

# A Brief Introduction to LoRa™

Simon Pirkelmann



December 2nd, 2019

# Outline

## LoRa

- Motivation

- LoRa facts

- PHY layer: modulation and demodulation

- Packet format

- Data rate and air time

## Playground Part I

## LoRaWAN

- Network topology

- TheThingsNetwork

- Applications

## Playground Part II

# Motivation

- Example: monitor well-being of honey bees



# Motivation

- Example: monitor well-being of honey bees



- Requirements:

- Low data rate (a few bytes per day)
- Low power consumption
- Low cost
- Long range



# Motivation

- Example: monitor well-being of honey bees



- Requirements:

- Low data rate (a few bytes per day)
- Low power consumption
- Low cost
- Long range



Options:

- Data line (e.g. ethernet)
- Wifi, Bluetooth
- Cellular

# Motivation

- Example: monitor well-being of honey bees



- Requirements:

- Low data rate (a few bytes per day)
- Low power consumption
- Low cost
- Long range



Options:

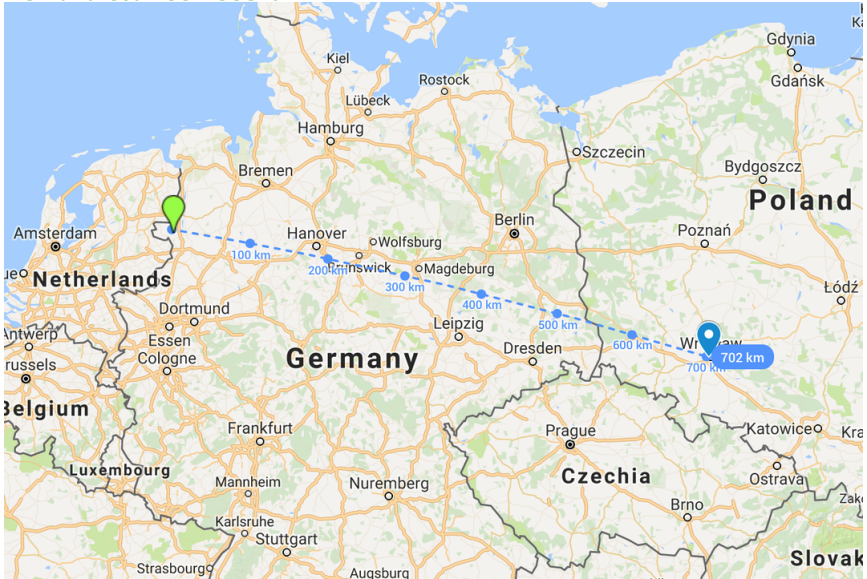
- Data line (e.g. ethernet)
  - Wifi, Bluetooth
  - Cellular
- Solution: LPWAN (**L**ow-**P**ower **W**ide-**A**rea **N**etwork)
    - **LoRa**, **LoRaWAN**, **TheThingsNetwork**
    - SigFox, NB-IoT, Weightless, ...

# LoRa Facts

- Developed by **Semtech** (originally by Cycleo)  
**Note:** parts of the PHY layer are **proprietary!**
- Frequency: **868 MHz SRD** band (in the EU)
- **25 mW** transmission power
- Bandwidth: **125 to 500 kHz**
- Data rate between **250 Bit/sec** and **21 kBit/sec**  
**BUT:** not made for a lot of data
- Low cost: **~10 EUR** CAPEX (for nodes), almost no OPEX
- Low power: devices can last **years on battery**
- Long distance: up to **10 km** range

# LoRa Facts

## LoRa distance record





# LoRa facts

LoRa is **NOT** for:

