IoT Internet of Things

Hartmut Seitter

Agenda – Topics which should be covered in this course

- Introduction
  - IoT Definition
  - What are the disciplines of IoT
  - A brief history about IoT
- Application scenarios for IoT
  - Cool applications and a lot of opportunities
- The IoT Value Stack a real world example
- Sensors
  - Wireless, sensor, networks and IoT
  - From Sensors and Actuators to ….
  - … embedded systems and computation
- **Circuits**
  - **Energy and Wireless**
  - **Digital & Analog**
- **Embedded Systems**
  - **Technology Drivers,**
  - **Energy,**
  - **Microcontroller,**
  - **Software**

Agenda – Topics which should be covered in this course

- Start with IoT Lab
- Connectivity and Networking
- IoT Networks and Communication frameworks
  - MQTT
  - LoRaWAN
  - CoAP
  - Zigbee
  - Thread / Matter
- Embedded power supplies, energy harvesting and constrains
- Architecture of distributed systems
- Internet principals, routing for IoT
- Information security and privacy concepts
- Big Data and Machine Learning

How do we connect our devices

- **we have end-nodes (sensor with microcontroller and connectivity)**

What are typical communications protocols used?

- **WiFi WLAN, Bluetooth BLE and in addition on the development boards**
- **e.g. Zigbee, LoRaWAN, LTE, Thread, HTTP, MQTT, Matter**

- **we have gateways (e.g. smartphones, hubs, etc.)**

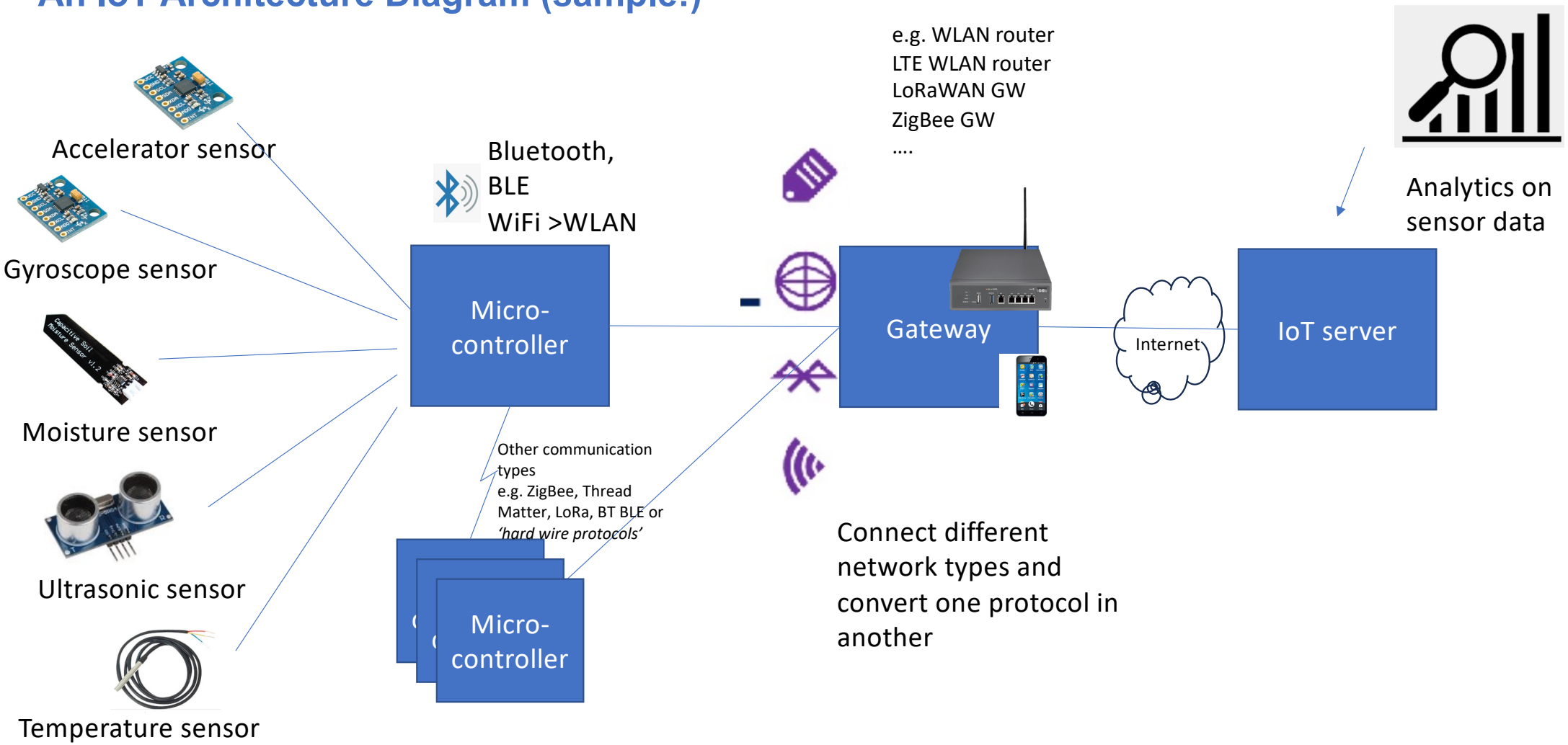
What are typical communications protocols used?

- **same as above**
- **LTE 4G,5G**
- **WiFi**
- **TCP/IP**

- **MQTT, HTTP**

- **and Server**
  - **TCP/IP, HTTP, MQTT, …….**

An IoT Architecture Diagram (sample!)



An IoT Architecture Diagram (sample!) MQTT subject

MQTT is a standard protocol in IoT solutions an it can be used on different levels

Accelerator sensor

Gyroscope sensor

Moisture sensor

Ultrasonic sensor

Temperature sensor

Micro-controller

Internet

IoT server

Analytics on sensor data

MQTT

MQTT

other comm. Types and protocols

MQTT requires a TCP/IP connection

MQTT

See https://en.wikipedia.org/wiki/MQTT

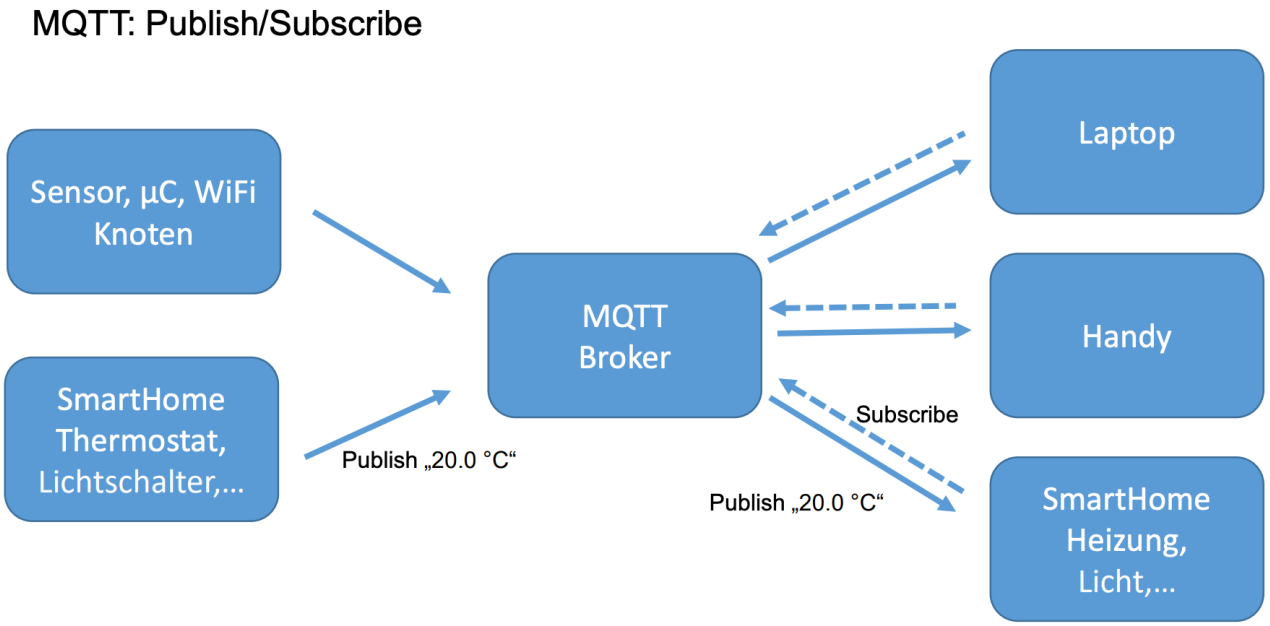
Andy Stanford-Clark (IBM) and Arlen Nipper (Cirrus Link, then Eurotech) authored the first version of the protocol in 1999.[6] It was used to monitor an oil pipeline through the desert. The goal was to have a protocol that is bandwidth-efficient, lightweight and uses little battery power, because the devices were connected via satellite link which, at that time, was extremely expensive.[7]

