IoT Engineering
8: Long Range Connectivity with LoRaWAN

CC BY-SA, Thomas Amberg, FHNW
(unless noted otherwise)
Slides: tmb.gr/iot-8

Overview

These slides introduce *LoRaWAN connectivity*.

How the network is built from nodes and gateways.

Sending small amounts of data over a long distance.

Getting the data to an application from the backend.

Prerequisites

Set up the Feather nRF52840 Express for Arduino.

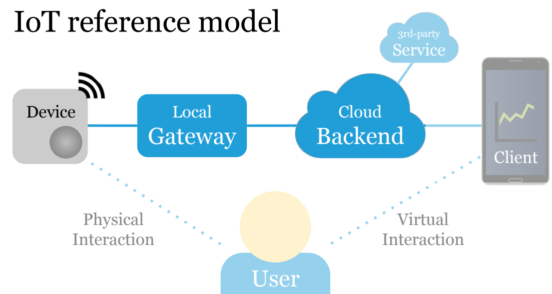
Or set up the Feather Huzzah ESP8266, both work.

A LoRaWAN gateway has to be in range for testing*.

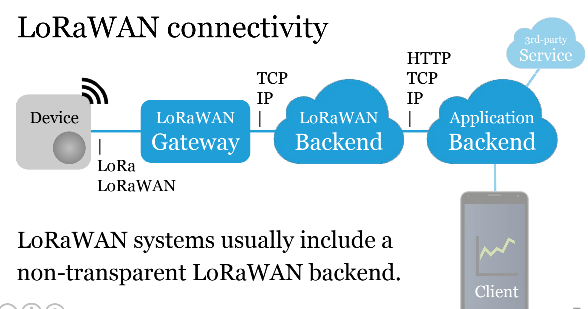
The Raspberry Pi with Node.js is our app backend.

*See, e.g. thethingsnetwork.org gateway map.

IoT reference model



LoRaWAN connectivity



LoRaWAN systems usually include a non-transparent LoRaWAN backend.

LoRa

LoRa is a digital wireless communication technology.

The LoRa physical layer protocol is proprietary.

Semtech, the owner, sells LoRa transceivers.

LoRa radio is long range* and low power.

*Around 1 km in cities, 10+ km in open terrain.

LoRaWAN

LoRaWAN is a low power*, wide area networking protocol (LPWAN) based on the LoRa physical layer.

The LoRaWAN specification (v1.0.3) is developed by the LoRa Alliance, a non-profit industry consortium.

LoRaWAN defines link layer parameters, addressing, a transport protocol, and the network architecture.

*RFM95W 10/30 mA vs. ESP8266 50/150 mA.

LoRaWAN terminology

The LoRaWAN community uses the following terms:

Node — device with sensors, LoRaWAN connectivity.

Gateway — LoRaWAN (to Internet) gateway.

Network server — LoRaWAN backend.

Application server — app backend.

LoRaWAN frequencies

LoRaWAN uses frequencies in license-free bands.

Frequencies depend on the geographic region.

EU 868 MHz, US 915 MHz, Asia 433 MHz, ...

There are frequency plans, per country*.

*Based on the regional parameters specification.

LoRaWAN network providers

There are various ways to get LoRaWAN coverage, e.g.

LoRaWAN network providers like Swisscom (Actility).

LoRaWAN backend/solution providers like Loriot.

Open infrastructure like The Things Network.

This course uses The Things Network.

The Things Network (TTN)

TTN is an open source project started in Amsterdam.

Everybody can put up a gateway to extend coverage.

Everybody can get an account and register devices.

The network is open, but your data stays private.

TTN has regional communities, e.g. TTN Zürich.

Mapping network coverage

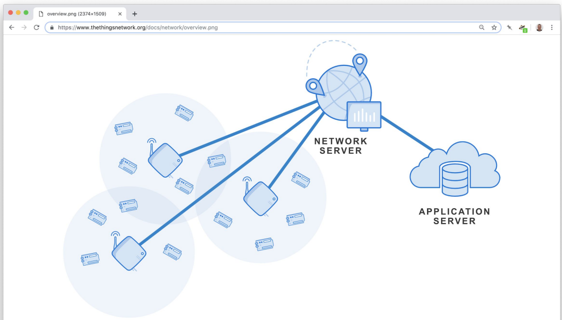
TTNMapper is a community effort to map coverage.

The iOS or Android app provides GPS location data.

The data is correlated with metadata from gateways.

Any LoRaWAN device sending* packets works fine.

*Actually *broadcasting*, as LoRa is radio.



LoRaWAN gateways

Gateways forward *uplink* data packets to the backend.

There, they are deduplicated & routed to applications.

Downlink packets are "broadcast" to nearby devices.

Everybody can operate a gateway in unlicensed band.

See products, indoor & outdoor, or build your own.

