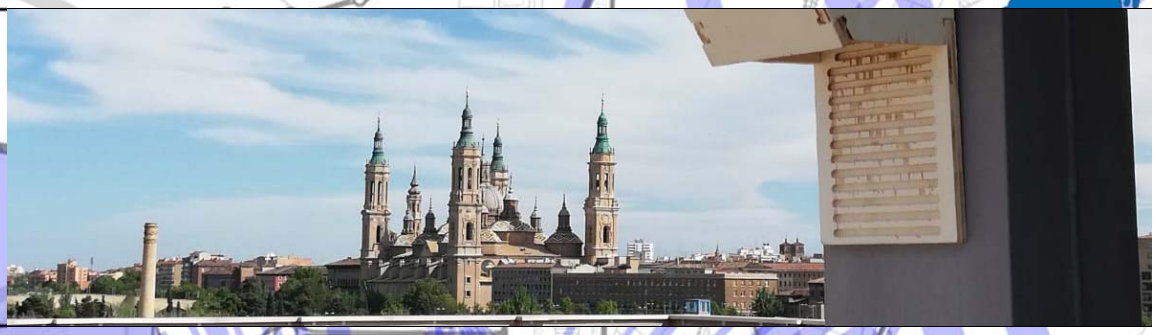
THE THINGS NETWORK ZARAGOZA



Intro TTN, arduino LMiC

Semana de la Ciencia Ciudadana

eTopia 8/6/2019

Enrique F. Torres

grupo de Arquitectura de Computadores de Zaragoza

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Maker Media Ceases Operations

Talleres

Explotación de datos con Node-Red, InfluxDB y Grafana

dockers
dockerscomposer

- Xose Pérez es Core Team de la comunidad TTN Barcelona, también conocido como Tinkerman y creador de ESPurna entre otros muchos proyectos.
- En este taller veremos cómo sacar partido a los datos que nos llegan a TTN integrando la gestión en Node-Red, almacenándolos en InfluxDB y visualizándolos en Grafana.

disclaimer

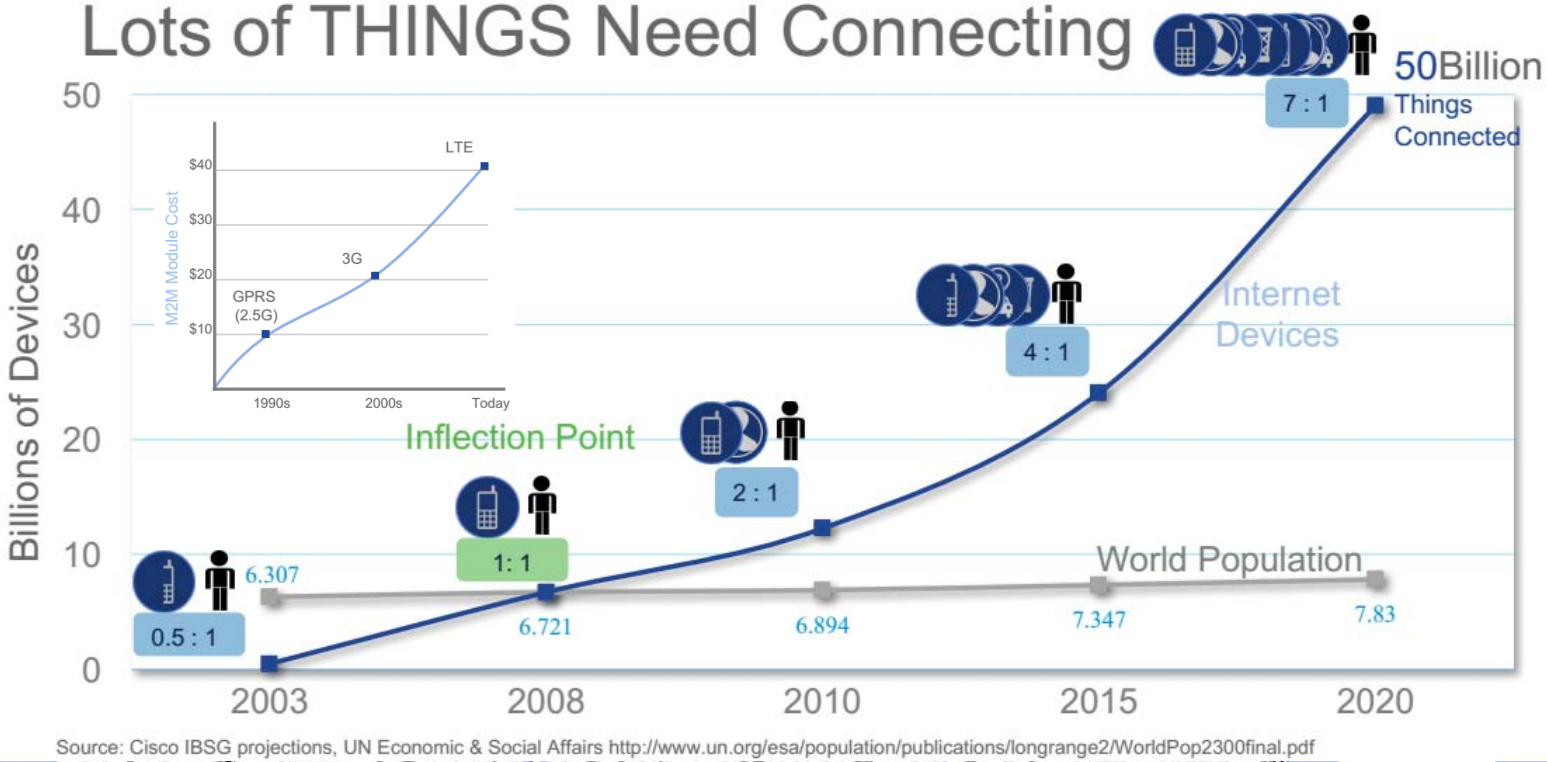
```
#ifndef usuario_de_TTN
  #define dame_5_minutos
#endif
#ifndef comunidad_TTN_zgz
  #define ya_estoy_en_ello
#endif
#ifndef IDE_arduino
  #define vengo_de_oyente  yo | me_ajunto
#endif
#ifdef https://github.com/matthijskooijman/arduino-lmic
  #undef https://github.com/matthijskooijman/arduino-lmic
    backup() //ya_es_hora_de_cambiar
#endif
```


IoT - Internet de los Trastos

- IoT-GSI: ITU-T Y.2060 (06/2012) a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving **interoperable** information and communication technologies

- Why
  - Generate, collect, process and use acquired information to make decisions

Evolución

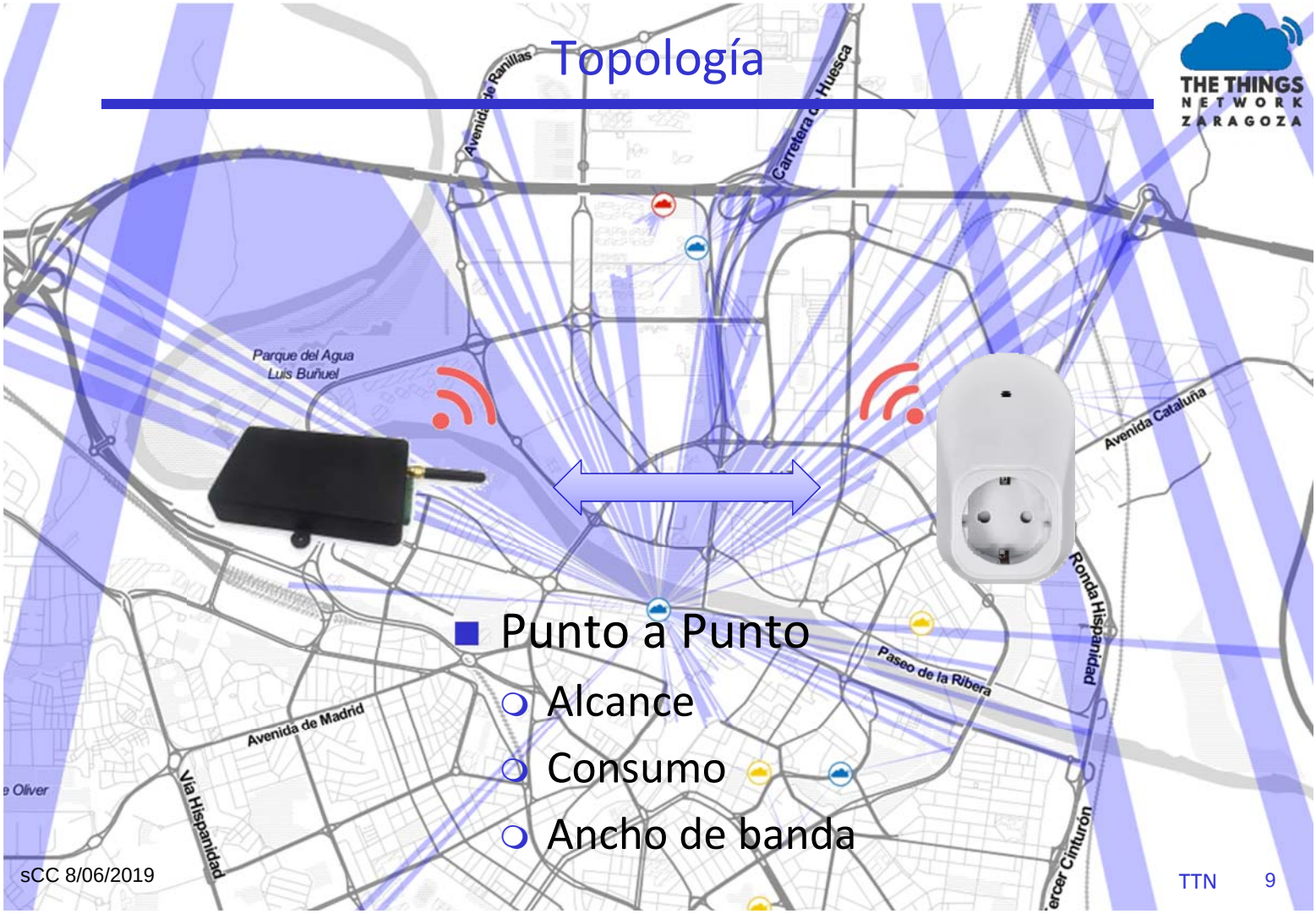


Source: Cisco IBSG projections, UN Economic & Social Affairs http://www.un.org/esa/population/publications/longrange2/WorldPop2300final.pdf