The MQTT protocol defines two types of network entities: a message broker and a number of clients. An MQTT broker is a server that receives all messages from the clients and then routes the messages to the appropriate destination clients.[13] An MQTT client is any device (from a micro controller up to a full-fledged server) that runs an MQTT library and connects to an MQTT broker over a network.[14]

See also
https://mosquitto.org/documentation

MQTT Pub/Sub

MQTT – Message Queue Telemetry Transport



Im Gegensatz dazu HTTP: Request/Response

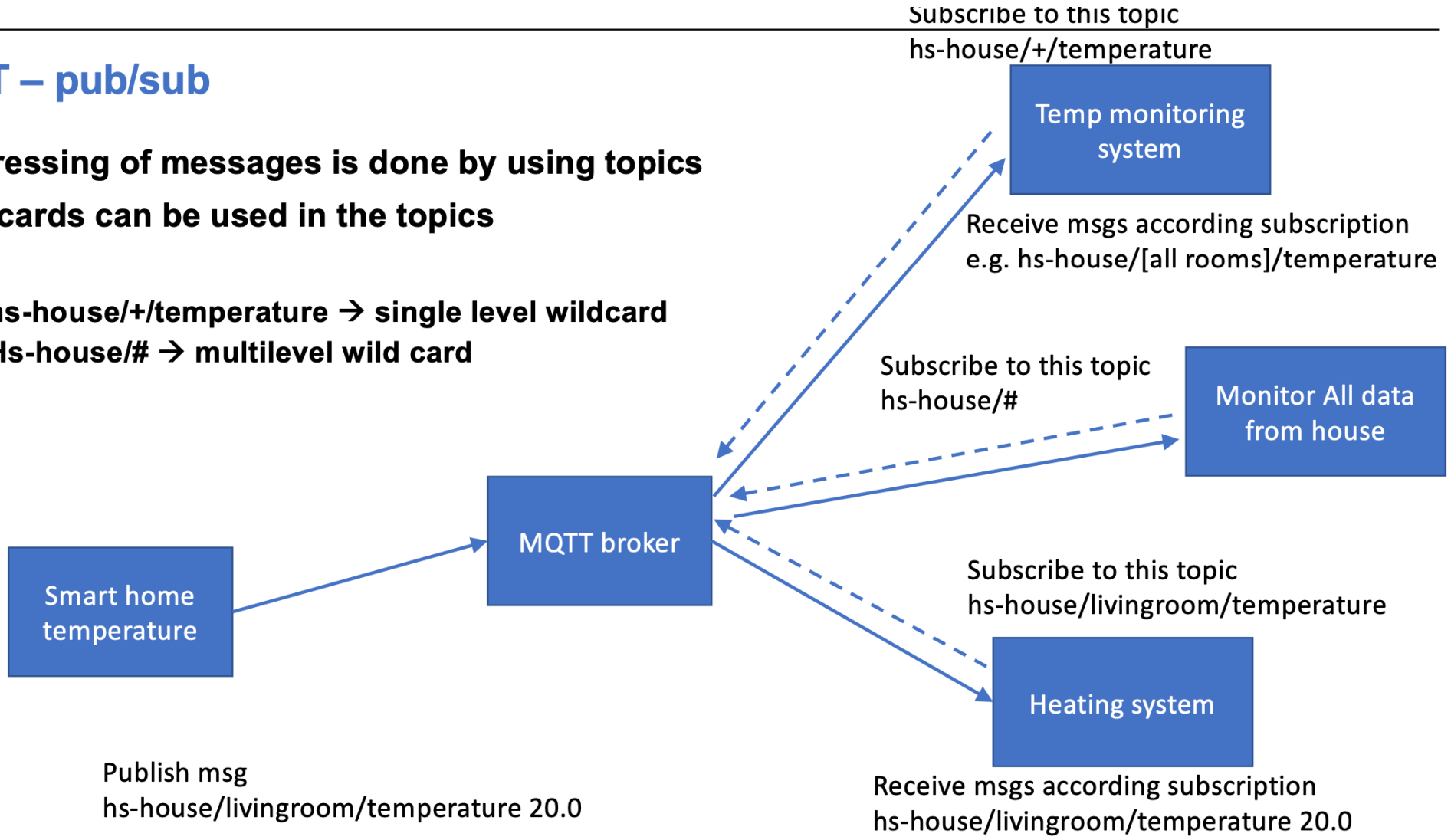
Bild nach: Walter Trokan, Das MQTT-Praxisbuch

MQTT basics

- **MQTT is the standard protocol for messaging and data exchange for the Internet of Things**
- **MQTT requires a TCP/IP connection**
- **MQTT is a data-centric, extremely lightweight, binary protocol.**
  - **HTTP is based on a request/response pattern**
  - **MQTT is based on pub/sub pattern**
- **MQTT is designed to allow very secure communication. As an application layer protocol, it introduces extensive device authentication and authorization possibilities. The underlying TCP/IP transport protocol can add additional security via TLS encryption.**

MQTT – pub/sub

- **Addressing of messages is done by using topics**
- **Wildcards can be used in the topics**
  - **hs-house/+/temperature → single level wildcard**
  - **Hs-house/# → multilevel wild card**



MQTT – Quality of Service

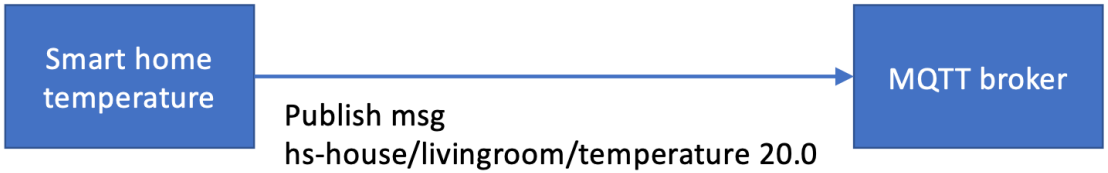
• The **Quality of Service** (QoS) level is an agreement between the sender of a message and the receiver of a message that defines the guarantee of delivery for a specific message.

There are 3 QoS levels in MQTT:
• *At most once*(0)
• *At least once*(1)
• *Exactly once*(2).

MQTT Quality of Serivce

- **QoS 0 - at most once**

  The minimal QoS level is zero. This service level guarantees a best-effort delivery. There is no guarantee of delivery. The recipient does not acknowledge receipt of the message and the message is not stored and re-transmitted by the sender. QoS level 0 is often called “fire and forget” and provides the same guarantee as the underlying TCP protocol.



