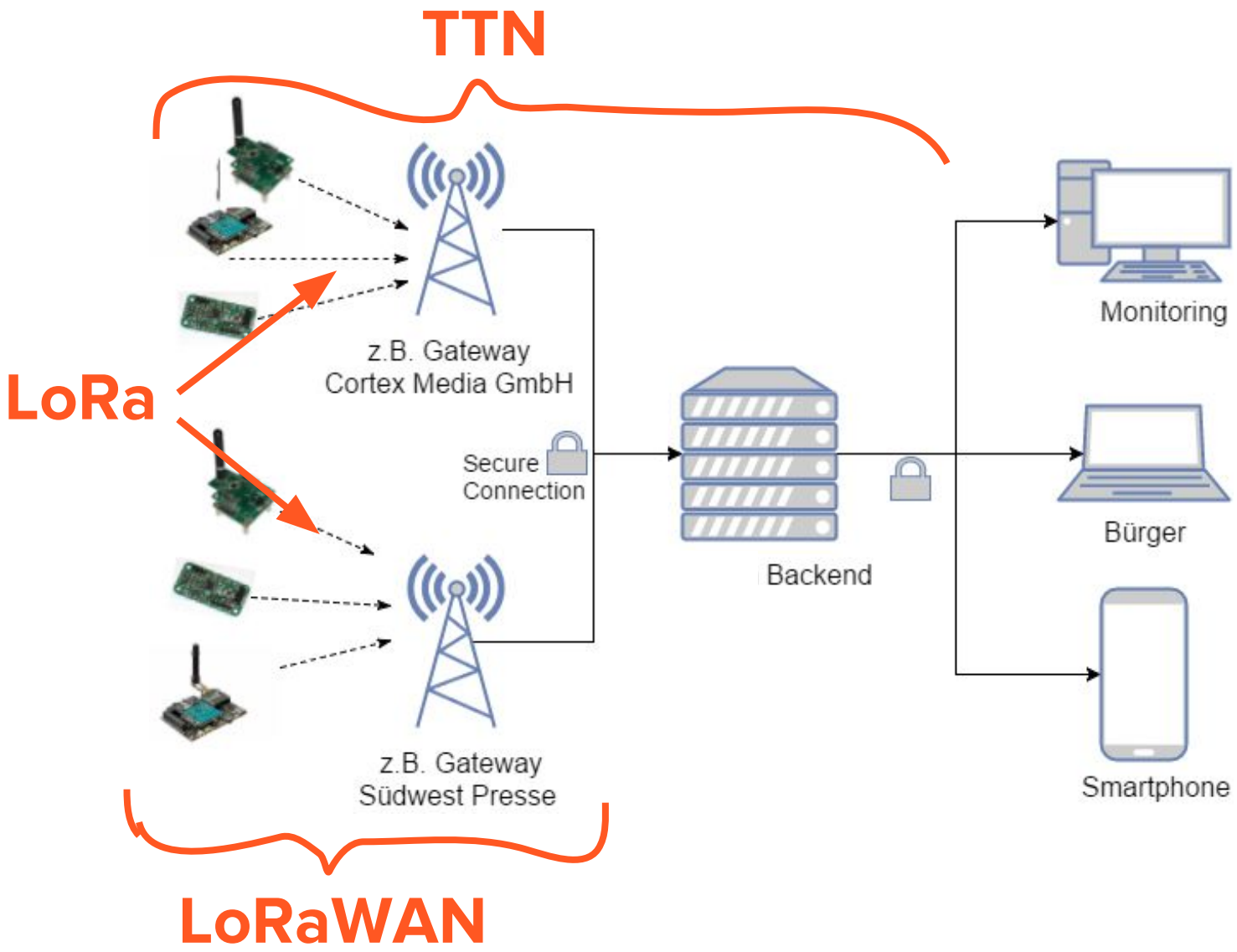
# LoRaWAN
# = Low Power Wide Area Network