IoT – Internet de las Cosas

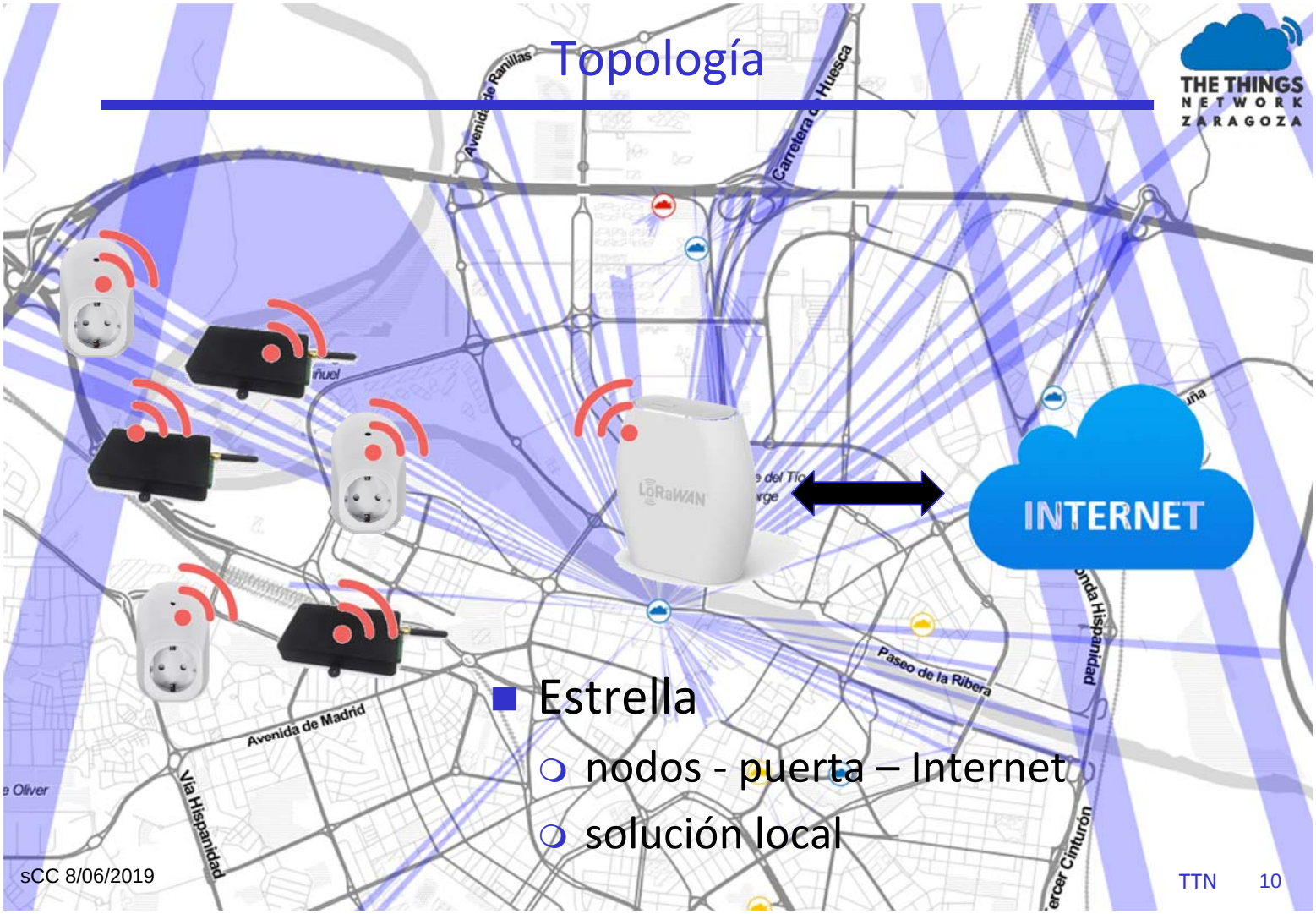
- Cosa -> NODO
- Sistema Empotrado

Topología



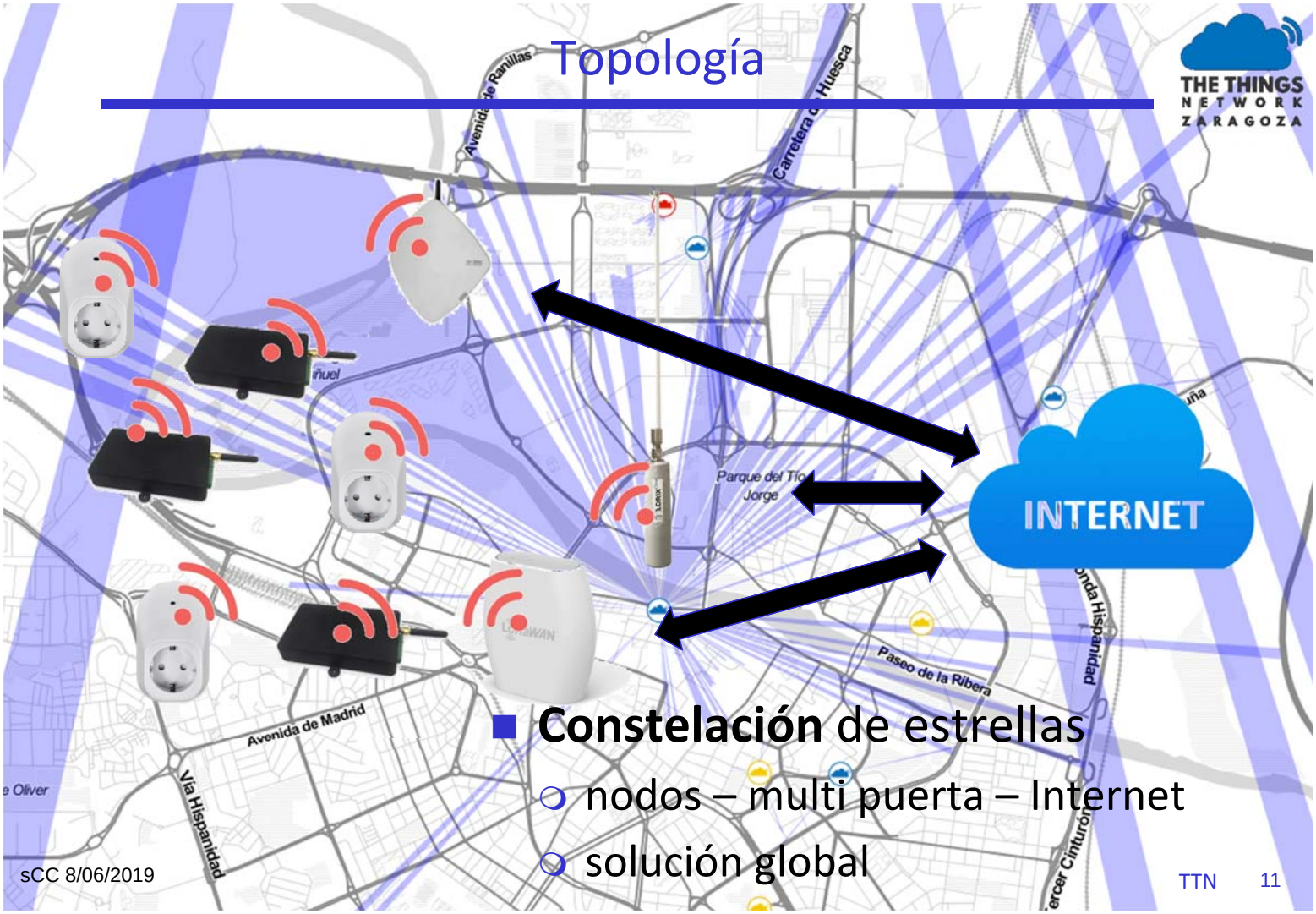
- Punto a Punto
  - Alcance
  - Consumo
  - Ancho de banda

Topología



- Estrella
  - nodos - puerta – Internet
  - solución local

Topología



- **Constelación** de estrellas
  - nodos – multi puerta – Internet
  - solución global

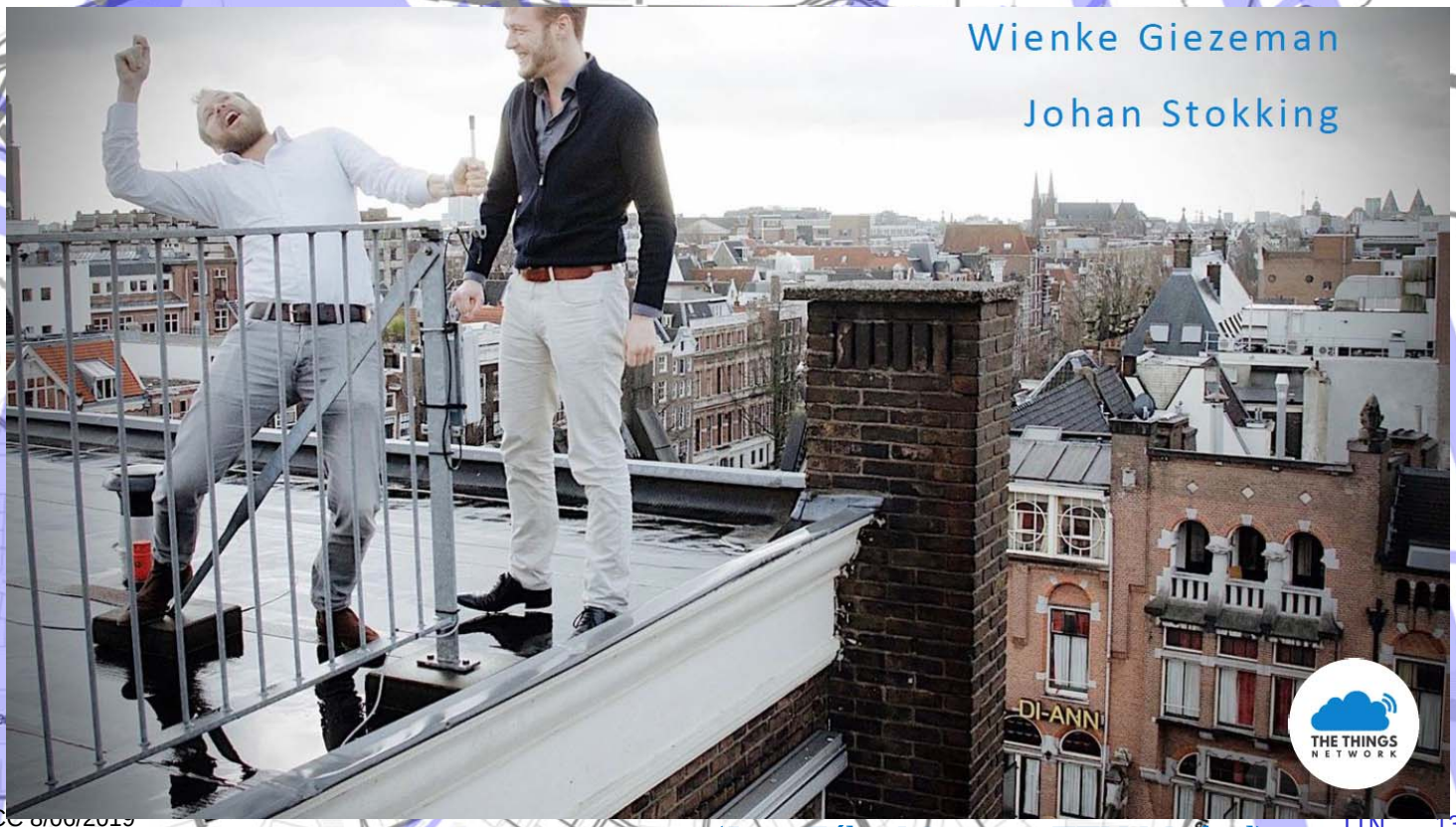
The Things Network

- Misión
  - Crear una red IoT descentralizada y tecnológicamente independiente, en la que los usuarios son a la vez propietarios del sistema
- Principios
  - Tus datos son tus datos
  - Neutralidad de la red
  - Código abierto

(Juan Félix Mateos – TTN Madrid)

Cómo empezó todo

- Julio de 2015, Amsterdam



(Juan Félix Mateos – TTN Madrid)