Minimum network load

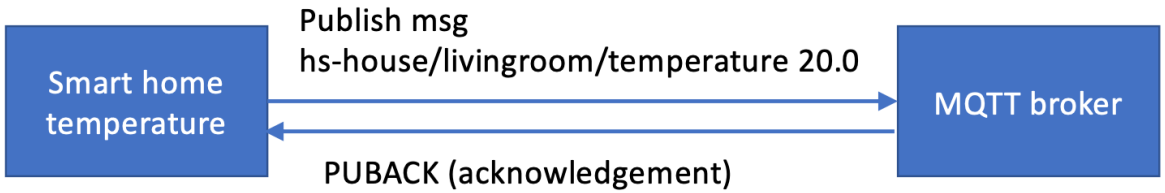
No acknowledgement

No buffering

MQTT Quality of Service

- **QoS 1 - at least once**

  QoS level 1 guarantees that a message is delivered at least one time to the receiver. The sender stores the message until it gets a **PUBACK** packet from the receiver that acknowledges receipt of the message. It is possible for a message to be sent or delivered multiple times.



moderate network load

acknowledgement

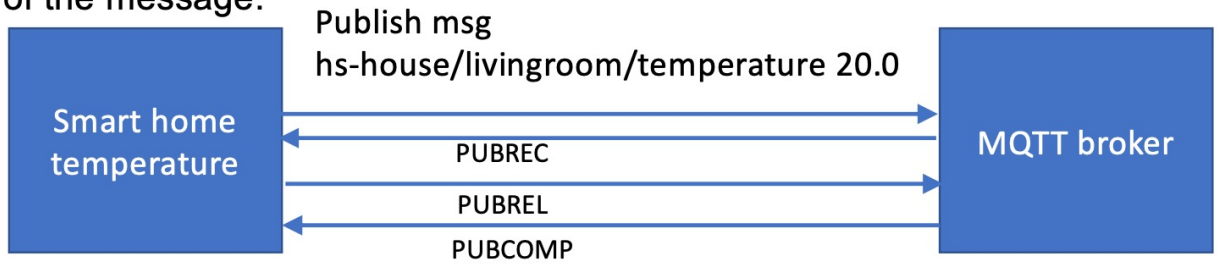
buffering till acknowledgement

Msg will be send several time if no acknowledgement is received

MQTT Quality of Service

- **QoS 2 - exactly once**

  QoS 2 is the highest level of service in MQTT. This level guarantees that each message is received only once by the intended recipients. QoS 2 is the safest and slowest quality of service level. The guarantee is provided by at least two request/response flows (a four-part handshake) between the sender and the receiver. The sender and receiver use the packet identifier of the original PUBLISH message to coordinate delivery of the message.



moderate network load

acknowledgement

buffering till acknowledgement

Msg will be send several time if no acknowledgement is received

When a receiver gets a QoS 2 PUBLISH packet from a sender, it processes the publish message accordingly and replies to the sender with a **PUBREC** packet that acknowledges the PUBLISH packet. If the sender does not get a PUBREC packet from the receiver, it sends the PUBLISH packet again with a duplicate (DUP) flag until it receives an acknowledgement.

LoRa and LoRaWAN – short introduction – we will cover LoRaWAN in more detail in one of the following lecture

Today I would like to give you a short overview about LoRaWAN only

- **Reason–we will step into the IoTLab today as well**

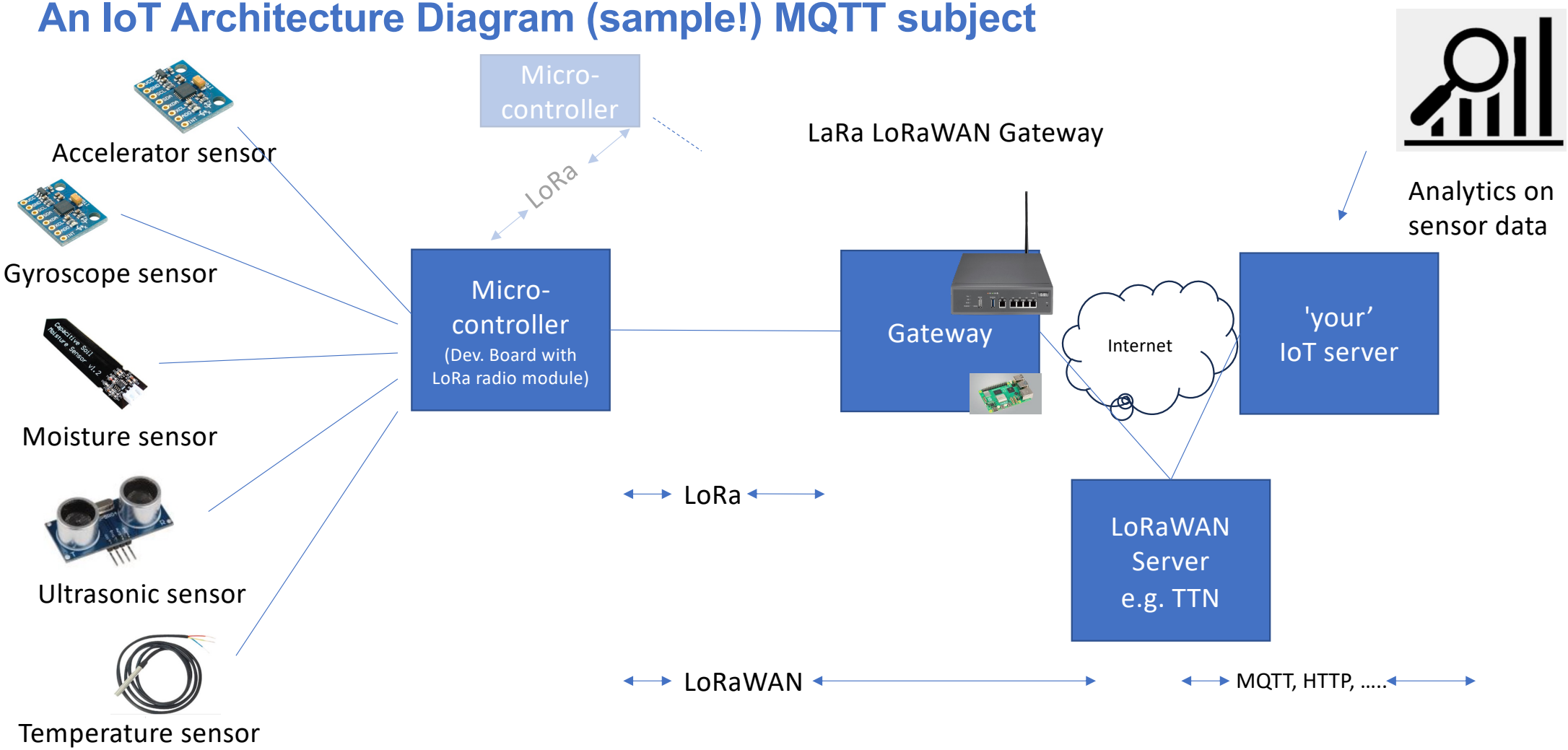
- **LoRa**
  - **Wireless modulation technology**
  - **Low bandwidth**
  - **Low battery usage**
  - **Long range**

- **LoRaWAN**
  - **Communications protocol and architecture that utilizes the LoRa physical layer**
  - **Data rates are defined that range from 300bps to 5.5kbps**
  - **with two high-speed channels at 11kbps and 50kbps (FSK modulation)**

Supports

- **secure bi-directional communication,**
- **Mobility**
- **localization.**

An IoT Architecture Diagram (sample!) MQTT subject

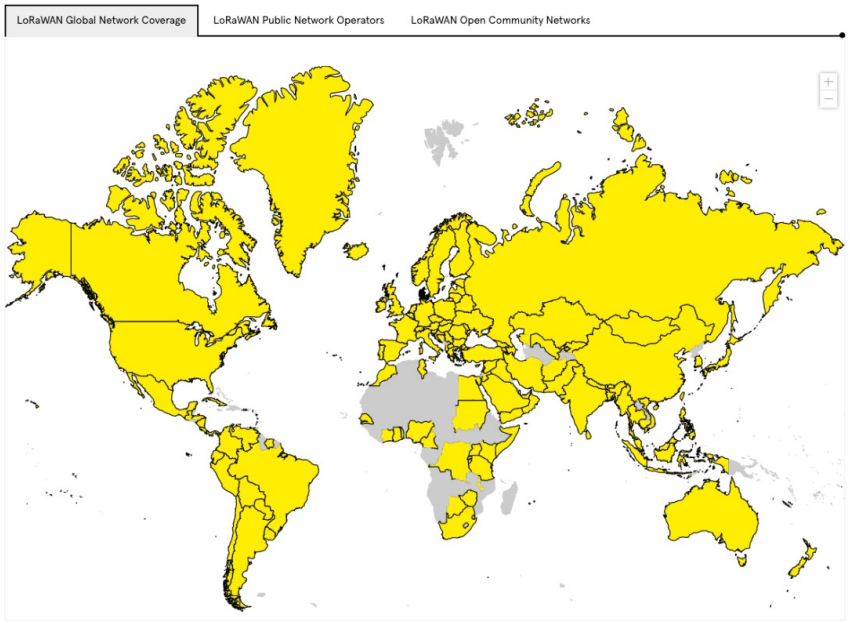


Why LoRaWAN

- Ultra low power
- Long range
- Deep indoor penetration
- License free
- Geolocation (network can determine the location of the devices)
- Public and private deployments
- End to end security (implemented in the network)
- Firmware updates over the air
- Certification program
- Large ecosystem which provides nodes, gateways .....