# ... dafür geringe Datenrate

# The Things Network (TTN)

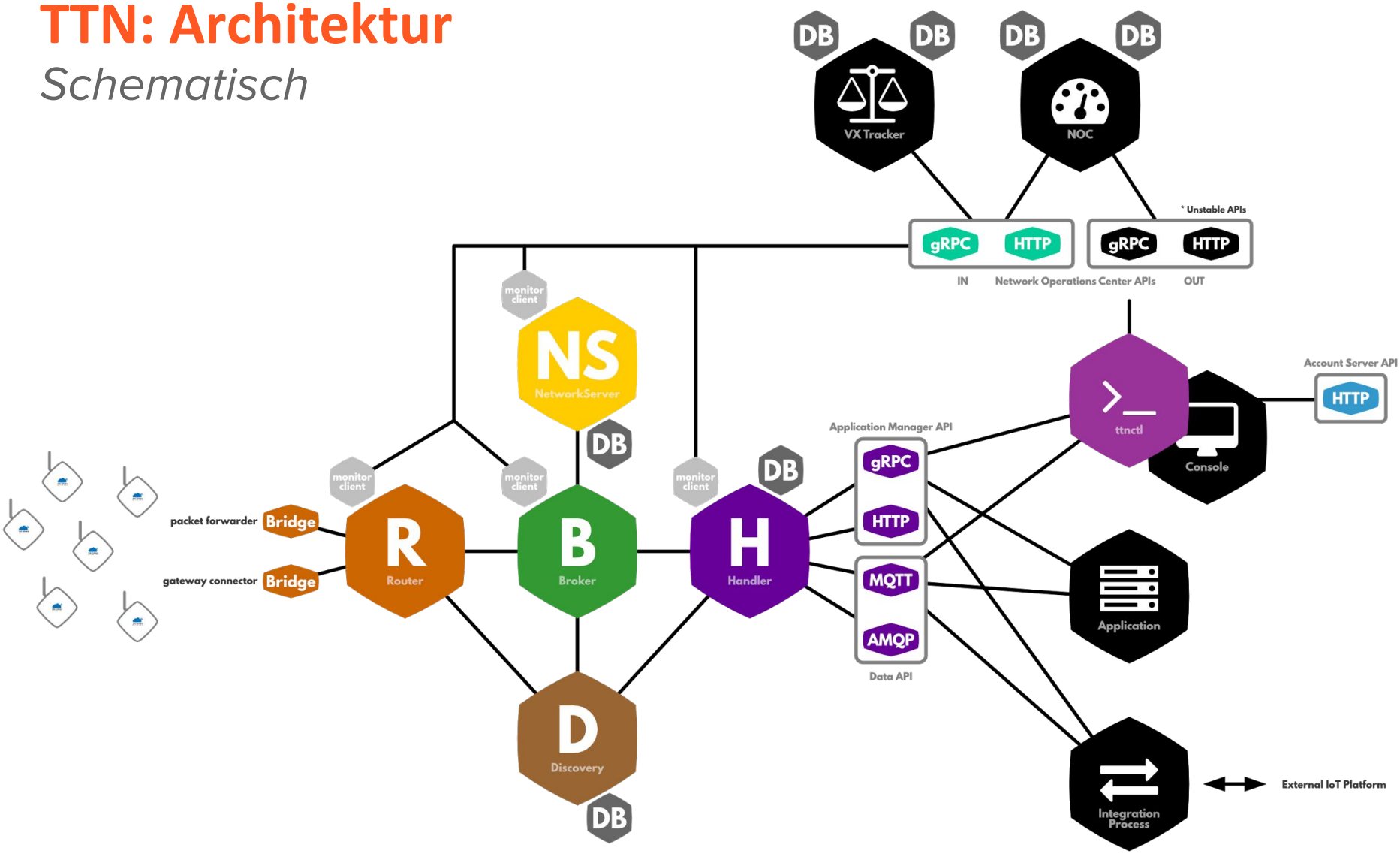
Building a Global Internet of Things Network Together