Cómo empezó todo

Año 2015 KICK STARTER

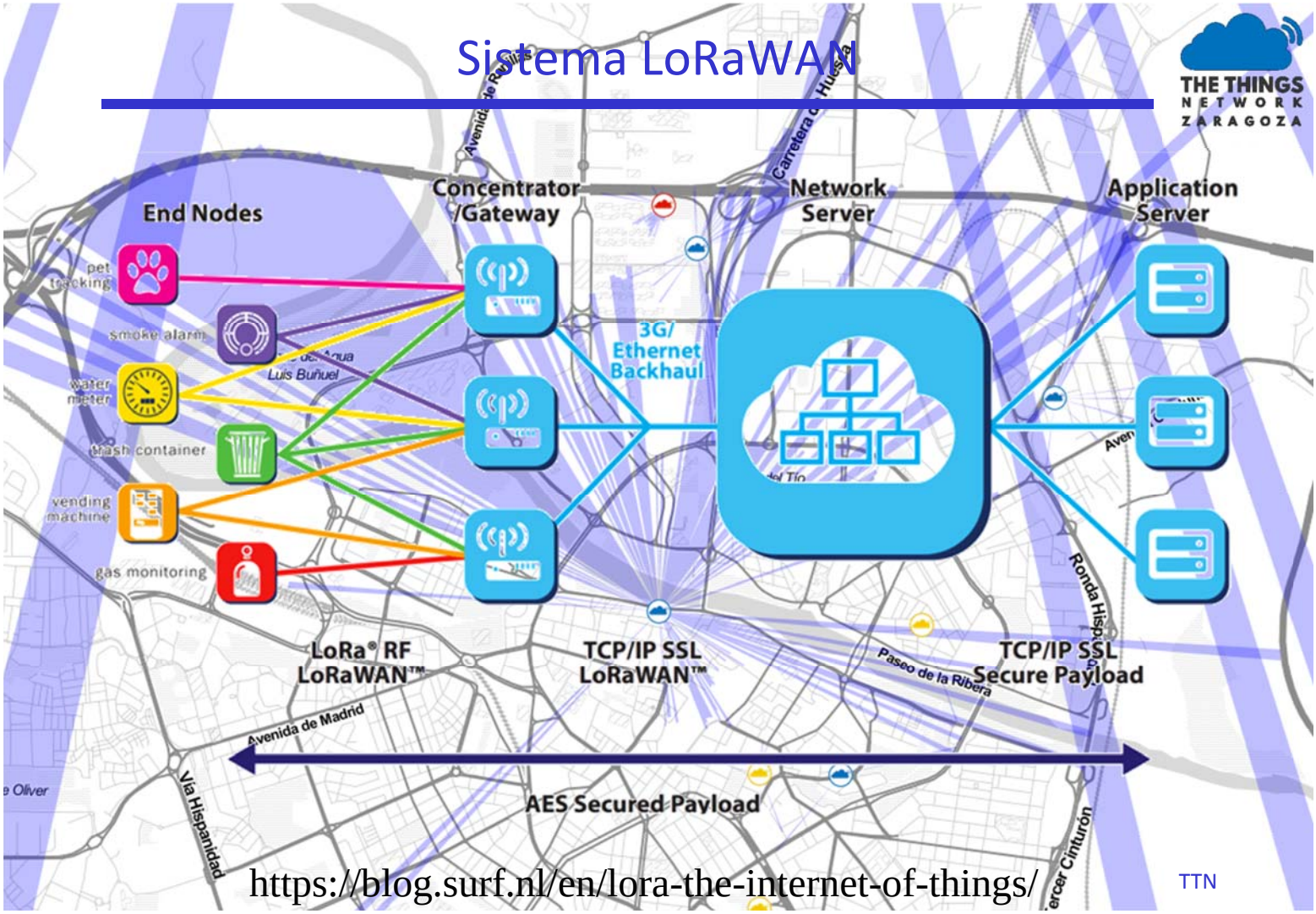
295.331 €

de la meta de 150.000 €

934

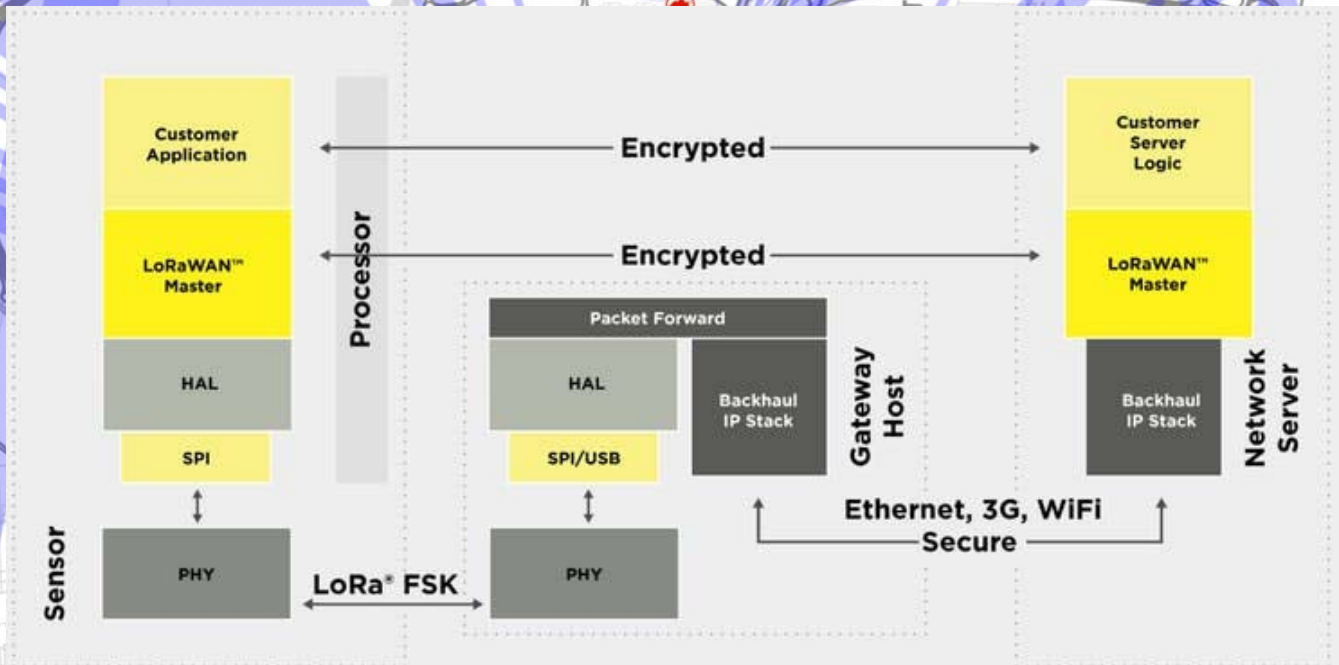
patrocinadores

Sistema LoRaWAN



https://blog.surf.nl/en/lora-the-internet-of-things/

Sistema LoRaWAN



https://shop.marcomweb.it/images/stories/virtuemart/product/Get_Start_with_RAK_LoRa_Develop_Kit3.pdf

LoRa - LoRaWAN

LoRa®

- **Lo**ng **Ra**nge & Low Power
  - 10km, Penetración en edificios (1-2km?)
  - Alta tolerancia interferencias y sensibilidad recepción
- Cycleo de Grenoble, Francia, comprada por Semtech en 2012
- Banda ISM, 868 Mhz en EU
  - Uso libre y gratuito
    - 25 mW
    - Uso 1% del tiempo

LoRaWAN™

- LPWAN (Low Power)
- Nivel MAC y RED basado en cloud
  - routing, direccionamiento, gestión de red
- Protocolo de red y arquitectura
  - bidireccional
  - seguridad
  - control tasa de datos y frecuencias
  - duplicados
  - geolocalización por triangulación

LoRa® Alliance

- 2015
  - interoperabilidad
  - sin lucro, 500 miembros, 100 operadores

TTN

LoRa

- Conexiones punto a punto
- Largo alcance
- Bajo consumo de energía
- Bajo ancho de banda

COMPARISON – main LPWAN technologies

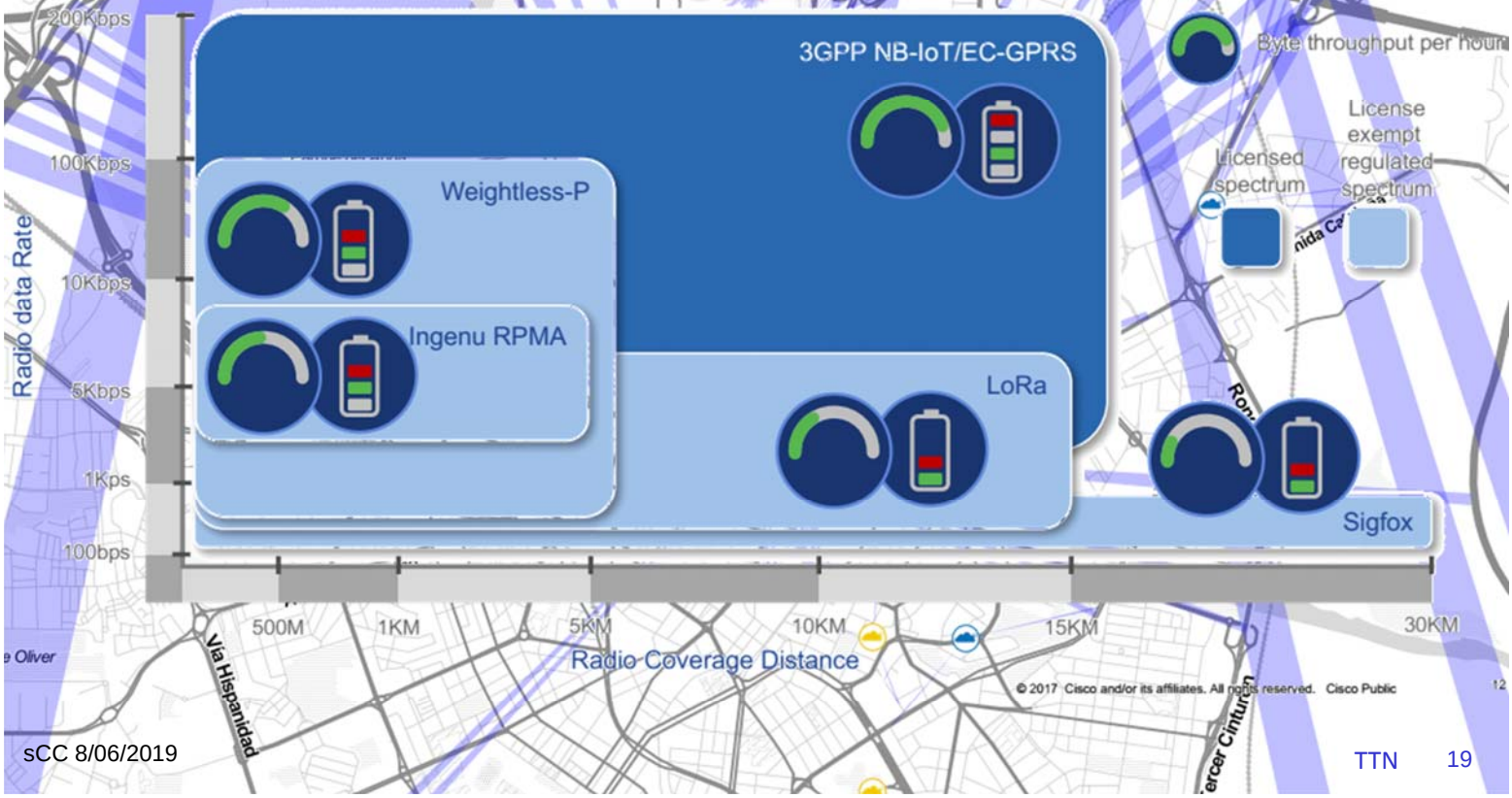
Vertical M2M

Feature	LORAWAN	SIGFOX	LTE Cat 1	LTE M	NB - LTE
Modulation	SS chip	UNB / GFSK / BPSK	OFDMA	OFDMA	OFDMA
Rx Bandwith	500 – 125 KHz	100 Hz	20 MHz	20 – 1.4 MHz	200 KHz
Data Rate	290bps – 50Kbps	100 bit / sec 12 / 8 bytes Max	10 Mbit /sec	200 kbps – 1 Mbps	Average 20K bit / sec
Max. # Msgs/day	Unlimited	UL: 140 msgs / day	Unlimited	Unlimited	Unlimited
Max Outpu Power	20 dBm	20 dBm	23 – 46 dBm	23/30 dBm	20 dBm
Link Budget	154 dB	151 dB	130 dB+	146 dB	150 dB
Battery lifetime – 2000 mAh	105 months	90 months		18 months	
Power Efficiency	Very High	Very High	Low	Medium	Med high
Interference immunity	Very High	Low	Medium	Medium	Low
Coexistence	Yes	No	Yes	Yes	No
Security	Yes	No	Yes Oui	Yes	Yes
Mobility / localization	Yes	Limited mobility, No localization	Mobility	Mobility	Limited mobility, No localization