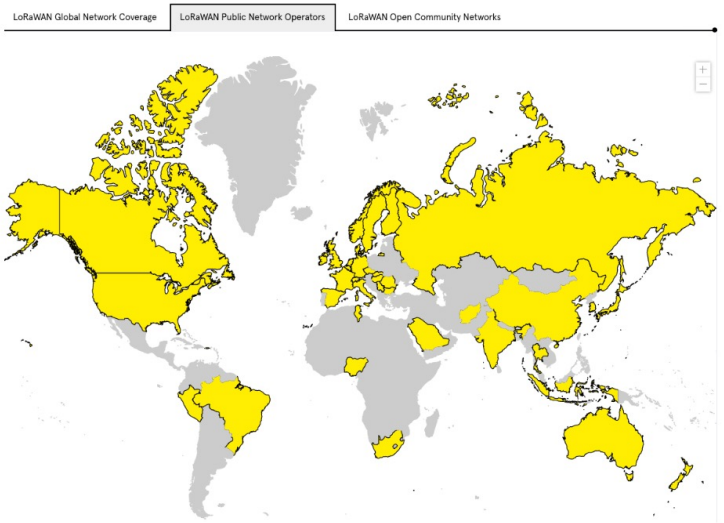
LoRaWan network coverage

181 LoRaWAN Network Operators globally.



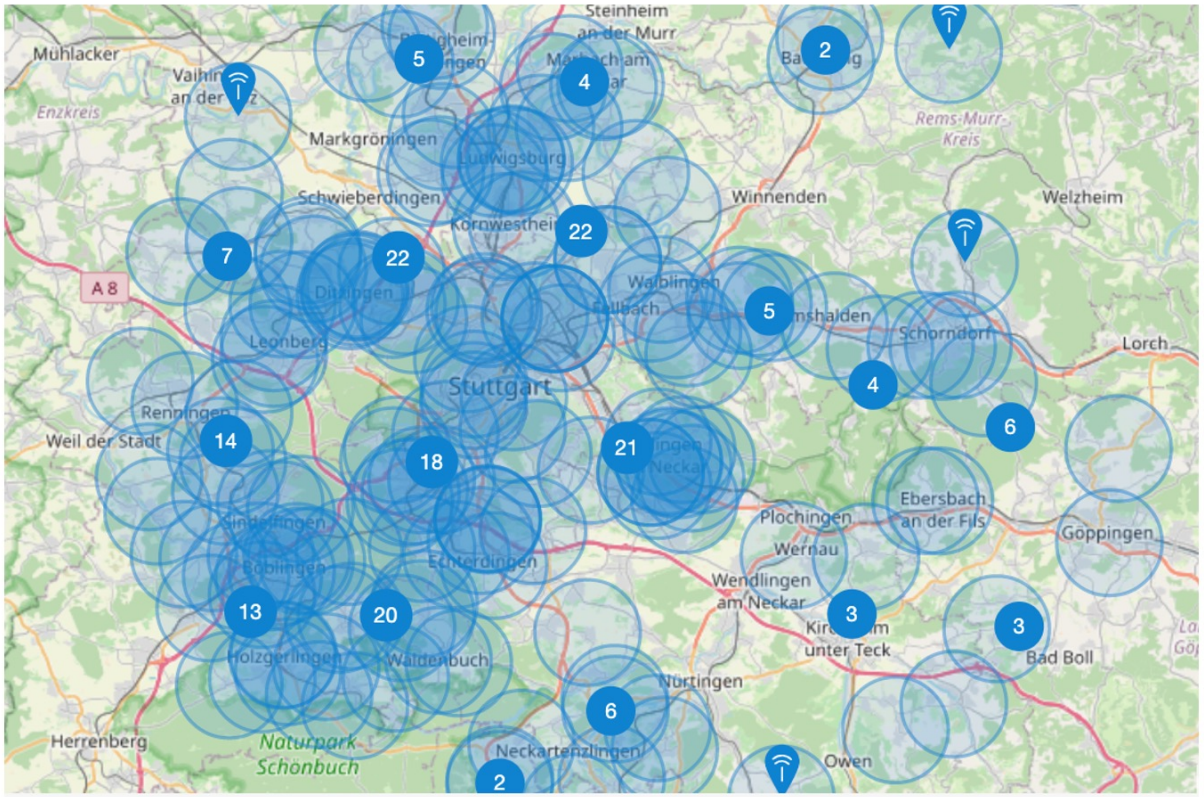
https://lora-alliance.org/

e.g.
Swiss
Telecom
KPN
Digimondo
LorIOT
The Things Network
....