# TTN: Überblick und Begriffe



# TTN: Architektur

*Schematisch*

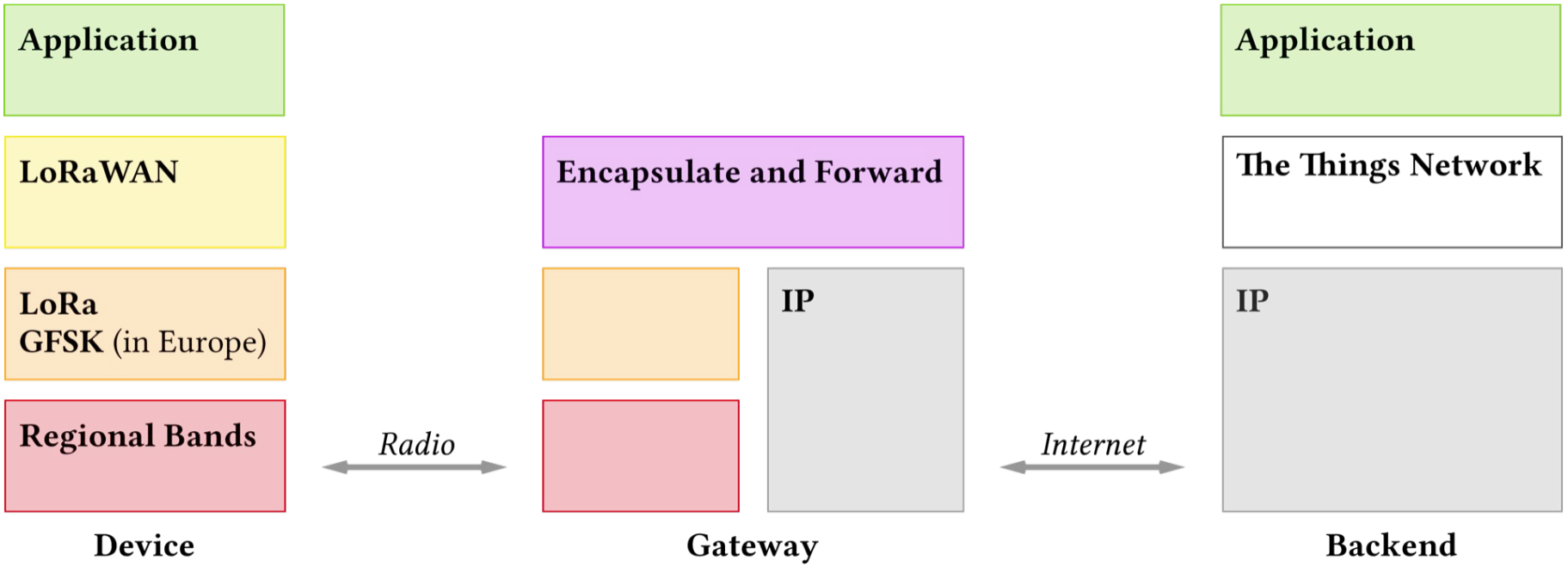



https://www.thethingsnetwork.org/wiki/Backend/Home

Auch zum Nachlesen.

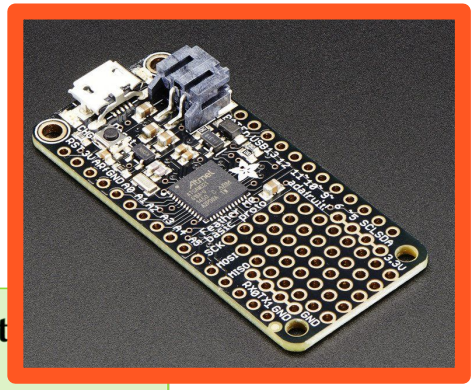
# TTN/LoRa(WAN)-Stack

*Einordnung ins ISO OSI-Modell*




# Was wollen wir bauen?

# Node



# Anwendung


Applicat

LoRaWAN

LoRa
GFSK (in Europe)