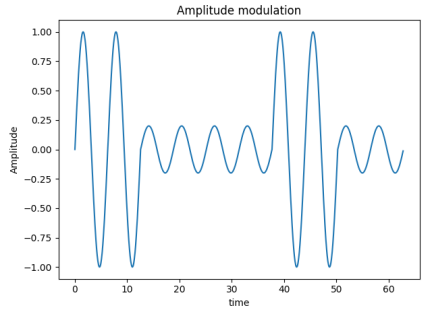
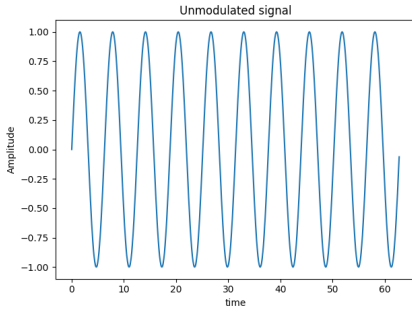
- Realtime data - only small packets, every couple of minutes
- Phone calls - you can do that with GPRS/3G/LTE
- Controlling lights in your house - check out ZigBee or BlueTooth
- Sending photos, watching Netflix - check out WiFi

Important:  LoRa™ ↔  LoRaWAN™ ↔  THE THINGS NETWORK

- **LoRa**: PHY layer ⇒ modulation technique
- **LoRaWAN**: Network protocol
- **TheThingsNetwork (TTN)**: Network server, handles routing of data to the *cloud*

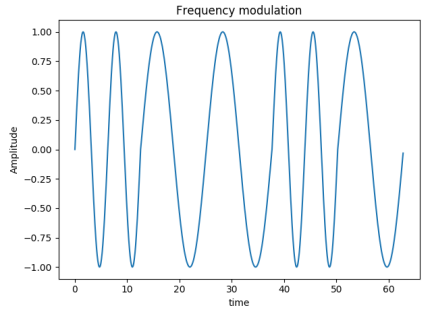
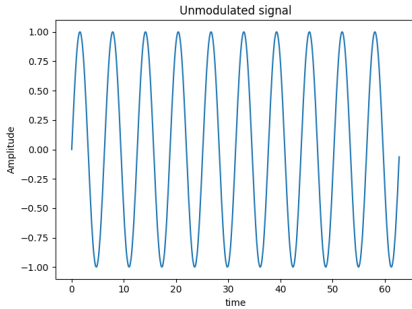
# Digital modulation

- **Amplitude Shift Keying (ASK)**



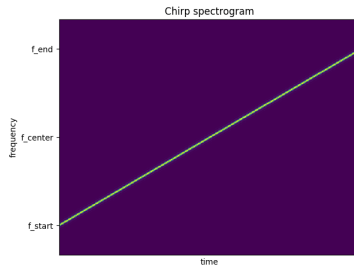
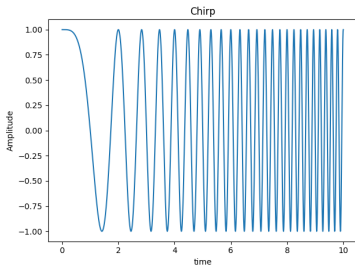
# Digital modulation

- **Frequency Shift Keying (FSK)**



# Digital modulation

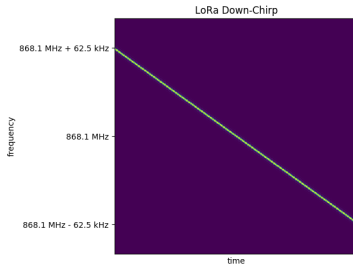
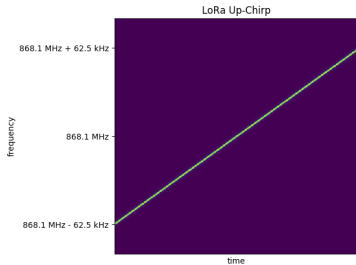
- Used by LoRa: **Chirp Spread Spectrum (CSS)** modulation