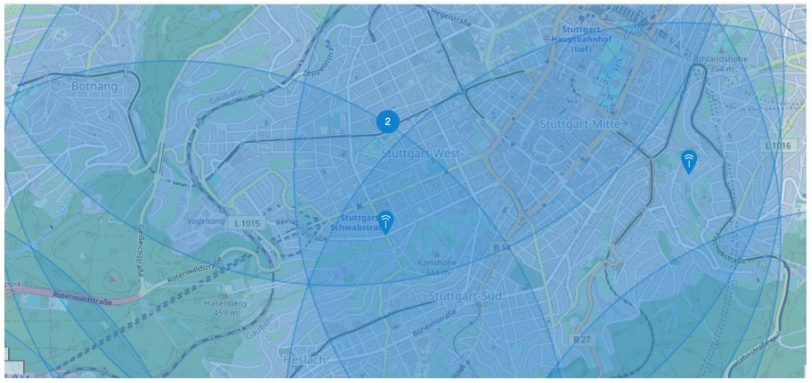
https://ttnmapper.org/heatmap/

TTN Gateways Region Stuttgart

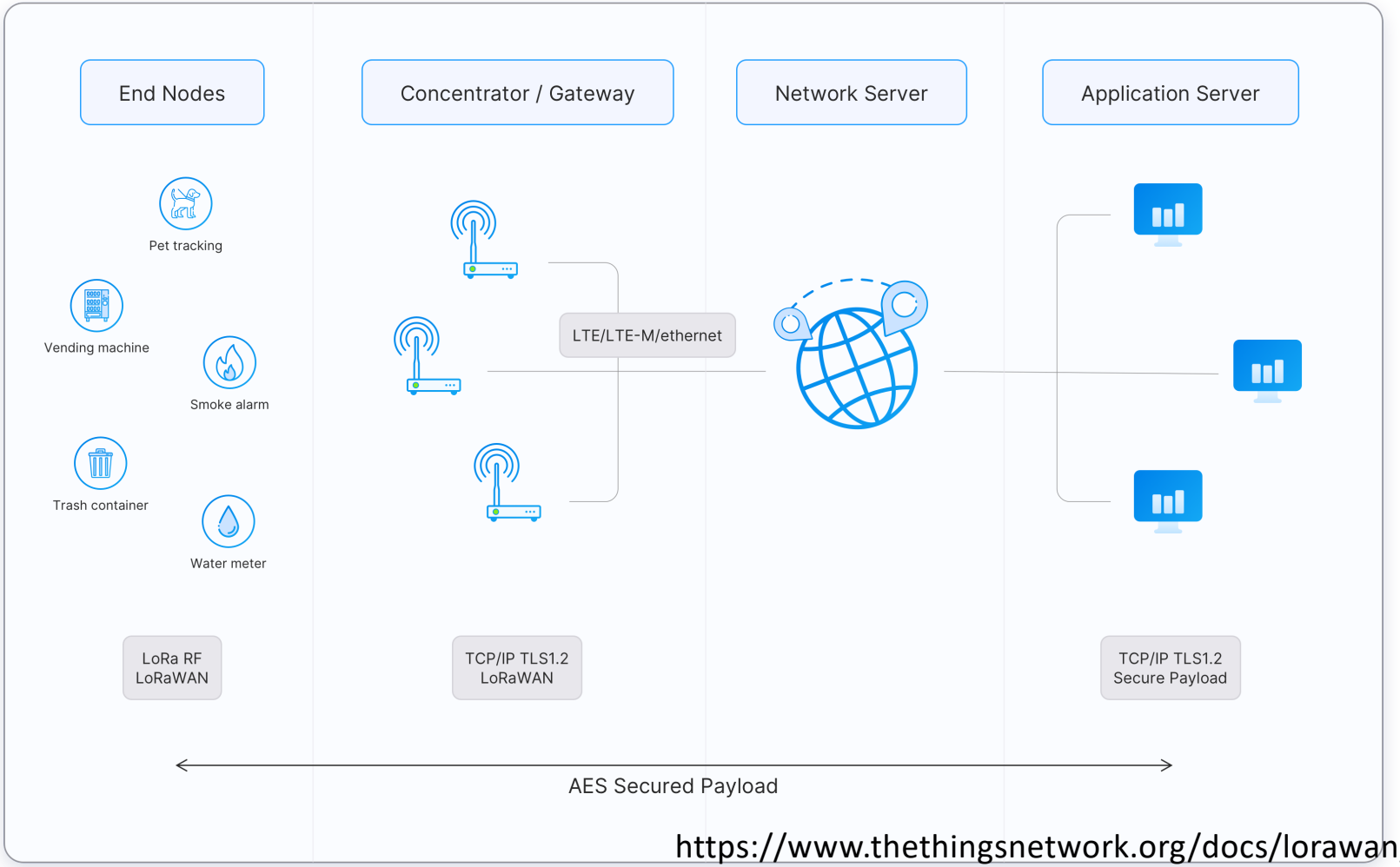


https://www.thethingsnetwork.org/community/region-stuttgart/

Stuttgart Mitte



LoRaWAN Architecture

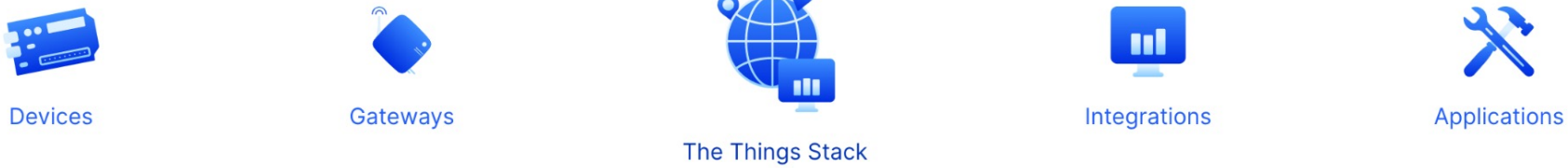


https://www.thethingsnetwork.org/docs/lorawan/architecture/

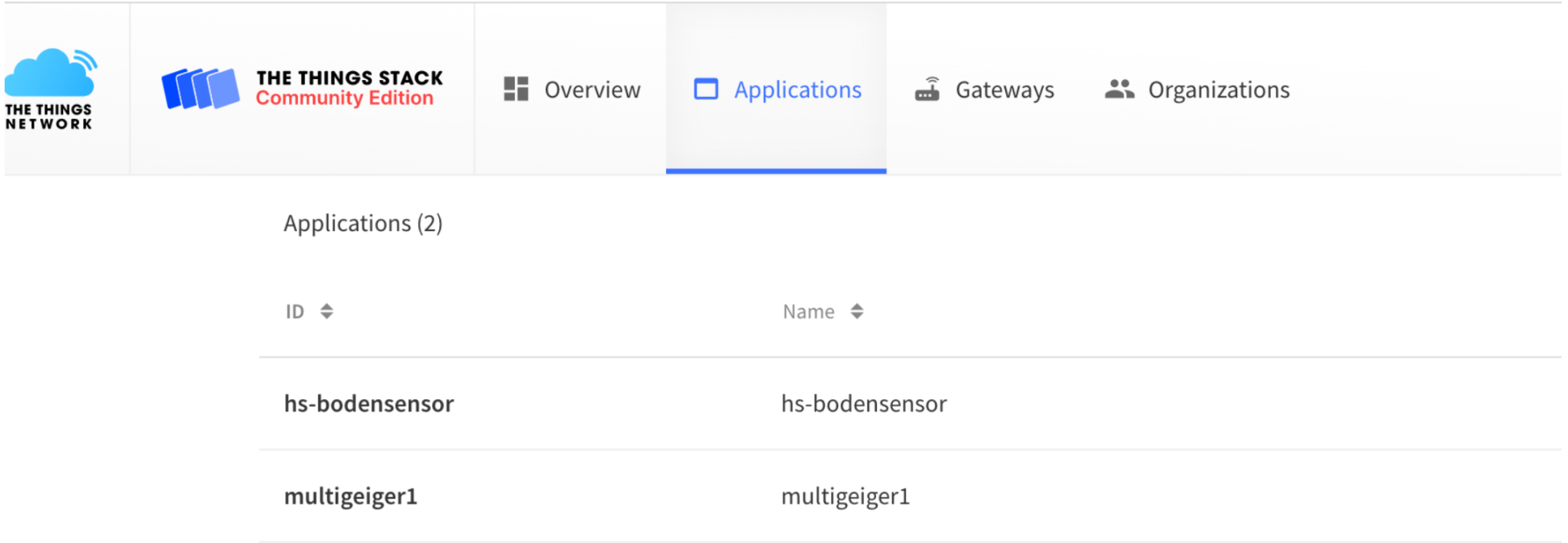
The Things Stack

Build end-to-end secured LoRaWAN® networks with 99,9%+ availability

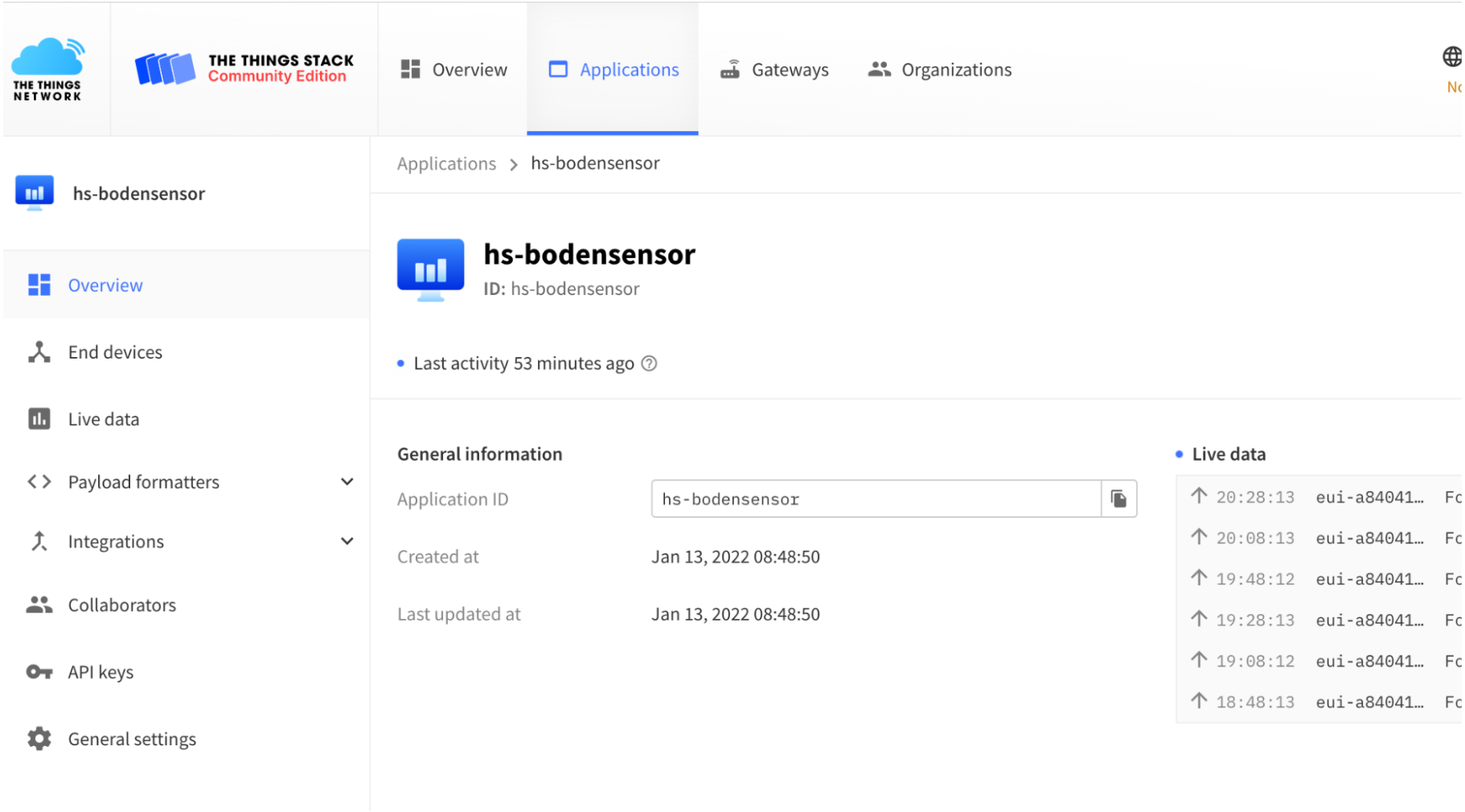
The Things Stack is a robust yet flexible LoRaWAN Network Server that caters to the needs of demanding LoRaWAN deployments, from covering the essentials to advanced security configurations and device life cycle management.