Regional Bands

Device

*Radio*

Encapsulate and Forward

IP

Gateway

*Internet*

on

The Things Network

IP

Backend

# Anwendung

**MQTT**

- Libraries für viele Sprachen
- https://mosquitto.org: Kommandozeile, Python

**Grafana**

- eine Möglichkeit für Visualisierungen

**Node-RED**

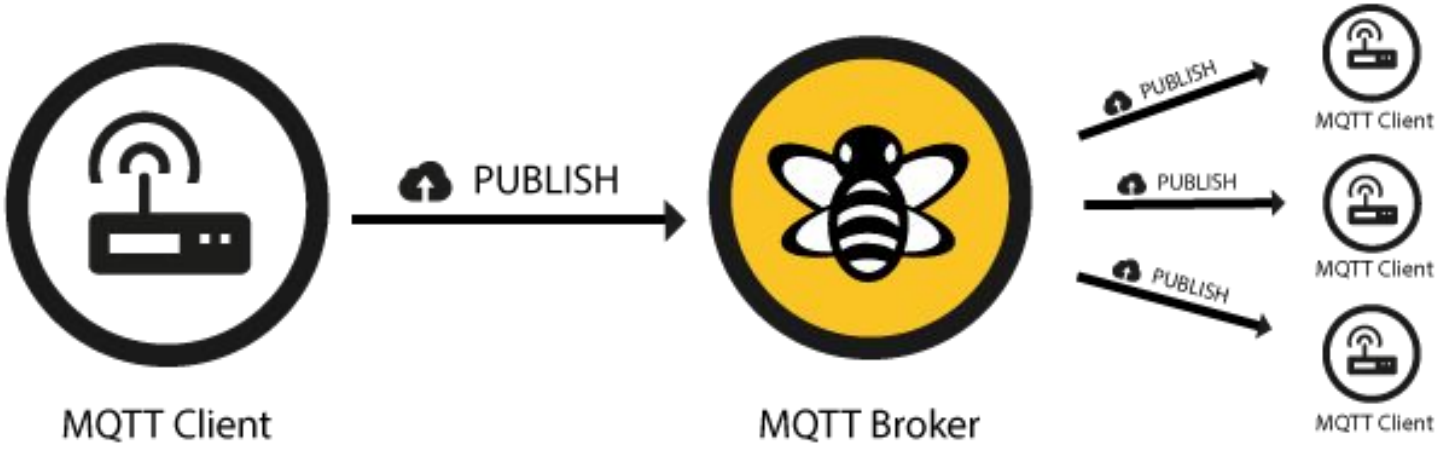
- Flow-based programming for the Internet of Things

**IFTTT**

- Kann auch an Node-RED angedockt werden.

**…?**

# Datenabruf: MQTT

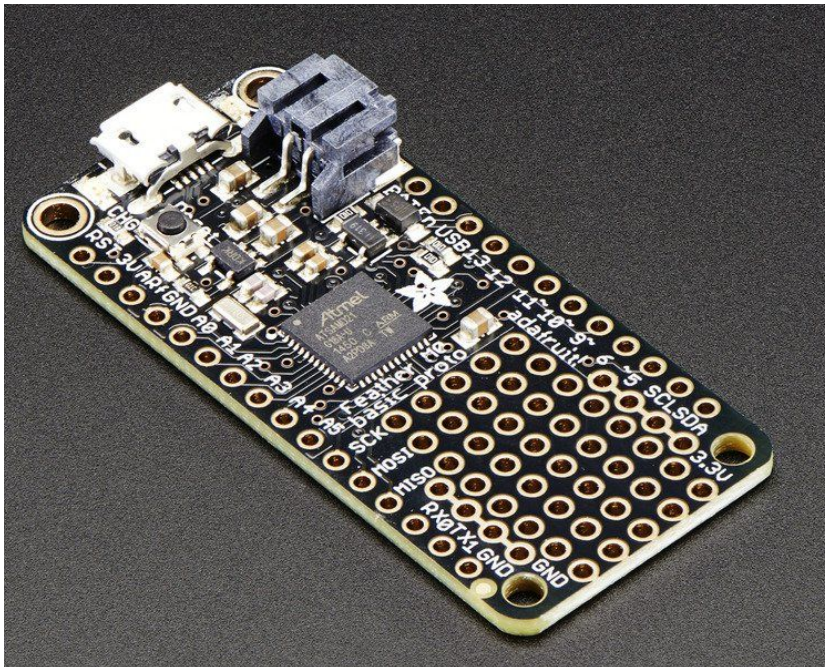



*Publish/Subscribe*

```
mosquitto_sub
    -h eu.thethings.network
    -t '+/devices/+/up'
    -u 'dhbw-ubiquitious_computing'
    -P ttn-account-v2.VauGzDcHcXqa5g1V[…]'
```