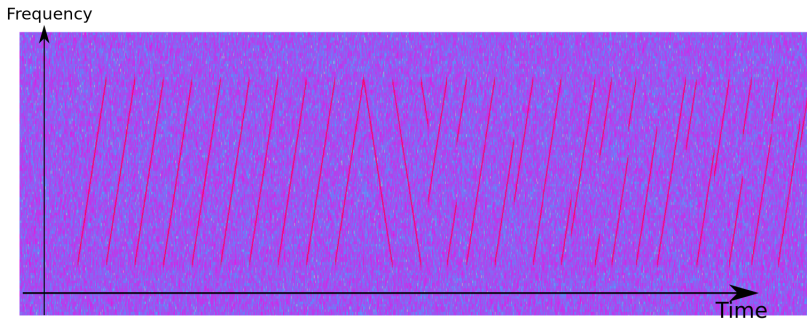
- Chirp = frequency change over time

# LoRa modulation

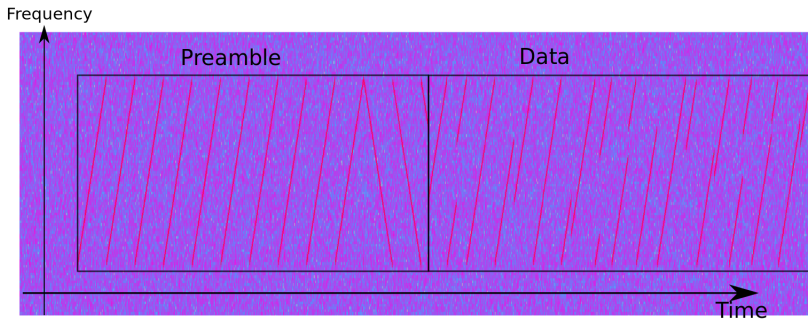
- LoRa uses **Up-Chirps** (frequency increases) and **Down-Chirps** (frequency decreases)