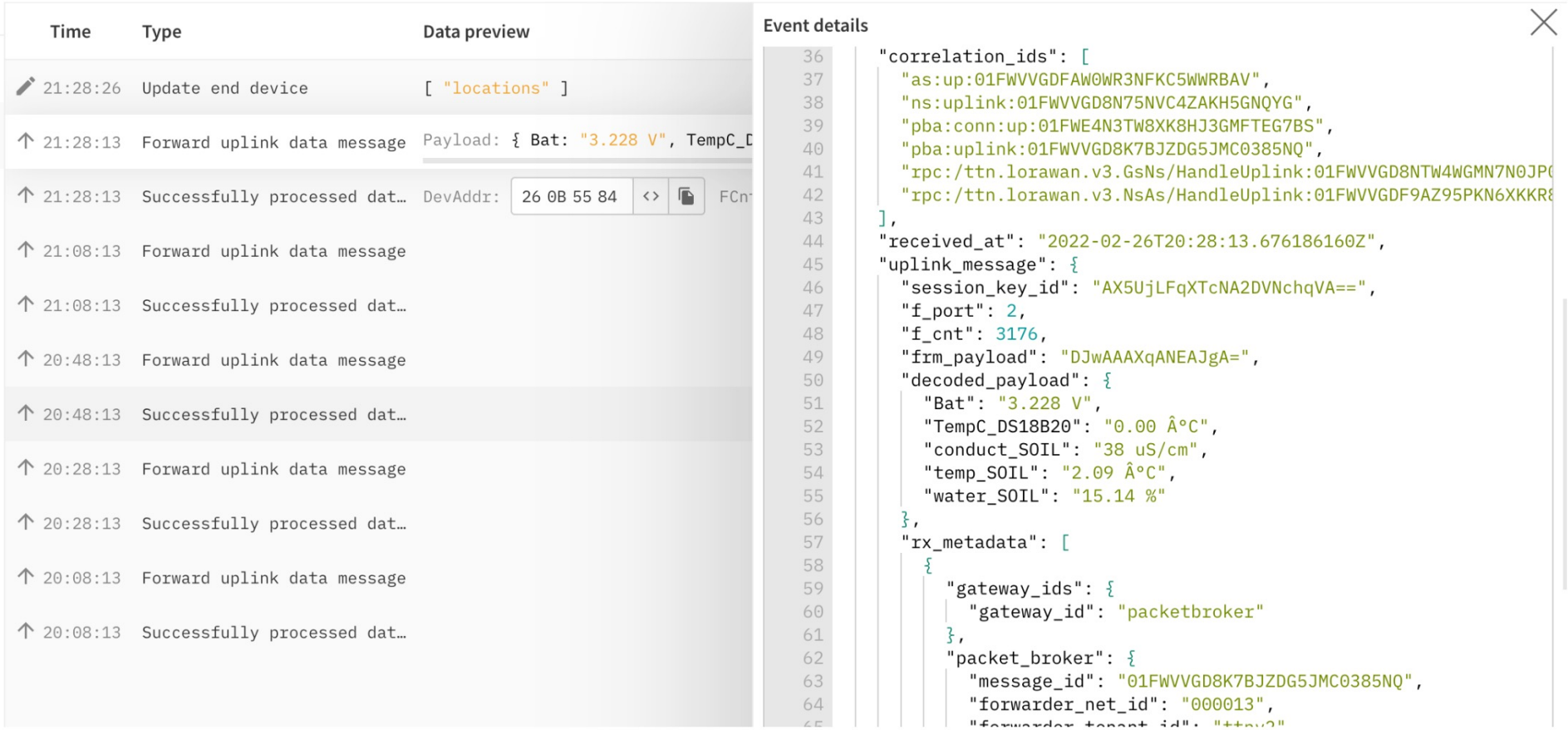
The Things Network console



The Things Network – End-Device- Live data – Payload - Integration



The Things Network – Data from an end node



Block diagram IoT Lab

Accelerator sensor

Gyroscope sensor

Moisture sensor

Ultrasonic sensor

Temperature sensor

Micro-controller

LoRaWAN gateway

TTN server

IoT server

LoRaWAN Payload

Temperature value
Humidity
Gps
…

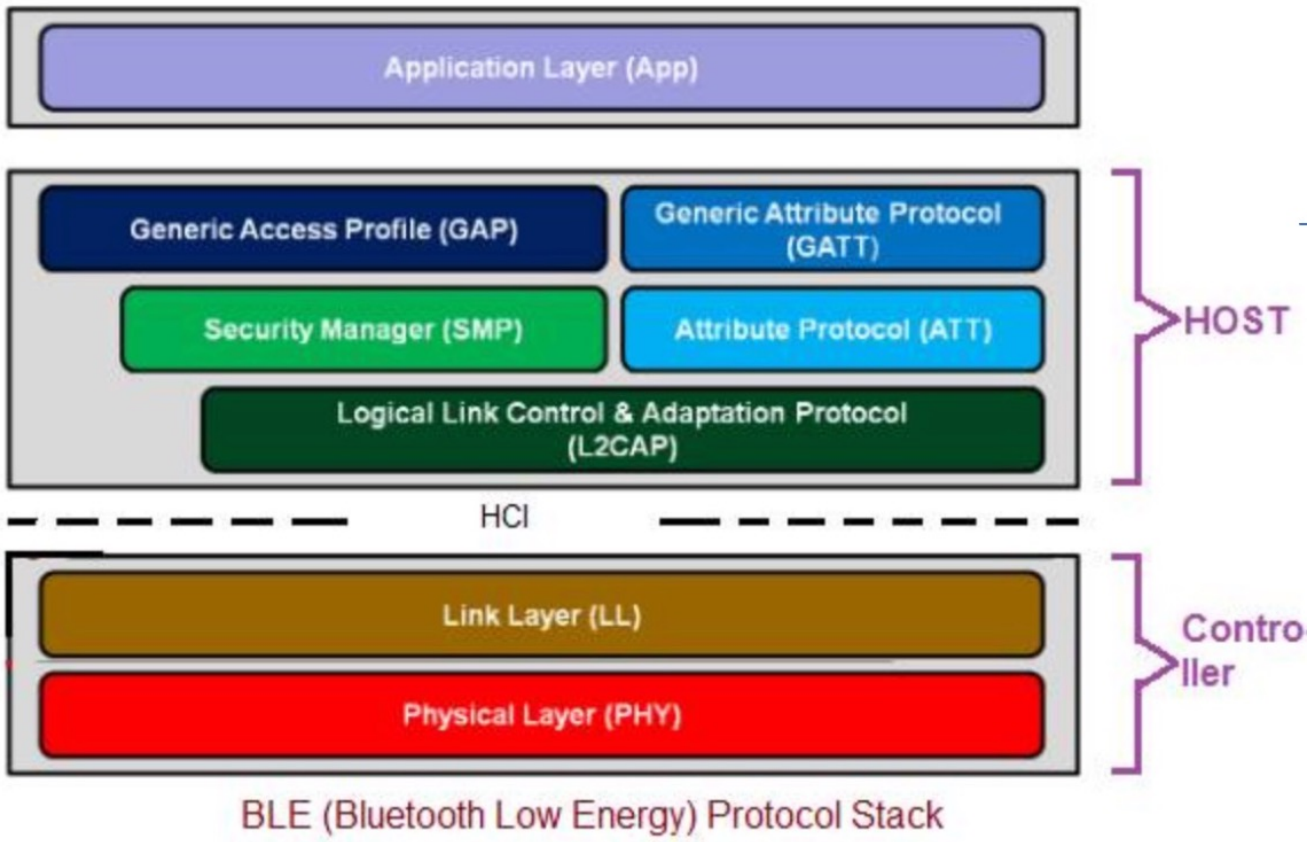
e.g. IP
MQTT
Subscribe to topic
Get LoRaWAN payload

Analytics on sensor data

A view LoRaWAN samples from previous IoT Labs

- **Wetter ‘ballon tracker’**
  - **Collect temperature, humidity, pressure and other values**
  - **Send the data via LoRaWAN to a server**
- **Zisterne – Regenwasser monitoring und Steuerung**
- **Umweltparameter Überwachung in der Stadt an verschiedenen Stellen(CO2, Temperatur, …..)**
- **Pferde auf der Weide überwachen (Futter, Wasser, Temperatur usw.)**
- **Wasserverbrauch zwischen an einer entfernten Lokation messen und Data via LoRa übertragen**
- **…..**

The BLE Architecture



BLE (Bluetooth Low Energy) Protocol Stack

Host Controller Interface
Link layer defines packed structure including state machine and radio control
Advertising, Scanning, connection establishment. ..