*https://www.thethingsnetwork.org/docs/applications/mqtt/*

# Node




Hardware

+

Software

# TTN: Loslegen

- Account → anlegen: https://account.thethingsnetwork.org/register
- Application anlegen.
- Devices/Nodes registrieren.
  - *Device: Settings*
    *> ABP auswählen*
    *> Frame Counter Check deaktivieren!*
- Daten empfangen :-) (Web GUI)

**zur Info:**

- Command line zur Bedienung (Device Status, Registrierung o.ä.)
- Text zur Architektur (Englisch)

*https://console.thethingsnetwork.org/*

# LMiC

*LoRaWAN MAC in C (Medium Access Control, Layer 2)*

## 1) Authentifikation

```
NWKSKEY, APPSKEY, DEVADDR
```

## 2) Pin Mapping: Steuerung/Kontrolle

```
const lmic_pinmap lmic_pins = { … };
```

## 3) Ereignsibasiert

```
void onEvent (ev_t ev) { … }
```

## 4) Senden

```
void do_send(osjob_t* j) { … }
```

# LMiC: Authentifizierung

*ABP, nicht OTAA (siehe weiter unten)*

```
static const PROGMEM u1_t NWKSKEY[16]
      = { 0xDB, …, 0x03 }; // MSB

static const u1_t PROGMEM APPSKEY[16]
      = { 0x41, …, 0xE0 }; // MSB

static const u4_t DEVADDR
      = 0x26....3E;
```

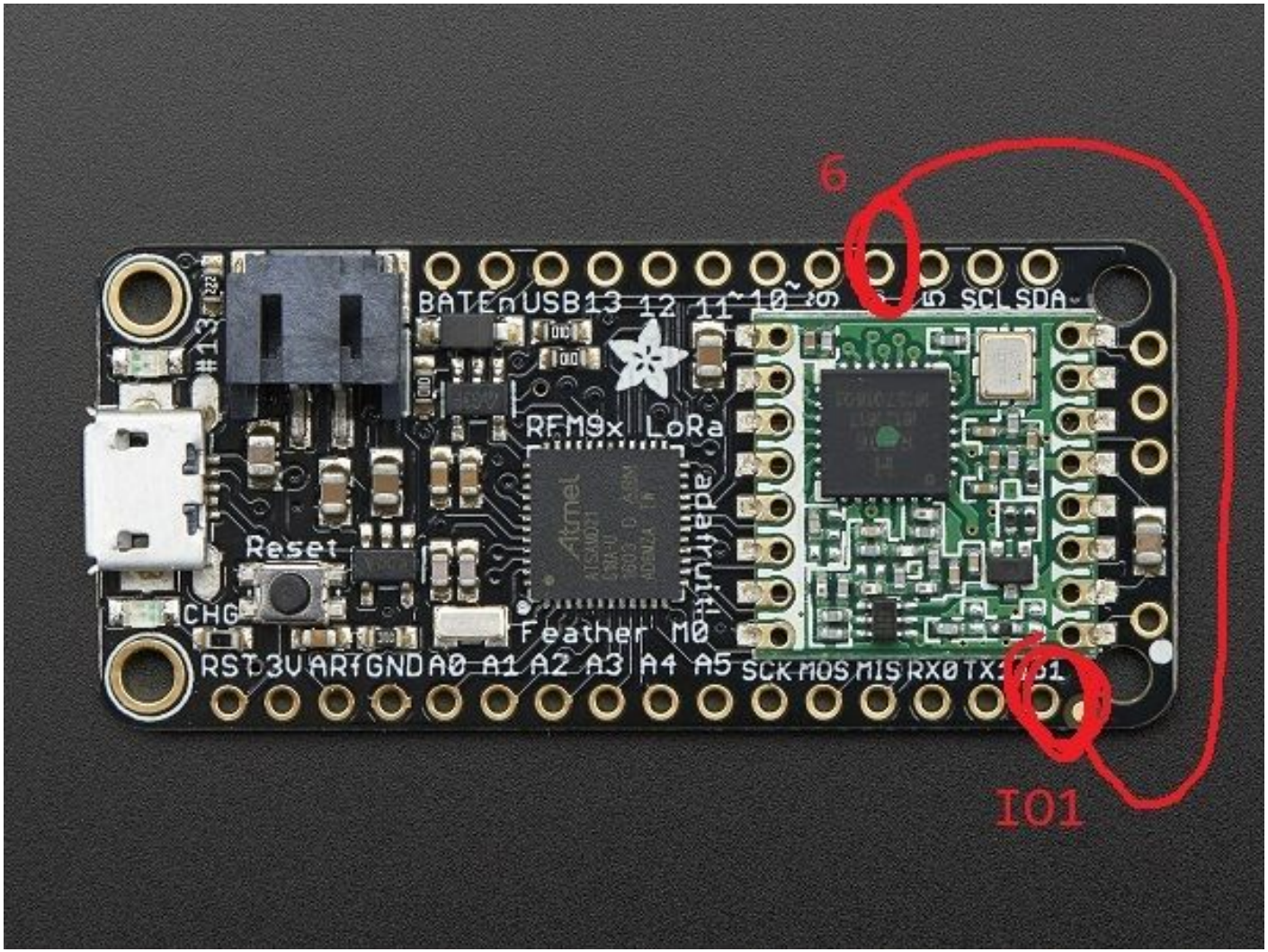
# LMiC: Pin mapping (Software)

```
const lmic_pinmap lmic_pins = {
    .nss = 8,
        // SPI chip select
    .rxtx = LMIC_UNUSED_PIN,
        // Antenna switch
    .rst = 4,
        // Reset, LMIC_UNUSED_PIN
    .dio = {3, 6, LMIC_UNUSED_PIN},
        //DIO0, DIO1, DIO2
};
```