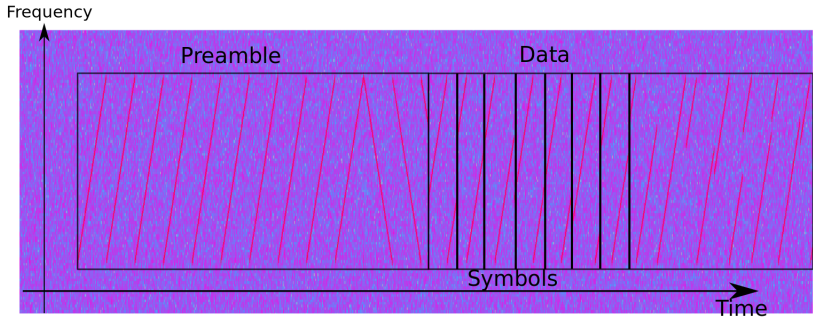
# Example LoRa packet



# Example LoRa packet

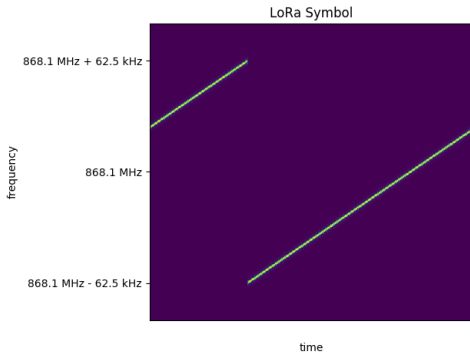


# Example LoRa packet

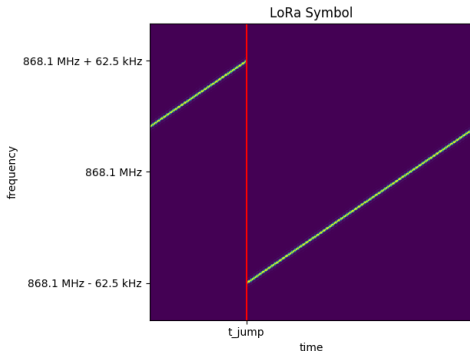




# Chirps and Symbols

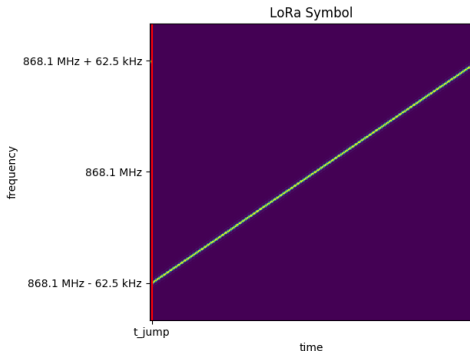


# Chirps and Symbols



Time of frequency jump determines which data is encoded

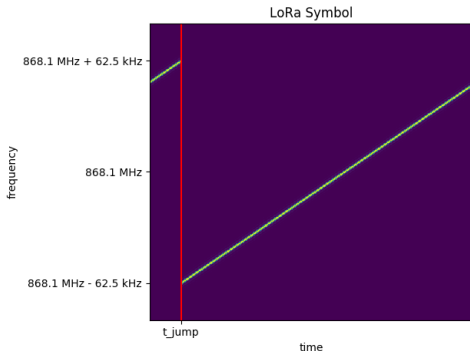
# Chirps and Symbols



Time of frequency jump determines which data is encoded

Example: data = 00000

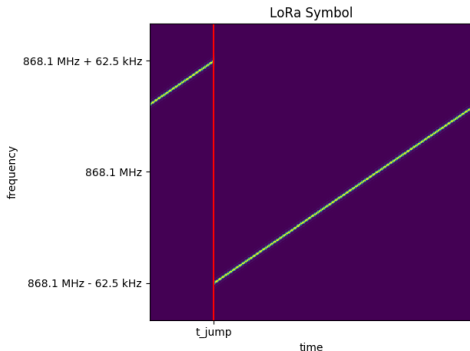
# Chirps and Symbols



Time of frequency jump determines which data is encoded

Example: data = 00010

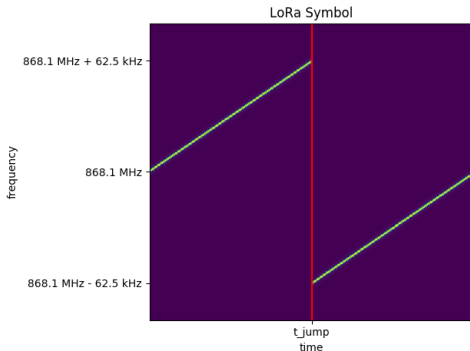
# Chirps and Symbols



Time of frequency jump determines which data is encoded

Example: data = 00011

# Chirps and Symbols

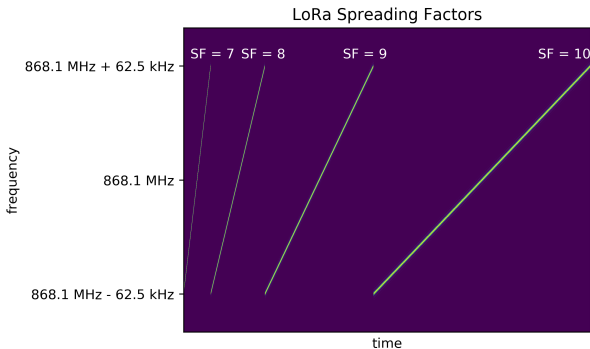


Time of frequency jump determines which data is encoded

Example: data = 10110

# Spreading factor

- Number of bits per symbol is determined by **spreading factor (SF)**



- Possible values: SF7 - SF12  
SF7: 7 bits per symbol  
SF12: 12 bits per symbol
- Spreading factor influences max. range