Source: LoRAWAN Alliance, 2015

www.vertical-m2m.com

LPWA Technologies

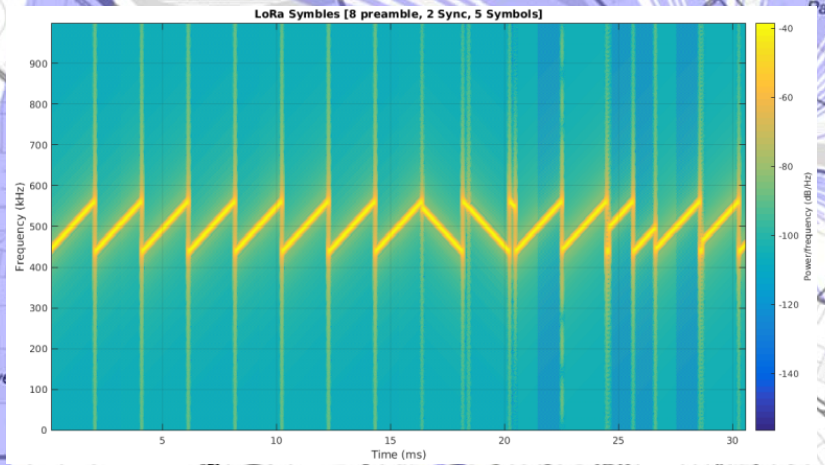


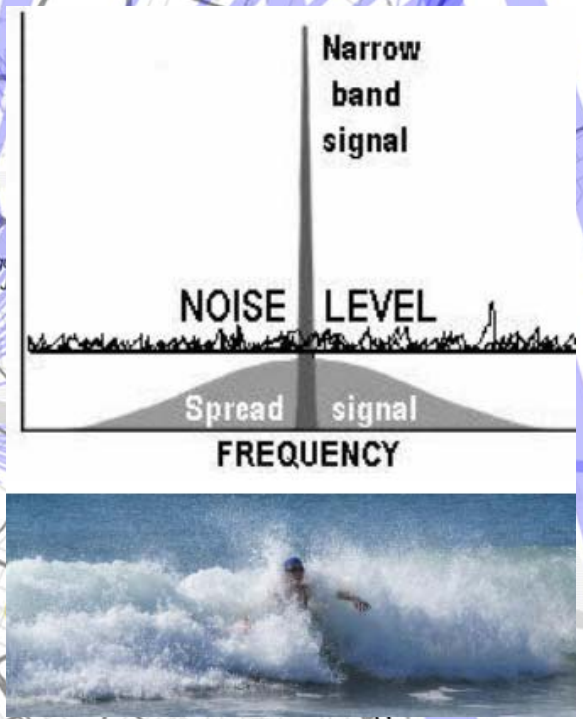
LoRaWAN vs SigFox

- CSS (Chirp Spread Spectrum)

vs

- UNB (Ultra Narrow Band)





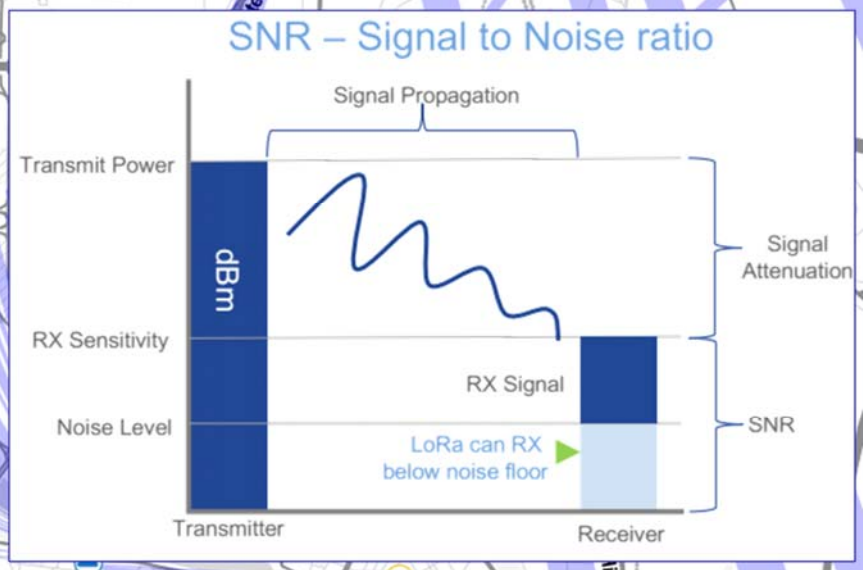
https://blog.surf.nl/en/lora-the-internet-of-things/

RSSI - SNR

RF Signal Primer

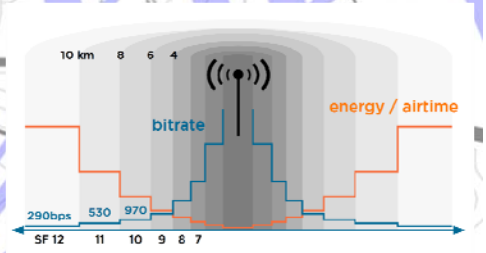
dBm – Decibel-milliwatts

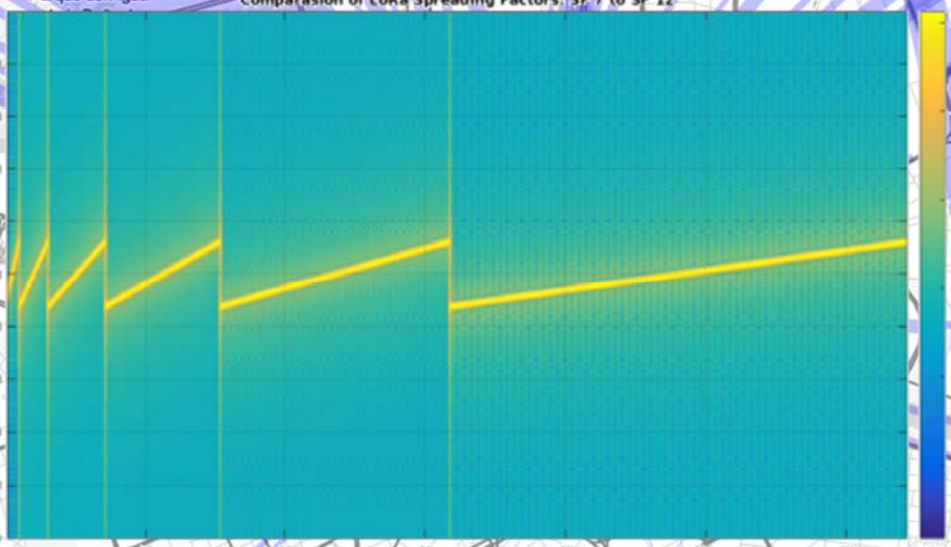
dBm	Power	
0	1 Milliwatt	Bluetooth at 1m
30	1 Watt	GSM Mobile Phone
-127	0.178 FemtoWatt	GPS Satellite Signal



LoRa: Spread Factor

- Velocidad vs Alcance vs Consumo
  - A mayor velocidad, menor alcance
  - A mayor velocidad, menor consumo

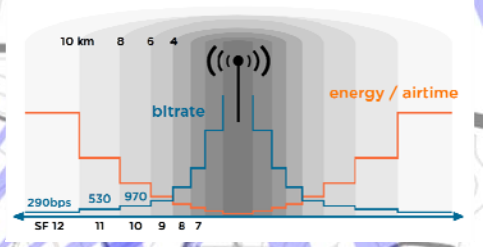




http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.htm

LoRa: Spread Factor

- Velocidad vs Alcance vs Consumo
  - A mayor velocidad, menor alcance
  - A mayor velocidad, menor consumo



LoRa Spreading Factors (125kHz bw)

Each spreading factor is orthogonal on the same transmission channel

Spreading Factor	Chips/symbol	SNR limit	Time-on-air (10 byte packet)	Bitrate
7	128	-7.5	56 ms	5469 bps
8	256	-10	103 ms	3125 bps
9	512	-12.5	205 ms	1758 bps
10	1024	-15	371 ms	977 bps
11	2048	-17.5	741 ms	537 bps
12	4096	-20	1483 ms	293 bps

http://www.techplayon.com/lora-link-budget-sensitivity-calculations-example-explained/

Channels and Duty Cycle

- supervised in Europe by the European Telecommunications Standards Institute (ETSI)
  - In Europe, a LoRaWan device can have up to 8 channels with 3 mandatory (868.1MHz, 868.3MHz and 868.5MHz).

Band	Edge Frequencies		Field Power	Spectrum Access	Band Width
g (Note1,2)	863 MHz	870 MHz	+14 dBm	0.1% or LBT+AFA	7 MHz
g (Note2)	863 MHz	870 MHz	-4.5 dBm / 100 kHz	0.1% or LBT+AFA	7 MHz
g (Note2)	865 MHz	870 MHz	-0.8 dBm / 100 kHz	0.1% or LBT+AFA	5 MHz
	865 MHz	868 MHz	+6.2 dBm / 100 kHz	1% or LBT+AFA	3 MHz
g1	868.0 MHz	868.6 MHz	+14 dBm	1% or LBT+AFA	600 kHz
g2	868.7 MHz	869.2 MHz	+14 dBm	0.1% or LBT+AFA	500 kHz
g3	869.4 MHz	869.65 MHz	+27 dBm	10% or LBT+AFA	250 kHz
g4	869.7 MHz	870 MHz	+14 dBm	1% or LBT+AFA	300 kHz
g4	869.7 MHz	870 MHz	+7 dBm	No requirement	300 kHz

Note1: Modulation bandwidth ≤ 300 kHz is allowed. Preferred channel spacing is ≤ 100 kHz.
Note2: Sub-bands for alarms are excluded (see ERC/REC 70-03 Annex 7).