Radio Frequency – Physical Layer controls radio transmission/receiving
2,4GHz (40 radio frequency channels with 2 MHz spacing Min TX power -> 0,01mW, max TX power 100mW

https://www.rfwireless-world.com/Terminology/BLE-Protocol-Stack-Architecture.html

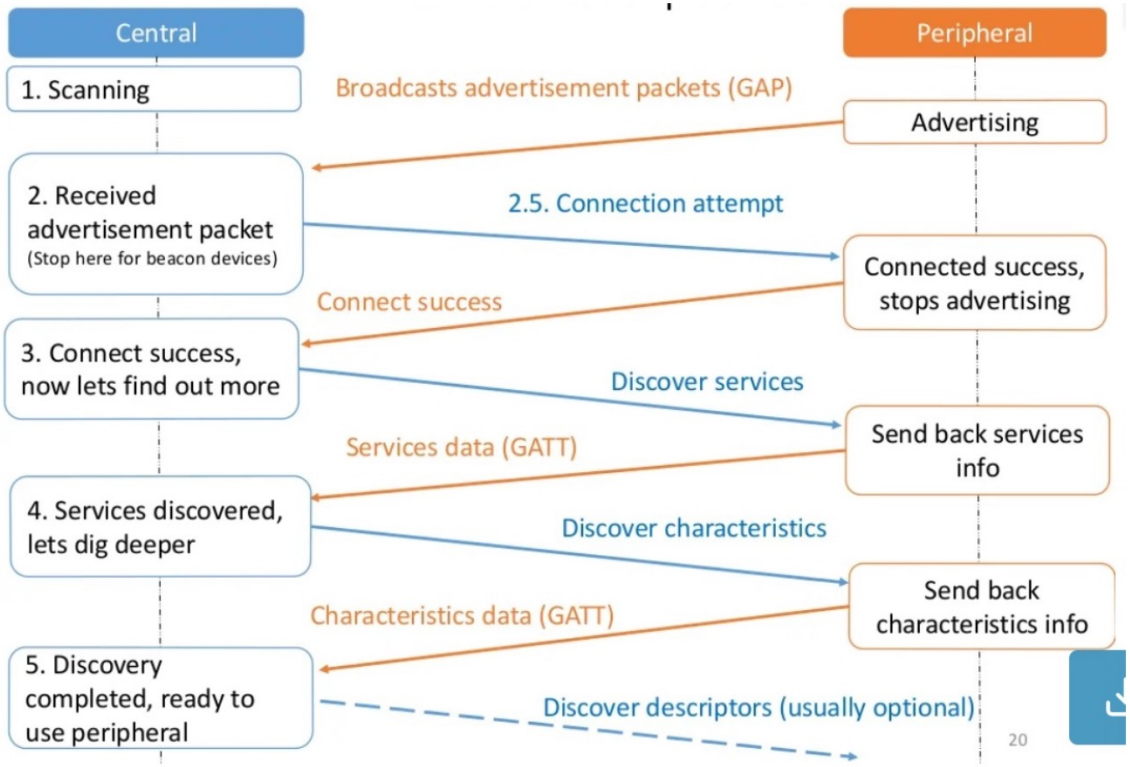
BLE Overview

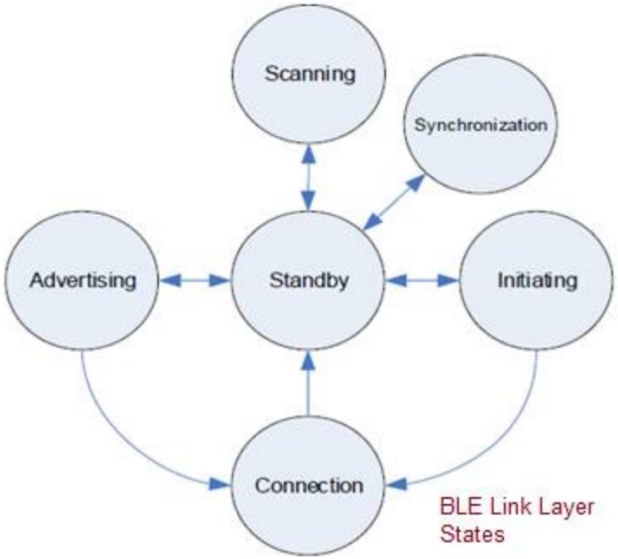
- BLE (Bluetooth Low Energy) is wireless PAN technology designed and maintained by Bluetooth Special Interest Group (SIG). There are various versions of bluetooth. The version 4.2 and above is referred as BLE. The latest in the series are v5.0 and v5.1. BLE specifications are intended to reduce power consumption and cost of devices while maintaining coverage range. BLE is known as "Bluetooth Smart" where as previous version is known as "bluetooth classic".

BLE is not backward compatible with BR/EDR protocols.

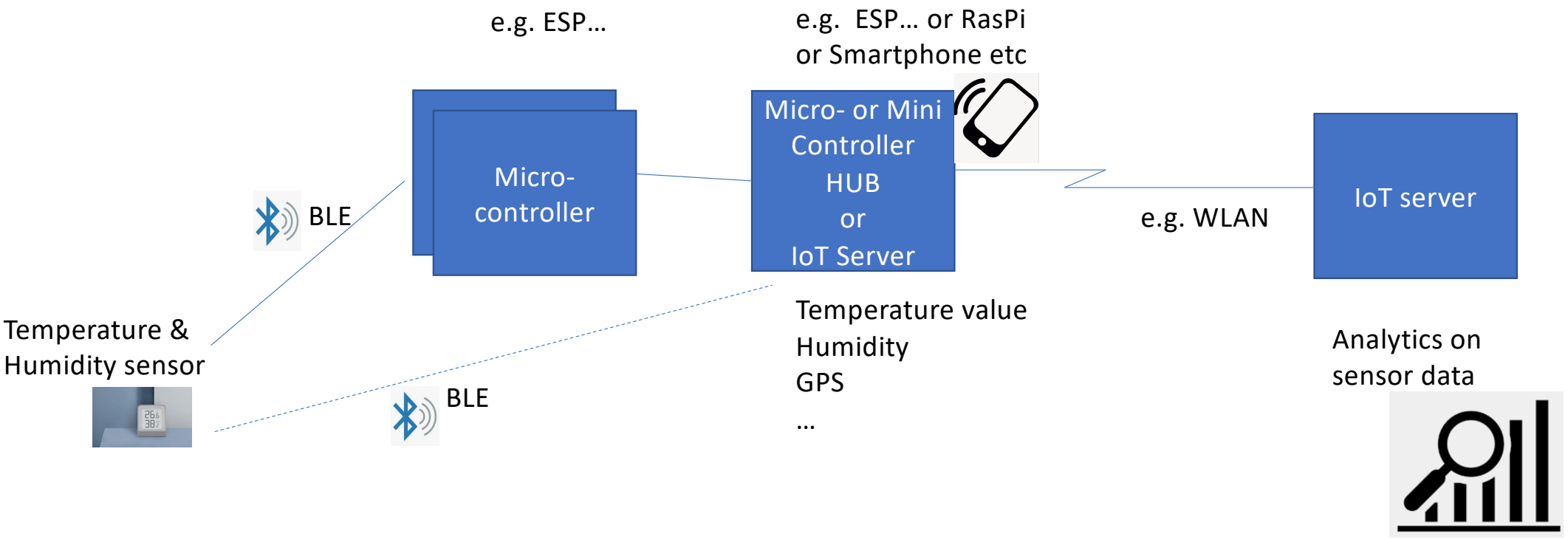
• BLE uses 2.4 GHz ISM frequency band either in dual mode or single mode. Dual mode supports both bluetooth classic and low energy peripherals.
• All BLE devices use the GATT profile (Generic Attribute Profile). The GATT protocol provides series of commands for the client to discover information about BLE server.
• The BLE protocol stack architecture consists of two parts viz. controller and host. Both are interfaced using HCI (Host to Controller Interface).
• Any profiles and applications run on top of GAP & GATT protocol layers.