# Demodulation

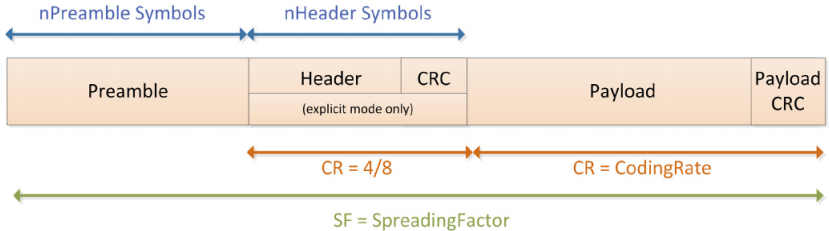
- Signal is multiplied by (mixed with) Down-Chirp
- Fourier Transform



# Forward error coding

- For each symbol several **parity bits** are added (= redundant information)  
Reason: allows to detect and fix errors that occur during transmission (due to interference, etc.)
- **Coding rate** determines number of parity bits:  
CR 4/5 : of 5 bits transmitted, 4 bits are actual data  
:  
CR 4/8 : of 8 bits transmitted, 4 bits are actual data
- Additionally: **C**yclical **R**edundancy **C**heck (CRC)

# LoRa package format



- Header: contains information about
  - payload length
  - coding rate
  - CRC present?
- Implicit header mode: no header sent

# Data rate and air time

- Data rate depends on bandwidth (BW), spreading factor (SF) and coding rate (CR)

**Symbol duration:**

$$T_{sym} = \frac{2^{SF}}{BW}$$

**Symbol rate:**

$$R_{sym} = \frac{1}{T_{sym}}$$

- Data rate:**

$$R_{data} = \underbrace{SF}_{\text{\#bits per symbol}} \cdot \underbrace{R_{sym}}_{\text{symbol rate}} \cdot \underbrace{\frac{4}{4 + CR}}_{\text{coding rate}}$$

- Example data rates:

SF7	BW250	CR4/5	$\approx 10.9$	$\frac{kbit}{s}$
SF7	BW125	CR4/5	$\approx 5.5$	$\frac{kbit}{s}$
SF12	BW125	CR4/5	$\approx 0.29$	$\frac{kbit}{s}$

# Time-On-Air

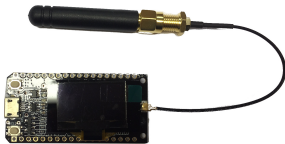
- Comply with **duty cycles** of the SRD band:

Frequency	Duty Cycle	ERP
868,0 - 868,6 MHz	1 %	25 mW

- This amounts to  $\approx 30$  seconds of transmission time per hour (maximum!). Try to keep it way below.
- Airtime calculator: <https://www.loratools.nl/#/airtime>
- Example: 20 bytes payload
  - max. 25 messages per hour on SF12
  - max. 600 messages per hour on SF7

# Playground Part I

- Module used: **Wemos® TTGO LORA32 868Mhz ESP32**
  - ESP32
  - LoRa Chip SX1276
  - OLED display
  - Antenna (needs to be connected!)
- Programmable in MicroPython thanks to uPyLora library by lemariva (<https://github.com/lemariva/uPyLora>)



```
from lora_transceiver import LoRaTransceiver
from uPySensors.ssd1306_i2c import Display

disp = Display()

# create transceiver
lora = LoRaTransceiver(display=disp)

# send a string
lora.send_string("Hello World!")

# send some raw binary data
lora.send([0x01, 0x02, 0x03])
```

- Alternative: HopeRF **RFM95W** chip

# Playground Part I

- Receiving data:

```
from lora_transceiver import LoRaTransceiver
from uPySensors.ssd1306_i2c import Display

disp = Display()

# create transceiver
lora = LoRaTransceiver(display=disp)

# start receiving data (and output on the screen)
lora.recv()
```