LoRa: Duty Cycle

- 36 segundos por hora
- Entre 59 y 250bytes por mensaje dependiendo del SF
- Sección 7.2.3 de la norma ETSI EN300.220
  - Sub Bandas
    - g0 (863.0 – 868.0 MHz): 1%
    - g1 (868.0 – 868.6 MHz): 1%
    - g2 (868.7 – 869.2 MHz): 0.1%
    - g3 (869.4 – 869.65 MHz): 10%
    - g4 (869.7 – 870.0 MHz): 1%

frequency plan definitions used in The Things Network
EU863-870
Uplink:
868.1 - SF7BW125 to SF12BW125
868.3 - SF7BW125 to SF12BW125 and SF7BW250
868.5 - SF7BW125 to SF12BW125
867.1 - SF7BW125 to SF12BW125
867.3 - SF7BW125 to SF12BW125
867.5 - SF7BW125 to SF12BW125
867.7 - SF7BW125 to SF12BW125
867.9 - SF7BW125 to SF12BW125
868.8 - FSK
Downlink:
Uplink channels 1-9 (RX1)
869.525 - SF9BW125 (RX2 downlink only)

https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html

LoRa: Duty Cycle

- 36 segundos por hora
- Entre 59 y 250bytes por mensaje dependiendo del SF
- Sección 7.2.3 de la norma ETSI EN300.220
  - Sub Bandas
    - g0 (863.0 – 868.0 MHz): 1%

250

The Things Network: política de acceso justo
- La red es un procomún
  - Todo@s nos beneficiamos de un uso razonable
- Uplink: 30 segundos por día y nodo (aprox. 550 mensajes en SF7)
- Downlink: 10 mensajes por día y nodo

frequency plan definitions used in The Things Network
EU863-870
Uplink:
868.1 - SF7BW125 to SF12BW125
Uplink channels 1-9 (RX1)
869.525 - SF9BW125 (RX2 downlink only)

https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html

LoRaWAN Description

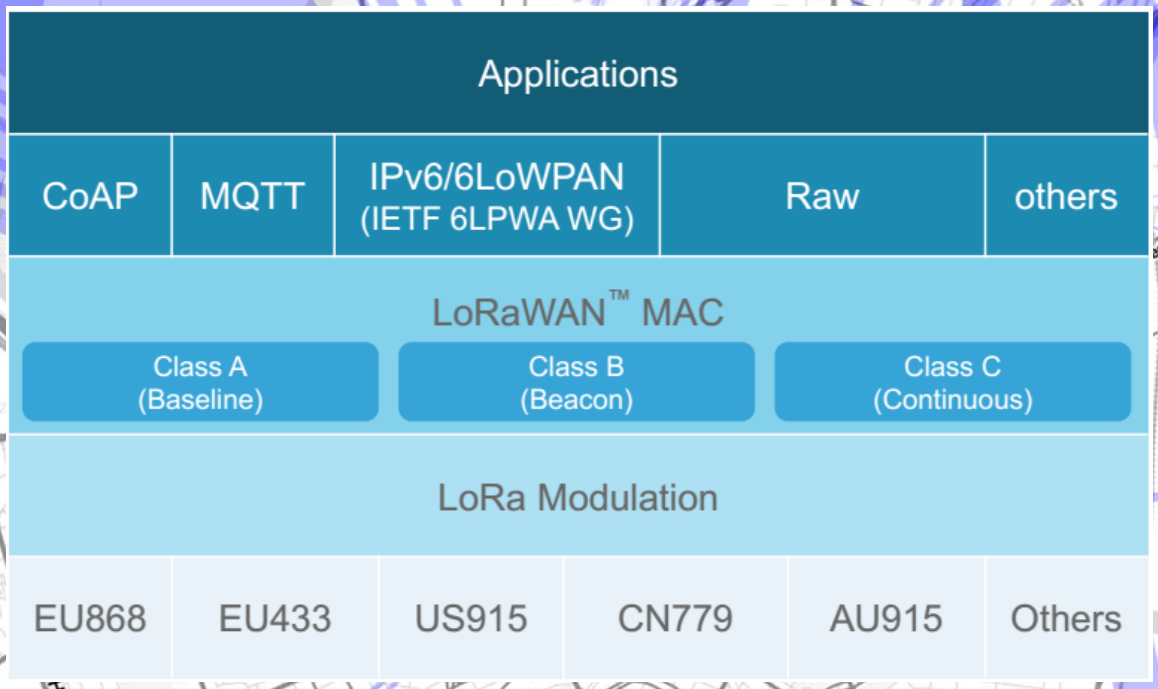
- Star-of-stars topology
  - Gateways (AP) act as transparent bridge relaying messages between end-devices and cloud based network server (NS)
- Sensors use single-hop wireless communication
  - To one or many gateways
- Communication sensors and Gateway is spread
  - Different frequency channels and data rates
- NS manages the data rate and RF output for each sensor
  - Using adaptive data rate (ADR) scheme

LoRaWAN is a Simple Network

- Any device can transmit to any channel at any time
- No synchronization between devices required
  - Easy to implement devices
- Mote changes channel randomly for each transmission
  - Robust to interferers and collisions
- Piggy-backing for gateway to node/mote communication
  - Acknowledgement may be sent in the next data packet
  - Predictable battery-life

MAC (Classes)

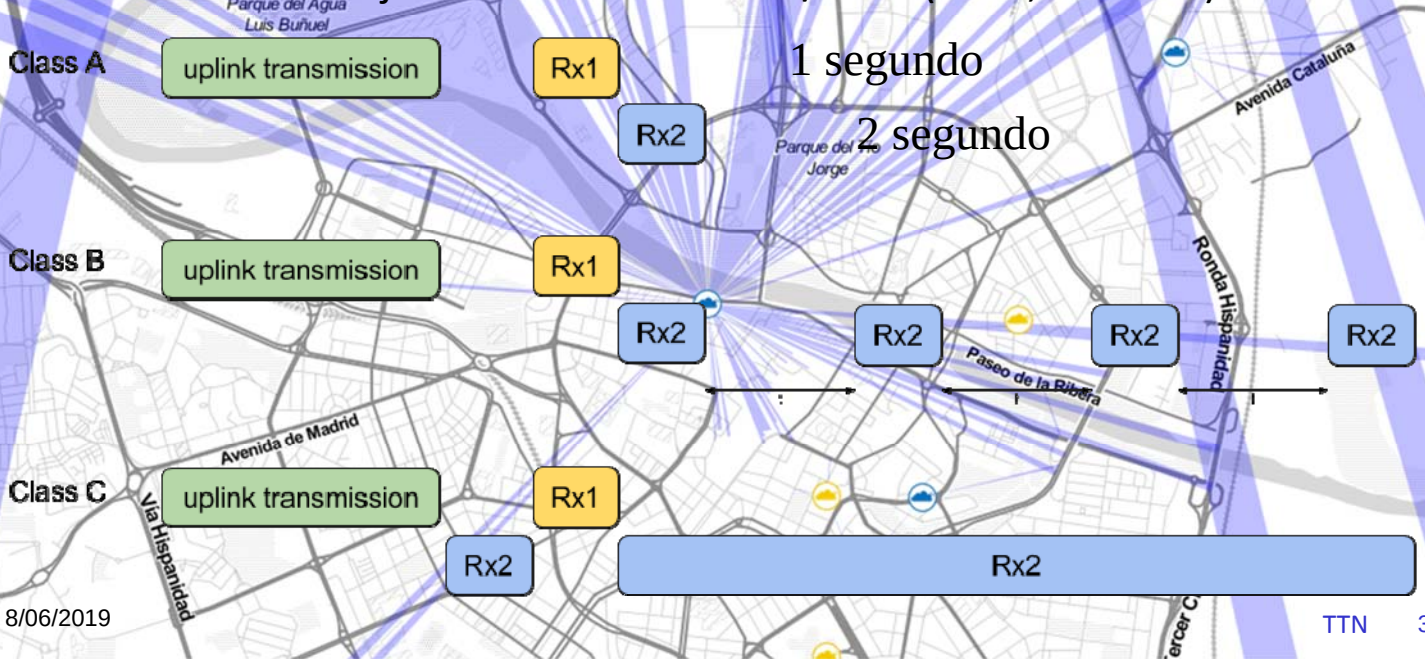
- A: Default (y todos), B: Beacon, C: Continuos



https://2018.apricot.net/assets/files/APIC674/apricot17-lpwa%20giving%20a%20voice%20to%20things-rel4-6_1488325419.pdf

MAC (Classes)

- A: datarate del downlink es:
  - RX1: function of the uplink frequency and a data rate that is a function of the data rate used for the uplink
  - RX2: Está fijado en 869.525 MHz / DR0 (SF12, 125 kHz)



MAC (Classes)

- ADR: Adaptive Data Rate
  - No habilitar hasta que el dispositivo esté estable (sin movimiento)
  - El sistema considera los últimos 20 mensajes para determinar el DR óptimo
- CAD: Channel Activity Detection
  - Se utiliza a nivel de hardware para detectar si hay downlink
- No LBT: no se usa, se opta por duty cycle

Paquete LoRaWAN

- El tamaño máximo aplica al MACPayload
- Message Integrity Code
- ADR y ACK se codifican en FCtrl
- El FRMPayload va encriptado en AES-128

Radio PHY layer:

Preamble	PHDR	PHDR_CRC	PHYPayload	CRC*

Figure 5: Radio PHY structure (CRC* is only available on uplink messages)

PHYPayload:

MHDR	MACPayload	MIC

or

MHDR	Join-Request or Rejoin-Request	MIC

or

MHDR	Join-Accept[2]

Figure 6: PHY payload structure

MACPayload:

FHDR	FPort	FRMPayload

Figure 7: MAC payload structure

FHDR:

DevAddr	FCtrl	FCnt	FOpts

Figure 8: Frame header structure

LoRaWan Spreading Factor

- Velocidad relativa de transmisión y tamaño tramas de datos

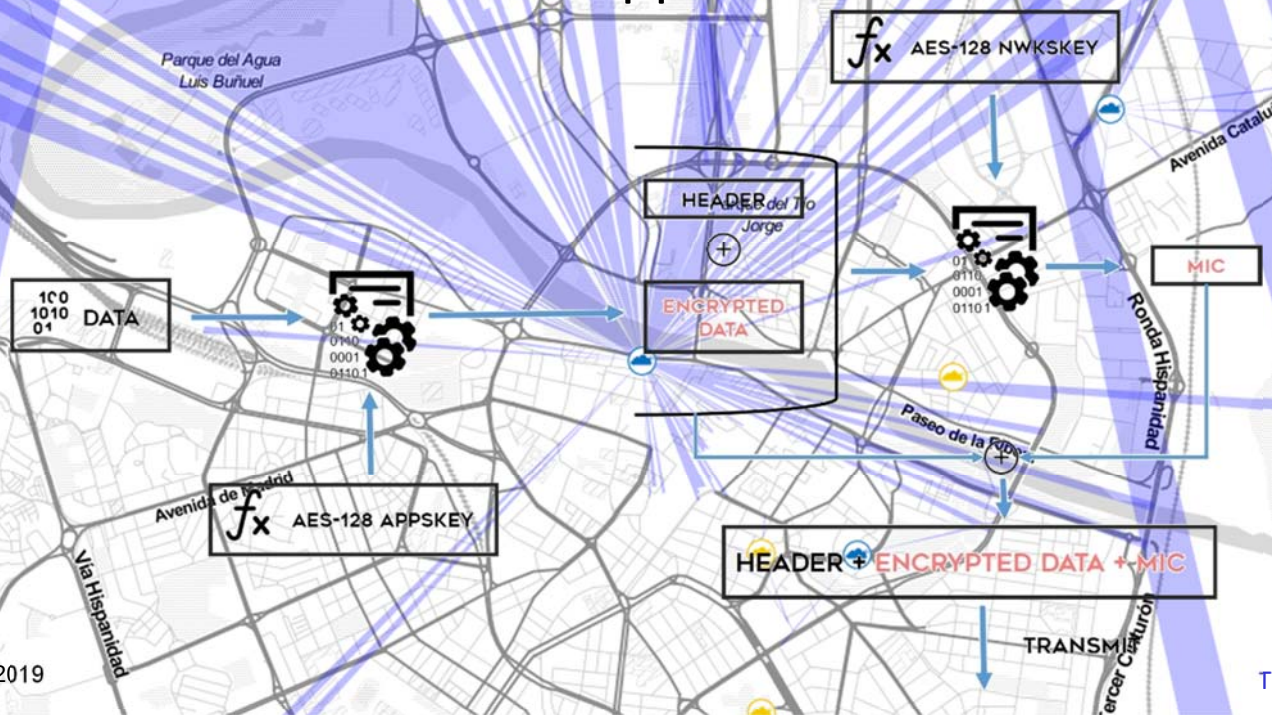
DataRate	Configuration	Indicative physical bit rate [bit/s]	M	N
0	LoRa: SF12 / 125 kHz	250	59	51
1	LoRa: SF11 / 125 kHz	440	59	51
2	LoRa: SF10 / 125 kHz	980	59	51
3	LoRa: SF9 / 125 kHz	1760	123	115
4	LoRa: SF8 / 125 kHz	3125	250	242
5	LoRa: SF7 / 125 kHz	5470	250	242
6	LoRa: SF7 / 250 kHz	11000	250	242
7	FSK: 50 kbps	50000	250	242
8..15	RFU		Not defined	