BLE Connection Procedure





Block diagram IoT Lab – BLE sample



Ein Beispiel von BLE Daten eines Temperatur und Luftfeuchtigkeitssensor

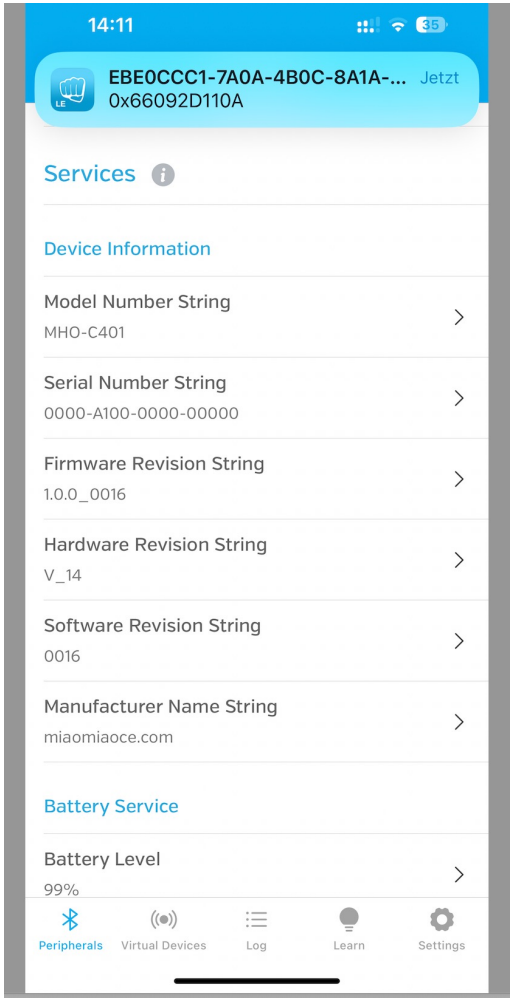
Nach dem Austausch der Characteristics und der verfügbaren Services werden die Daten übermittelt

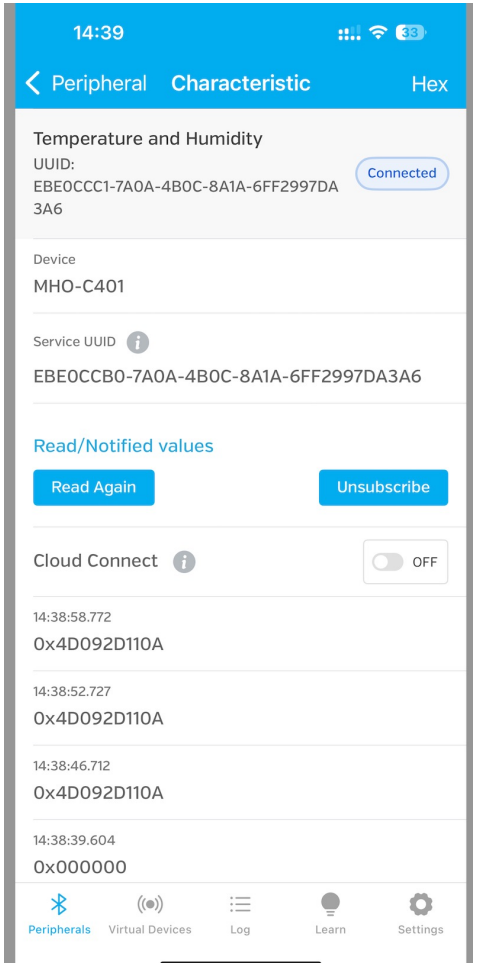
u.U. benötigt es etwas Entdeckergeist um die Daten herauszufinden (bei reverse engineering). Wenn man BLE client und Server Anwendung selber entwickelt, kennt man natürlich die Daten und das Datenformat

Meine Interpretation:

4D09 2D 110A

094d = Luftfeuchtigkeit in hex in % = 23,8%
0a11 = Temperatur = 25,77Grad

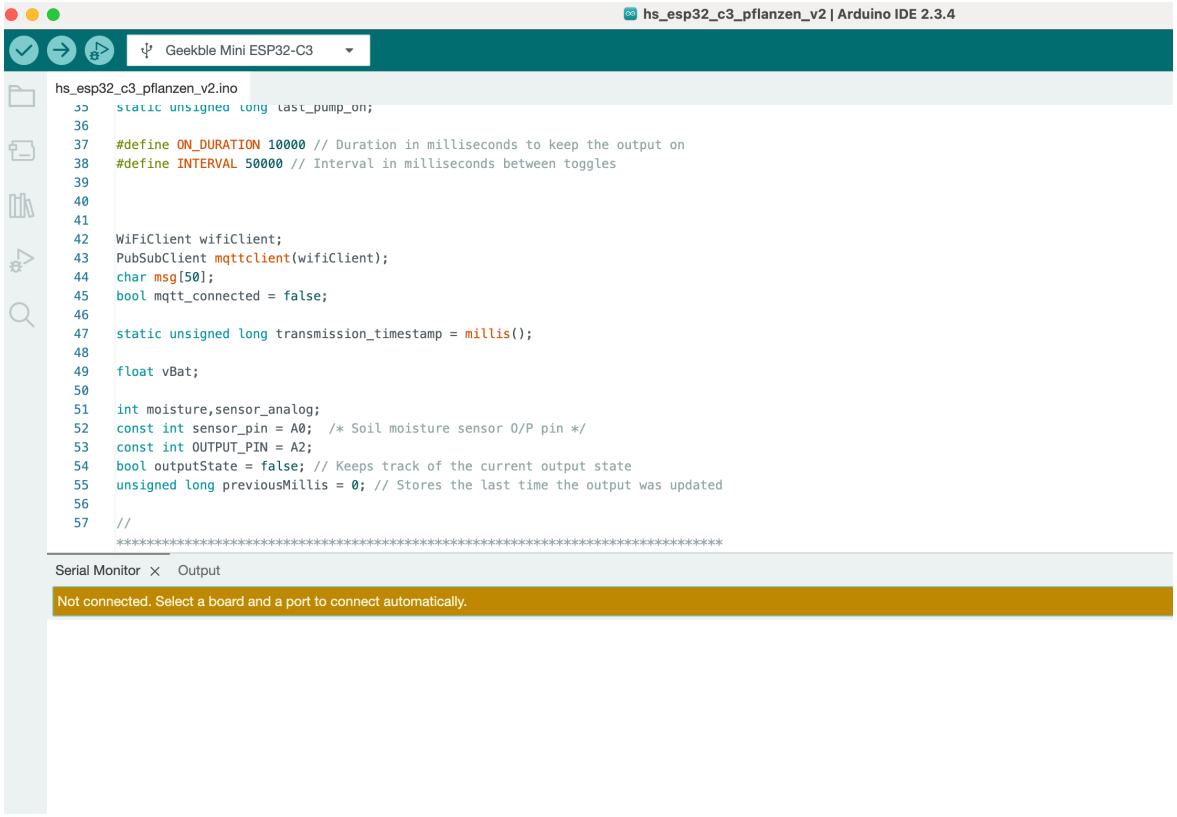





IoT
fresh and new ideas sensors, microcontroller, connectivity, AI and a lot more

Arduino Programmierungumgebung

- **https://www.arduino.cc/en/software/**



Arduino Programmierumgebung

- **Visual Studio Code und PlatformIO**