- Change LoRa parameters

```
# change the spreading factor
lora.setSpreadingFactor(10)

# change the frequency
lora.setFrequency(868.1e6)

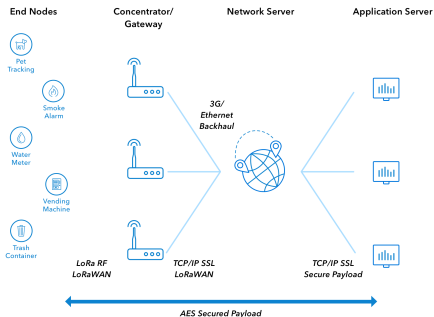
# change signal bandwidth
lora.setSignalBandwidth(250e3)

# change sync word
lora.setSyncWord(0x34)
```

- Task: *Turn off your neighbors LED!*

# LoRaWAN

- LoRaWAN is for getting your sensor data online
- Media access control (MAC) protocol
- Network topology:

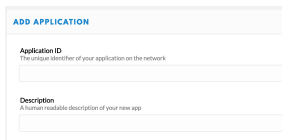


- **Gateways** forward data from nodes to the *cloud*
- Transmission is secured by **AES-128 encryption**

# Device registration

- **TheThingsNetwork** aims to build a global LoRaWAN network
- Devices need to be registered and assigned to an application before they can communicate with the network

1. Create an account on  
<https://www.thethingsnetwork.org/>
2. Log in and open the *Console*
3. Create an application

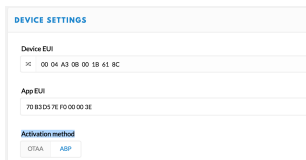


ADD APPLICATION

**Application ID**  
The unique identifier of your application on the network

**Description**  
A human readable description of your new app

4. Create a device and register it. Go to *Settings* and change activation method to Activation by Personalisation (ABP)



DEVICE SETTINGS

**Device EUI**

9C 00 04 A3 08 00 1B 61 8C

**App EUI**

7D B3 D5 7E F0 00 00 3E

**Activation method**

OTAA **ABP**



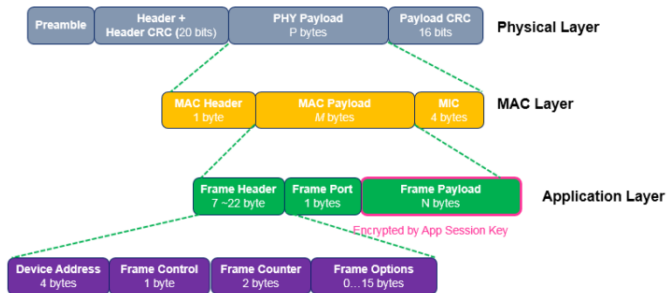
# Keys

- Three important keys (when using ABP):

Device Address ( <b>DevAddr</b> )	identification of the device in TTN
Network Session Key ( <b>NwkSKey</b> )	secure communication between device and TTN
Application Session Key ( <b>AppSKey</b> )	secure communication between device and application

- DevAddr tells TTN where to route the data
- NwkSKey used for message validity check (MIC)  
(prevents tampering with messages)
- AppSKey are used for payload encryption/decryption  
(prevents reading the data)
- Need to be **hardcoded** into the device
- Alternatively: use Over-the-Air Activation (OTAA) (more secure)
- Frame counters: Each message is equipped with a counter that prevents re-transmit attacks

# LoRaWAN payload format



Useful link:

- LoRaWAN packet decoder

<https://lorawan-packet-decoder-0ta6puiniaut.runkit.sh/>

---

<http://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/>

# Use cases

Some applications of LoRa(WAN):

- Plant monitoring:



## Playground Part II

Example code for sending data to **TheThingsNetwork**:

```
from lora_transceiver import LoRaTransceiver
from uPySensors.ssd1306_i2c import Display
import uLoRaWAN
from uLoRaWAN.MHDR import MHDR

disp = Display()

# create transceiver for LoRaWAN frequency (channel 0 = 868.1 Mhz)
lora = LoRaTransceiver(frequency=868.1E6, syncword=0x34, display=disp)

# set address and keys for LoRaWAN (with ABP)
devAddr = [0x26, 0x01, 0x16, 0x5C] # ir_test_device_01
nwksKey = [0x9D, 0x95, 0x0F, 0xAB, 0xCB, 0x63, 0xD3, 0x04, 0xBC, 0x09,
           0xC4, 0x9E, 0xC5, 0xDF, 0x3C, 0x37]
appSKey = [0xC9, 0x6C, 0x00, 0xD1, 0xB0, 0x1C, 0x2E, 0x42, 0x11, 0xBA,
           0x32, 0x6F, 0x2F, 0xC2, 0x75, 0x6A]

# lorawan object for conversion of data in LoRaWAN message format
lorawan = uLoRaWAN.new(nwksKey, appSKey)

message = list(map(ord, 'Hello World!')) # convert to bytes
lorawan.create(MHDR.UNCONF_DATA_UP, { 'devaddr': devAddr, 'fcnt': 0,
                                       'data': message })
payload = lorawan.to_raw()

lora.send(payload)
```

(the code is based on the uLoRaWAN library by mallagant, see:  
<https://github.com/mallagant/uLoRaWAN>)

## Playground Part II

For testing you can use the following device addresses and keys:

- **nwkSKey**

```
9D 95 0F AB CB 63 D3 04 BC 09 C4 9E C5 DF 3C 37
```

- **appSKey**

```
C9 6C 00 D1 B0 1C 2E 42 11 BA 32 6F 2F C2 75 6A
```

- **devAddr** (choose one):

```
1: 26 01 16 5C      10: 26 01 1A CE
2: 26 01 18 52      11: 26 01 12 F1
3: 26 01 1E 4F      12: 26 01 1B 18
4: 26 01 17 83      13: 26 01 19 40
5: 26 01 1B 5C      14: 26 01 19 96
6: 26 01 1E B5      15: 26 01 18 FF
7: 26 01 13 DA
8: 26 01 1E 8F
9: 26 01 18 F1
```

- Watch incoming data (forwarded as HTTP POST request) at  
<https://t1p.de/ocp6>  
and check the gateway logs at (connected to @BayernWLAN  
wifi)  
<http://192.168.1.1:1337>

# References and more information

## References:

- Decoding LoRa <https://revspace.nl/DecodingLora>
- *LoRa und The Things Network* - talk by Hubert Högl (FH Augsburg)
- Stackexchange thread about LoRa symbols
- LoRa talk
- Mobilefish.com LoRa youtube tutorials

## Further reading:

- TTN Applications: APIs, Python SDK, Integrations  
<https://www.thethingsnetwork.org/docs/applications/>
- Best practices to reduce payload size: <https://www.thethingsnetwork.org/forum/t/best-practices-to-limit-application-payloads/1302>
- Forum about all things LoRa:  
<https://www.thethingsnetwork.org/forum/>
  - Gateway guides
  - Radio module/antenna recommendations
  - ...
- Cayenne Low Power Payload (LPP)

# References and more information

## References:

- Decoding LoRa <https://revspace.nl/DecodingLora>
- *LoRa und The Things Network* - talk by Hubert Högl (FH Augsburg)
- Stackexchange thread about LoRa symbols
- LoRa talk
- Mobilefish.com LoRa youtube tutorials

## Further reading:

- TTN Applications: APIs, Python SDK, Integrations  
<https://www.thethingsnetwork.org/docs/applications/>
- Best practices to reduce payload size: <https://www.thethingsnetwork.org/forum/t/best-practices-to-limit-application-payloads/1302>
- Forum about all things LoRa:  
<https://www.thethingsnetwork.org/forum/>
  - Gateway guides
  - Radio module/antenna recommendations
  - ...
- Cayenne Low Power Payload (LPP)

**Thanks for your attention!**