Table 8 : EU863-870 maximum payload size (not repeater compatible)

```
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7), BAND_CENTI); // g-band
```


Data Encryption

- The LoRaWan specifications uses 2 session keys, unique to each LoRa device, to secure the data at both network and application level



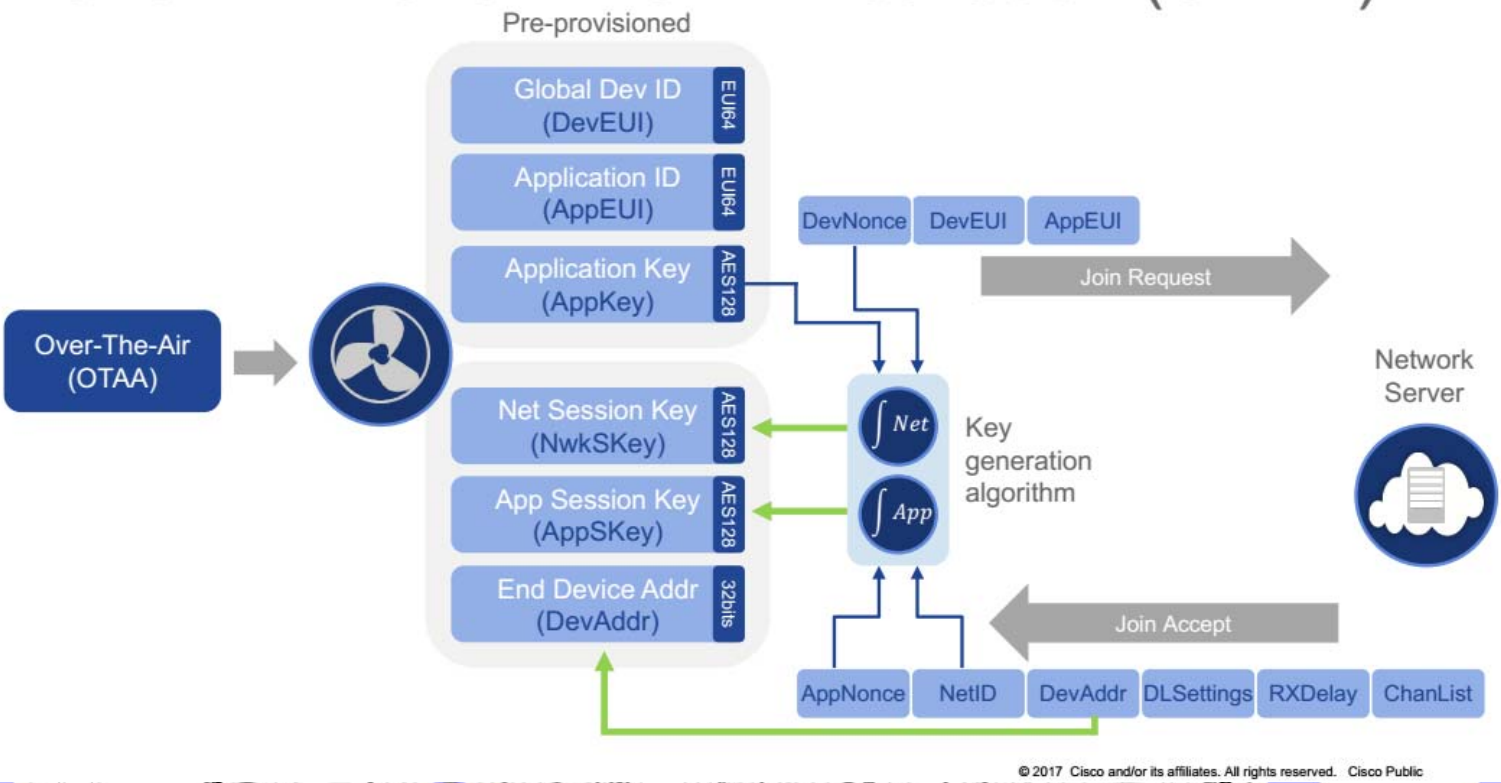
2 keys

- ABP: “Activation-by-personalisation”
  - saved in the LoRaWan device memory
    - **AppSKey** (Application Session Key)
    - **NwkSKey** (Network Session Key)
    - ***(+ DevAddr)***
- OTAA “Over The Air Activation”
  - In this case, the LoRaWan device will hold:
    - An Application Key (AES-128 **AppKey**) assigned by the application owner for each device (unique)
    - A device UUID (EUI64 **DevEUI**) assigned by the LoRa chip manufacturer (unique)
    - An Application Identifier (EUI64 **AppEUI**) assigned by the network provider to the application (non-unique)

OTAA

- When the application starts,
  - the device will send a specific message to the network with the **AppEUI**, the **DevEUI** and a MIC calculated with the **AppKey**.
  - the network that also recognizes the AppKey will get the AppEUI and DevEUI and will verify that the device can join the network or not (the DevEUI should exist for the AppEUI in the network provider settings).
  - finally, when the device can join the network, a message is sent back to it with data to derive the **AppSKey**, the **NwkSKey** and a Device Address (**DevAddr**) used by the device to communicate.

Keys!!!



Pa' saber más

- LoRa alliance:
  - https://www.lora-alliance.org/
- LoRaWAN® Specification v1.0.3
  - https://lora-alliance.org/resource-hub/lorawanr-specification-v103
- LoRaWAN® Specification v1.1
  - https://lora-alliance.org/resource-hub/lorawanr-specification-v11
    - This describes the LoRaWAN® network protocol which is optimized for battery-powered end-devices. LoRaWAN® Specification v1.1 adds support for handover roaming, Class B and security enhancements

Índice

- TTN – LoRa – LoRaWAN
- Arduino LMiC
- Taller

Overview of LoRaWAN libraries

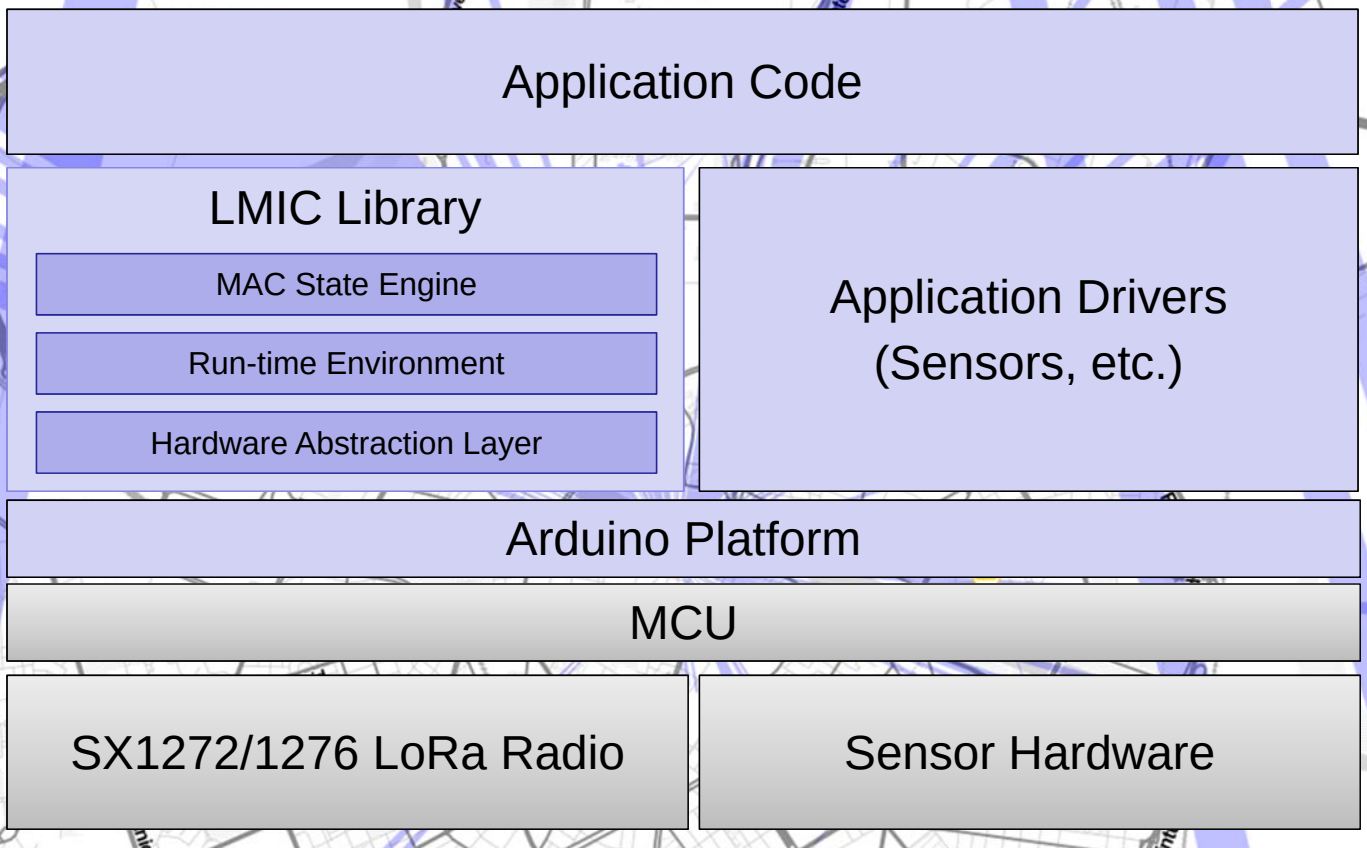
- Semtech's LoRaMac-node reference implementation.
  - > 32KB no Arduino frienly
- IBM's original LMiC (Zurich)
  - LMIC-Arduino Version: 1.5.0+arduino-2
    - https://github.com/matthijskooijman/arduino-lmic
    - hasn't been updated for over a year and the last real code change is even older (IBM 1.5 may 2015)
  - MCCI LoRaWAN LMIC library Version: 2.3.2 by Terry Moore
    - fixing old issues and getting an official certification for the code.
    - IBM 1.6 (Jul 2016)
    - https://github.com/mcci-catena/arduino-lmic

https://www.thethingsnetwork.org/forum/t/overview-of-lorawan-libraries-lmic-loramac-node-and-their-variations/24692

Arduino IBM LoRaWAN C-library (LMIC)

- portable implementation of the LoRaWAN™ 1.0.2 end-device specification in C programming language. ("LMIC" stands for "LoRaWAN MAC in C").
  - supports EU-868, US-915, AU-921, AS-923 and IN-866
  - class A and class B devices.
  - The library takes care of:
    - all logical MAC states
    - timing constraints
    - drives the SEMTECH SX1272 or SX1276 radio.
  - event-based programming model
  - hardware abstraction layer (HAL)

Application device components



https://github.com/mcci-catena/arduino-lmic

LoRaWAN Versions and Features Supported

- The LMIC library supports the LoRaWAN V1.0.2.
  - It is also believed to support V1.0.3. It does not V1.1.
- Class B support with V1.0.2 in the US and V1.0.3 networks is not tested.
  - Devices (and the LMIC library) start out as Class A devices, and switch to Class B based on requests from the application layer of the device. (pinging, tracking)
- LoRaWAN 1.0.3 Class B multicast downlinks are not supported.
- Class C operation is not supported.