http://www.cplusplus.com/doc/tutorial/structures/

Siehe LMiC-Doku (3.1 HAL Interface).

# LMiC: Pin mapping (Hardware)



# LMiC: Event based

*Kein loop()*

Rechts zwei Beispiele.

Es gibt viel mehr Ereignisse!

*Programmplanung:*

Was soll welche Aktivität auslösen?

```
void onEvent (ev_t ev) {
    switch(ev) {
        …
        case EV_BEACON_FOUND:
            …
            break;
        case EV_TXCOMPLETE:
            …
            break;
        default:
            …
            break;
    }
}
```

# LMiC: Event based

*timed callback*

```
case EV_TXCOMPLETE:

   …

   os_setTimedCallback(

      &sendjob,

      os_getTime()+sec2osticks(TX_INTERVAL),

      do_send

   );
```

# LMiC: LMIC_setTxData2

```
LMIC_setTxData2(
      1,
      mydata,
      sizeof(mydata)-1,
      0
   );
```

mydata

LoRaWAN kennt nur Bytes!

→ Enkodieren zum Senden + Dekodieren!

Also:

- Bitshift
- Bitmask
- Repräsentation von Zahlen (s.o,)
- Grundlagen bei TTN

# Hands On

# **Tx Hello World**

# ABP oder OTAA

*activation by personalization/over-the-air activation*

## ABP

- DevAddr, NwkSKey, und AppSKey
- *Geheimnis* in der Programmierung auf dem Gerät, dafür kein erfolgreicher Handshake (up+down) nötig

## OTAA

- join request + join accept
  - join request: DevEUI, AppEUI und DevNonce
  - join accept: AppNonce, NetID, DevAddr, DLSettings, RxDelay und CFList
  - NwkSKey und AppSKey mit aes128_encrypt() abgeleitet von AppNonce

*LoRaWAN Specification, v1.0.2, 6 End-Device Activation, S. 32ff*

# Nachweise/Quelle

- https://de.wikipedia.org/wiki/Parallele_Daten%C3%BCbertragung#/media/File:Parallel_and_Serial_Transmission.gif
- https://en.wikipedia.org/wiki/Digital_timing_diagram
- The Things Network
- Initiative Ulm Digital
- The Things Network Wiki
- Apps und Männchen: CC0
- https://www.heise.de/developer/artikel/Kommunikation-ueber-MQTT-3238975.html
- https://www.adafruit.com/product/2772
- *https://www.thethingsnetwork.org/forum/t/adafruit-lora-feather-gateway/2440/48*