LoRaWAN security

LoRaWAN transport security is based on 128 bit keys:

Network Session Key — admits a device to a network.

Application Session Key — encrypts/decrypts payload.

These keys are unique per device and per "session".

They are generated with OTAA, or static with ABP.

See, e.g. TTN security docs and this whitepaper.

Over The Air Activation (OTAA)

OTAA uses an *AppKey* to generate keys per session.

- Device has a *DevEUI*, *AppEUI* and *AppKey*
- Device sends a *Join Request*, uses *Join Response* and *AppKey* to derive an *AppSKey* and *NwkSKey*
- Device must be able to store the generated keys
- Join decision can be delegated to a *Join Server*

See *End-device activation* in the LoRaWAN spec.

Activation by Personalization (ABP)

ABP stores application and network session keys.

- Device has a *Device Address*, *AppSKey* & *NwkSKey*
- No *DevEUI*, *AppEUI* or *AppKey* is needed here
- There is no *Join*, the device just sends data
- Overall ABP is simpler, but less flexible
- Changing the provider is not possible

See, e.g. LoRaWAN OTAA or ABP?

Registering an application on TTN

An *application* is required to register devices later on.

On The Things Network, to register a new application:

- Open https://console.thethingsnetwork.org/
- Go to *Applications* > *Add application*
- Enter a name, e.g. `fhnw-iot`
- Click *Add application*

The steps are similar for most backend providers.

Registering a device on TTN

On The Things Network, to register a new device:

- Open https://console.thethingsnetwork.org/
- Go to *Applications* > click, e.g. `fhnw-iot`
- Click *Register device*
- Enter a *Device ID*, e.g. `fhnw-iot-arduino-0`
- Click the *Device EUI* icon, so it *will be generated*
- Click *Register*

Getting OTAA keys on TTN

On The Things Network, to get keys for OTAA:

- Open https://console.thethingsnetwork.org/
- Go to *Applications* > click, e.g. `fhnw-iot`
- Go to *Devices* > click, e.g. `fhnw-iot-arduino-2`
- OTAA is the default, device registration generates a *Device EUI*, and sets *Application EUI* and *App key*

Use either OTAA or ABP depending on the code.

Getting ABP keys on TTN

- Open https://console.thethingsnetwork.org/
- Go to *Applications* > click, e.g. `fhnw-iot`
- Go to *Devices* > click, e.g. `fhnw-iot-arduino-2`
- Go to *Settings* > as *Activation Method* click *ABP*
- Deactivate *Frame Counter Checks* (testing only!)
- Click *Save*
- This generates a *Device Address* as well as a *Network Session Key* and *App Session Key*

LoRaWAN hardware modules

Some LoRaWAN modules, based on Semtech SX127*x*:

RN2483 — via UART/AT commands (or stand-alone).

RFM95W — via SPI, stack runs on separate controller.

Murata — SoC including an ARM STM32 Cortex M0.

Always make sure the frequency fits your region.

FeatherWing RFM95W

RFM95W is a popular 868 MHz LoRa radio module.

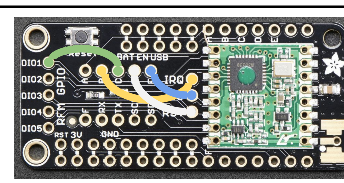
The RFM95W FeatherWing needs a microcontroller.

Both Feather boards work, nRF52840 and ESP8266.

The pin mapping has to be adapted in the code.

Note: Always add an antenna before using it.

Jumpers



These FeatherWing RFM95W jumpers must be connected.

```
IRQ  – B
DIO1 – C
RST  – D
CS   – E
```

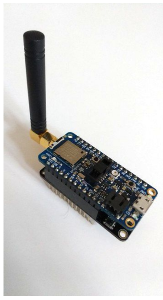
Here is a simple adapter PCB to replace jumpers.

Setup

The nRF52840 goes on top* of the FeatherWing.

The ESP8266 fits below* the wing.

*Depending on the headers used.



nRF52840 pin mapping

```
const lmic_pinmap lmic_pins = { // nRF52840
  .nss = 5, // E = CS
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 6, // D = RST
  .dio = {
    10, // B = DIO0 = IRQ
    9,  // C = DIO1
    LMIC_UNUSED_PIN
  } };
```


ESP8266 pin mapping

```
const lmic_pinmap lmic_pins = { // ESP8266
  .nss = 2, // E = CS
  .rxtx = LMIC_UNUSED_PIN,
  .rst = 16, // D = RST
  .dio = {
    15, // B = DIO0 = IRQ
    0,  // C = DIO1
    LMIC_UNUSED_PIN
  } };
```


Arduino LoRaWAN

The following examples work on a range of boards*.

LMIC is a network stack library used with RFM95W.

We use the IBM LMIC library (v1.5.0+arduino-2).