Library

- API functions
- run-time functions
- callback functions
- global LMIC data structure
- HAL

```
#include "lmic.h"

//versiones Semantic Versioning 2.0.0
#define ARDUINO_LMIC_VERSION
        ARDUINO_LMIC_VERSION_CALC(2, 2, 2, 0)
//original IBM LMiC
#define LMIC_VERSION_MAJOR 1
#define LMIC_VERSION_MINOR 6
```

Programming Model

- JOBs: run-time environment (`os_` …)
  - timer **queues** and job management
  - simple event-based programming model
    - all protocol events are dispatched to the application's `onEvent()` **callback** function
    - free the application of details like timings or interrupts
  - **jobs** which are executed on the main thread by the run-time scheduler function `os_runloop()`
    - per job control `struct osjob_t`
    - Jobs must not be long-running
      - They should only update state and schedule actions, which will trigger new job or event callbacks

Main event loop

- initialize run-time by `os_init()` or `os_init_ex()`
- call the job scheduler function `os_runloop_once()`
- an initial job needs to be set up

```
osjob_t initjob;
void setup () {
    os_init(); // initialize run-time env
    // setup initial job
    os_setCallback(&initjob, initfunc);
}
void loop () {
    // execute scheduled jobs and events
    os_runloop_once();
} // os_runloop ()
```

p.e. initfunc()

- Initializes the MAC and starts joining the network
- The `initfunc()` function will return immediately,
  - the `onEvent()` callback function will be invoked by the scheduler later on for the events `EV_JOINING`, `EV_JOINED` or `EV_JOIN_FAILED`

```
// initial job
static void initfunc (osjob_t* j) {
    LMIC_reset(); // reset MAC state
    LMIC_startJoining(); // start joining
    // init done - onEvent() callback will be invoked...
}
```


Run-time Functions

- job related

`void os_init ()` Initialize the operating system by calling os_init_ex(NULL).

`void os_init_ex (const void * pHalData)` on multiple platforms, reference to a C++ struct lmic_pinmap object.

```
void os_runloop ()
void os_runloop_once ()

void os_setCallback (osjob_t* job, osjobcb_t cb)
void os_setTimedCallback (osjob_t* job, ostime_t time,
osjobcb_t cb)
void os_clearCallback (osjob_t* job)
```

OS Time

- LMIC uses ticks values (of the type ostime_t)
  - defaults to 32768 ticks per second,
    - but may be configured at compile time to any value between 10000 ticks per second and 64516 ticks per second
    - Note: signed integer value; care must be taken when computing differences to avoid being fooled by overflow. OS time starts at zero, and increments uniformly to INT32_MAX; it then wraps to INT32_MIN and increments uniformly up to zero, and repeats. Rather than comparing two ostime_t values, recommended practice is to subtract them and see if the result is positive or negative.

```
typedef int32_t ostime_t;
```


Run-time Functions

- Time related

```
ostime_t os_getTime () //Query absolute system time (in ticks)
ostime_t us2osticks(s4_t us)
ostime_t us2osticksCeil(s4_t us)
ostime_t us2osticksRound(s4_t us)
ostime_t ms2osticks(s4_t ms)
ostime_t ms2osticksCeil(s4_t ms)
ostime_t ms2osticksRound(s4_t ms)
ostime_t us2osticks(s4_t sec)
S4_t osticks2ms(ostime_t os)
S4_t osticks2us(ostime_t os)
```

Application callbacks

- will be called by the state engine to
  - query application-specific information and
  - to deliver state events to the application

```
void os_getDevEui (u1_t* buf) //copy in buf
device EUI (8 bytes in little-endian format (LSBF))
void os_getDevKey (u1_t* buf) //device-specific
application key is a 128-bit AES key (16 bytes in length).
void os_getArtEui (u1_t* buf) //application EUI
(8 bytes LSBF)
void onEvent (ev_t ev)
```


Application callbacks

- The following events will be reported:

```
EV_JOINING
EV_JOINED
EV_JOIN_FAILED
EV_REJOIN_FAILED
…
EV_TXCOMPLETE //data sent (via LMIC_setTxData())
…
EV_RESET
```

API Functions

- LMIC library offers a set of API functions to
  - control the MAC state and
  - to trigger protocol actions

```
void LMIC_reset () //Reset the MAC state
(Session and pending data)

bit_t LMIC_startJoining ()
void LMIC_tryRejoin ()

void LMIC_setSession (u4_t netid,
devaddr_t devaddr, u1_t* nwkKey, u1_t*
artKey)
```


API Functions

- LMIC library offers a set of API functions to
  - control the MAC state and
  - to trigger protocol actions

```
bit_t LMIC_setupBand (u1_t bandidx, s1_t
txpow, u2_t txcap)

bit_t LMIC_setupChannel (u1_t channel,
u4_t freq, u2_t drmap, s1_t band)

void LMIC_disableChannel (u1_t channel)
//Disable specified channel.
…
```

API - LMIC Struct

- protocol state can be accessed via a global LMIC structure
  - most fields are read-only and should not be modified

```
struct lmic_t {
    u1_t          frame[MAX_LEN_FRAME];
    u1_t          dataLen;      // 0 no data or zero length data, >0 byte count of data
    u1_t          dataBeg;      // 0 or start of data (dataBeg-1 is port)

    u1_t          txCnt;
    u1_t          txrxFlags;    // transaction flags (TX-RX combo)

    u1_t          pendTxPort;
    u1_t          pendTxConf; // confirmed data
    u1_t          pendTxLen;
    u1_t          pendTxData[MAX_LEN_PAYLOAD];

    u1_t          bcnChnl;
    u1_t          bcnRxsyms;
    ostime_t      bcnRxtime;
    bcninfo_t     bcninfo;    // Last received beacon info
    …
```


API Functions

- sending data

```
void LMIC_setTxData ()

int LMIC_setTxData2 (u1_t port, xref2u1_t data, u1_t dlen, u1_t confirmed)
```

Prepare upstream data transmission at the next possible time. Convenience function for `LMIC_setTxData()`. If data is NULL, the data in `LMIC.pendTxData[]` will be used.

Receiving Downlink Data

- check for downlink data in EV_TXCOMPLETE or EV_RXCOMPLETE

```
// Any data to be received?
    if (LMIC.dataLen != 0) {
        // Data was received. Extract port number if any.
        u1_t bPort = 0;
        if (LMIC.txrxFlags & TXRX_PORT)
            bPort = LMIC.frame[LMIC.dataBeg - 1];
        // Call user-supplied function with port #, pMsg, nMsg
        receiveMessage(
            bPort, LMIC.frame + LMIC.dataBeg, LMIC.dataLen);
    }
```


Hardware Abstraction Layer

- portable code and a small platform-specific part
- HAL Interface
  - Four digital I/O lines are needed in output mode to drive the radio's antenna switch (RX and TX), the SPI chip select (NSS), and the reset line (RST).
  - Three digital I/O lines are needed in input mode to sense the radio's transmitter and receiver states (DIO0, DIO1 and DIO2).
  - A SPI unit is needed to read and write the radio's registers.
  - A timer unit is needed to precisely record events and to schedule new protocol actions.
  - An interrupt controller is needed to forward interrupts generated by the digital input lines

function interface to access hardware

```
//called from os_init()/os_init_ex()
void hal_init_ex (const void *pHalData);
//pointer to a lmic_pinmap structure

void hal_waitUntil (u4_t time)
void hal_disableIRQs ()
void hal_enableIRQs ()

void hal_sleep () //low power?
//Arduino reference implementation, this is a no-op.
//void hal_sleep () {    // Not implemented}
```


struct lmic_pinmap

```
#if defined(ARDUINO_AVR_FEATHER32U4)
// Pin mapping for Adafruit Feather 32u4 LoRa, etc.
// Just like Feather M0 LoRa, but uses SPI at 1MHz; and that's only
// because MCCI doesn't have a test board; probably higher frequencies
// will work.
const lmic_pinmap lmic_pins = {
    .nss = 8,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 4,
    .dio = {7, 6, LMIC_UNUSED_PIN},
    .rxtx_rx_active = 0,
    .rssi_cal = 8,// LBT cal for the Adafruit Feather 32U4 LoRa, in dB
    .spi_freq = 1000000,
};
```

example ttn-abp

```
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
...
```

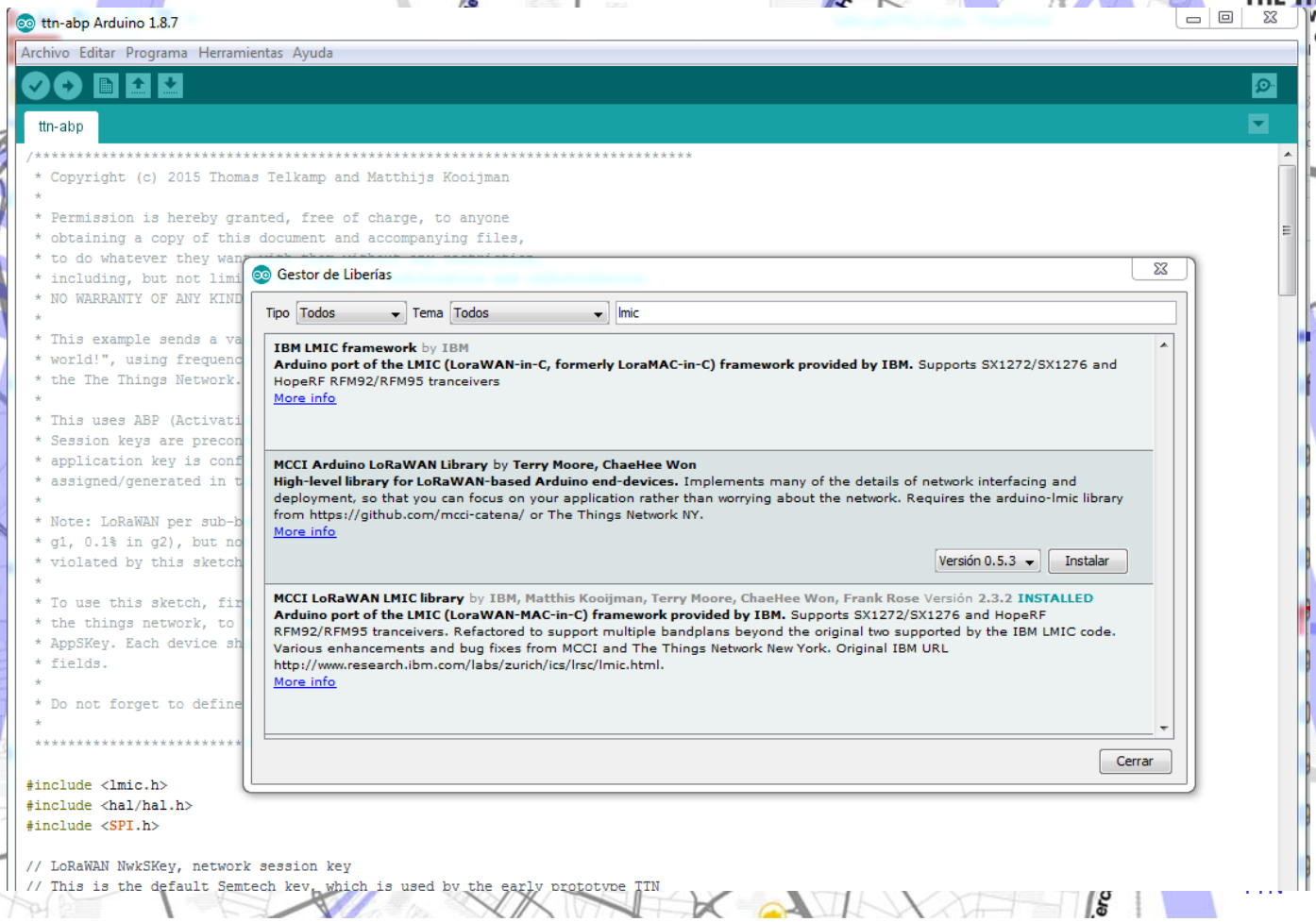

example ttn-abp

```
void setup() {
    os_init(); // LMIC init
    LMIC_reset(); // Reset the MAC state. Session and pending data transfers will
    LMIC_setSession (0x13, DEVADDR, NWKSKEY, APPSKEY);
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),  B
    ...
    LMIC_setLinkCheckMode(0); // Disable link check validation
    LMIC.dn2Dr = DR_SF9; // TTN uses SF9 for its RX2 window.
    LMIC_setDrTxpow(DR_SF7,14); // Set data rate and transmit power f
    do_send(&sendjob); // Start job
}
void loop() {
    os_runloop_once();
    ...
}
```

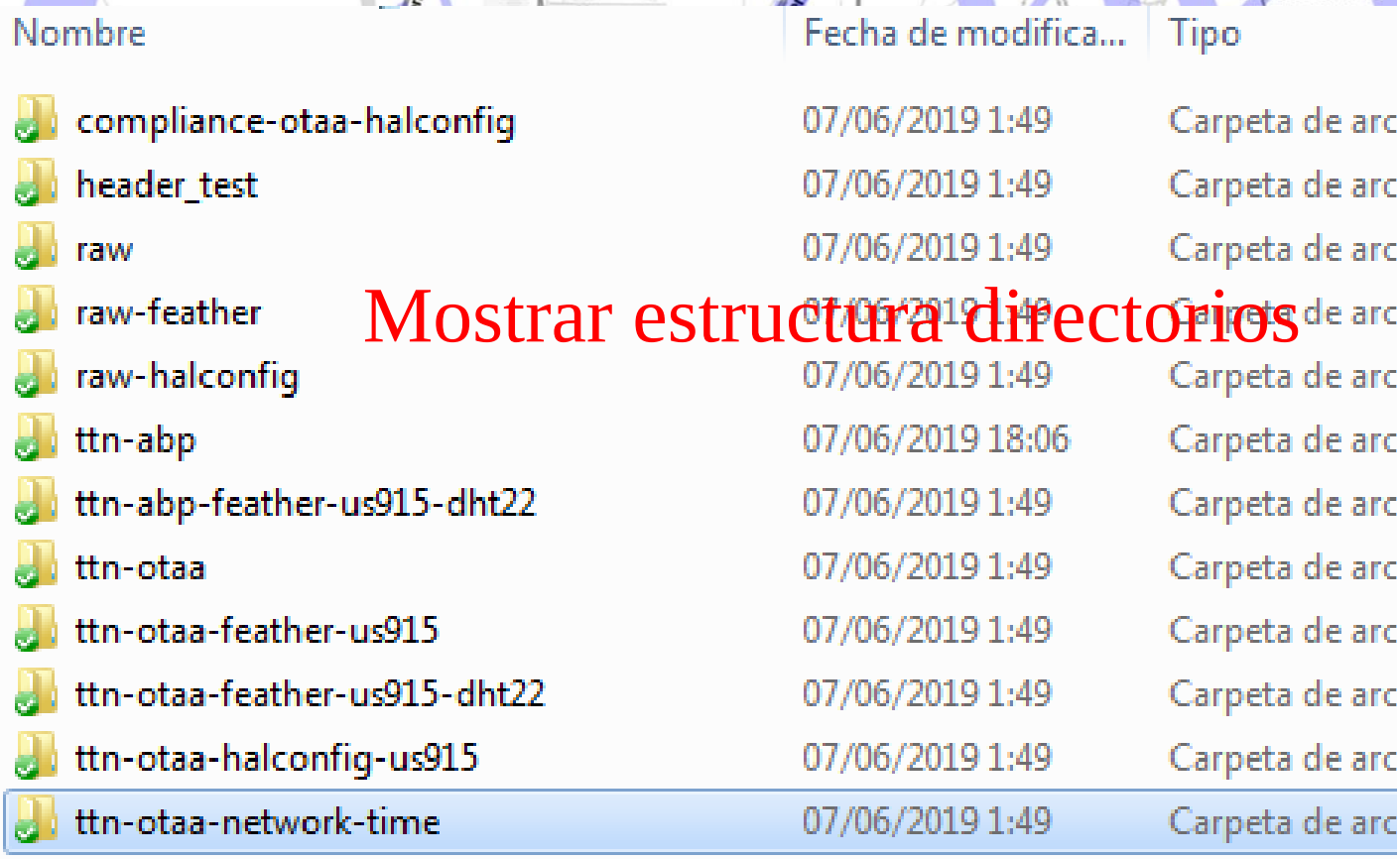
Índice

- TTN – LoRa – LoRaWAN
- Arduino LMiC
- Taller

LMIC (MCCI)



examples



Mostrar estructura directorios

lmic_project_config.h

- ttn-abp.ino

```
* Do not forget to define the radio type correctly in
* arduino-lmic/project_config/lmic_project_config.h or from your BOARDS.txt
// project-specific definitions
//#define CFG_eu868 1
#define CFG_us915 1
//#define CFG_au921 1
//#define CFG_as923 1
// #define LMIC_COUNTRY_CODE LMIC_COUNTRY_CODE_JP /
as923-JP */
//#define CFG_in866 1
#define CFG_sx1276_radio 1
//#define LMIC_USE_INTERRUPTS
```

ABP only?

- Si solo usamos ABP
  - Eliminar código relacionado con JOIN

```
// DISABLE_JOIN is set in arduino-
lmic/project_config/lmic_project_config.h,

// DISABLE_JOIN is set in arduino-lmic/lmic/config.h,
```


ttn-otaa.ino

```
u4_t netid = 0;
devaddr_t devaddr = 0;
u1_t nwkKey[16];
u1_t artKey[16];

LMIC_getSessionKeys(&netid, &devaddr, nwkKey, artKey);

Serial.print("netid: "); Serial.println(netid, DEC);
Serial.print("devaddr: "); Serial.println(devaddr, HEX);
Serial.print("artKey: ");
for (int i=0; i<sizeof(artKey); ++i) {
      Serial.print(artKey[i], HEX);  }
Serial.println("");
Serial.print("nwkKey: ");
for (int i=0; i<sizeof(nwkKey); ++i) {
      Serial.print(nwkKey[i], HEX);  }
Serial.println("");
```

- EV_JOINED:

Keys.h

- Ofuscar claves del *.ino para distribuirlo

```
//en *.ino
#include "keys.h“
//en keys.h en misma carpeta (u otra…)
static const u1_t PROGMEM APPEUI[8] = { 0x73…};

// This should also be in little endian format,
static const u1_t PROGMEM DEVEUI[8] = { 0xD5,…}
…
// The key shown here is the semtech default ke
static const u1_t PROGMEM APPKEY[16] = { 0x3E,..
```


Hal.h

- Nuevas formas interesantes… de momento…

```
extern const lmic_pinmap lmic_pins;
```

lmic_pinmap

```
// https://www.thethingsnetwork.org/forum/t/big-esp32-sx
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = LMIC_UNUSED_PIN,
    //If DIO2 is not connected use:
    //.dio = {/*dio0*/ 26, /*dio1*/ 33, /*dio2*/ LMIC_UNU
    //If DIO2 is connected use:
    .dio = {/*dio0*/ 26, /*dio1*/ 33, /*dio2*/ 32}
};
#define OLED_SCL 22
#define OLED_SDA 21
#define OLED_ADDR 0x3C //Default 0x3C for 0.9",0x78 for 1
```


const unsigned TX_INTERVAL = 60;

- Modificar velocidad de envios

```
case EV_TXCOMPLETE:
  Serial.println(F("EV_TXCOMPLETE (includes waiting for R
  if (LMIC.txrxFlags & TXRX_ACK) Serial.println(F("Receiv
  if (LMIC.dataLen) {
…
  // Schedule next transmission
  os_setTimedCallback(&sendjob,
os_getTime() + sec2osticks(TX_INTERVAL), do_send)
  break;
```

os_setCallback (LMiC controls duty cycle)

1ch gateway

- Test de Gateway de un solo canal

```
#ifdef MODO_TEST
    // ---------------------
    // Desactivamos todos los canales menos el canal 0
    //--> SOLO PARA TESTING!!!
    // Define the single channel and data rate (SF) to use
    // channel = 0   /     DR_SF7
    forceTxSingleChannelDr(0, DR_SF7);
    // ---------------------
 #endif //MODO_TEST
```



```
// https://www.thethingsnetwork.org/forum/t/can-lmic-1-6-be-set-up-to-use-a-single-channel-and-sf/5207/11
// Disables all channels, except for the one defined above, and sets the
// data rate (SF). This only affects uplinks; for downlinks the default
// channels or the configuration from the OTAA Join Accept are used.
//
// Not LoRaWAN compliant; FOR TESTING ONLY!
//
void forceTxSingleChannelDr(int channel, int dr) {
  for (int i = 0; i < 9; i++) { // For EU; for US use i<71
    if (i != channel) {
      LMIC_disableChannel(i);
    }
  }
  // Set data rate (SF) and transmit power for uplink
  LMIC_setDrTxpow(dr, 14);
}
```

LPP cayenne

- Usar el formato Low Power cayenne

```
#include <CayenneLPP.h>
CayenneLPP lpp(51);

void build_data(){
  lpp.reset();
  lpp.addTemperature(1, temperatura1);
…
void do_send(osjob_t* j){
  build_data();
  LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(
```


THE THINGS NETWORK ZARAGOZA

https://www.thethingsnetwork.org/community/zaragoza

https://t.me/TTNZGZ

@TtnZgz