Library Manager doesn't work, get the .ZIP and use

Sketch > Include Library... > Add .ZIP Library...

*Including nRF52840 and ESP8266.

Arduino LoRaWAN ABP .ino^ESP, .ino^nRF

Set the *NwkSKey*, *AppSKey* and *Device Address*:

```
static const PROGMEM u1_t NWKSKEY[16] = {...}
static const u1_t PROGMEM APPSKEY[16] = {...}
static const u4_t DEVADDR = 0x01234567;
```

Double check to use the pin mapping for your board:

```
const lmic_pinmap lmic_pins = {...} // nRF52…
```

Set a custom message:

```
static uint8_t mydata[] = "Hello, world!";
```


Arduino LoRaWAN OTAA .ino^ESP, .ino^nRF

Set the *AppEUI*, *DevEUI* and *AppKey*:

```
static const u1_t PROGMEM APPEUI[8]= { … }
static const u1_t PROGMEM DEVEUI[8]= { … }
static const u1_t PROGMEM APPKEY[16] = { … }
```

Double check to use the pin mapping for your board:

```
const lmic_pinmap lmic_pins = {...} // nRF52…
```

Set a custom message:

```
static uint8_t mydata[] = "Hello, world!";
```

Hands-on, 15': Arduino LoRaWAN

Get an account at https://thethingsnetwork.org/

Register an application with two (Arduino) devices.

Get ABP keys for one device, OTAA keys for another.

Run the previous Arduino LoRaWAN *.ino* examples.

Make sure to set the pinout, keys in the source.

Uplink and downlink

Uplink — sending data from a device to the backend.

Downlink — sending from the backend to a device.

There's an *asymmetry* due to duty cycle limitations.

Gateways are *half-duplex*, they either send or listen.

LoRaWAN is better suited to send data *to* the cloud.

MQTT integration

The TTN backend is also an MQTT broker/proxy.

To get uplink packets from a device:

```
$ mqtt sub -t "<AppID>/devices/<DevID>/up" \
-h "eu.thethings.network" -u "<AppID>" \
-P "<AppAccessKey>" # see TTN console, apps
```

To send a packet downlink, Base64 encoded:

```
$ mqtt pub -t "<AppID>/devices/<DevID>/down" \
-m '{"port":1,"payload_raw":"<Bytes>"}' -h …
```


HTTP Webhook integration

The TTN backend provides a RESTful HTTP API.

A PUT request allows to send packets downlink.

A *Webhook* URL can be set to receive uplink data.

The TTN backend calls this URL for each packet.

The backend also defines the JSON data format.

How to debug Webhook calls

To debug Webhook calls, set up a simple Web service:

```
$ wget https://bitbucket.org/tamberg/\
iotworkshop/raw/tip/NodeJS/http-logger.js
$ node http-logger.js # runs on 127.0.0.1:8080
```

Make it accessible via Ngrok, PageKite or Yaler relay.

=> URL, e.g. `https://RELAY_DOMAIN.try.yaler.io/`

Set this URL as Webhook URL, watch the shell.

Product-specific integrations

LoRaWAN backends (here TTN) provide product specific integrations with 3rd-party services.

On The Things Network, to create a new integration:

- Open https://console.thethingsnetwork.org/
- Go to *Applications* > click, e.g. `fhnw-iot`
- Go to *Integrations* > click *Add integration*

Hands-on, 15': TTN integrations

Read the TTN HTTP and MQTT data API docs.

Use the Raspberry Pi as an application backend.

Set up an HTTP Service to log TTN Webhook calls.

Run a MQTT (sub) client to log incoming messages.

Data formats

Bandwidth is very limited, payload is ≤ 51 Byte.

JSON or plain ASCII formats use too much space:
`{"temperature":20.63}` vs. `20.63` vs. `0x080F`

The TTN backend has payload decoders & encoders.

TTN works well with the CayenneLPP binary format.

Consuming less Bytes means sending more often.

Limitations

LoRaWAN has physical, legal & operator limitations:

Duty cycle limitations allow only 1% air time in EU*, apply to nodes *and* gateways, creating asymmetry.

The TTN Fair Access Policy limits uplink air time to 30 s and downlink to 10 messages per 24 h per node.

*See ETSI EN300.220 standard, 7.2.3.

Hands-on, 15': LoRaWAN use cases

Which applications become possible with LoRaWAN?

Does free wide area connectivity change anything?

Sketch the reference model for an application.

Find a case that clearly beats Wi-Fi, 3/4G.

Summary

LoRaWAN brings long range, low power connectivity.

We learned about gateways and network architecture.

We sent packets uplink, from a device, and downlink.

We understand how data arrives at the app backend.

Next: Dashboards and Apps for Sensor Data.

Feedback or questions?

Write me on https://fhnw-iot.slack.com/

Or email thomas.amberg@fhnw.ch

Thanks for your time.