

CONSTRUYENDO UNA RED GLOBAL DE INTERNET DE LAS COSAS ENTRE TOD@s

Faciendo IoT con LoRaWAN. Introducción

juanfelixmateos@gmail.com



THE THINGS NETWORK

UNA RED IOT ENTRE Y PARA TOD@S

The Things Network: Misión

Crear una red IoT descentralizada y tecnológicamente independiente, en la que los usuarios son a la vez los propietarios y los operadores del sistema



The Things Network: Principios

Tus datos son tus datos

Neutralidad de la red

Código abierto



Cómo empezó todo

Julio de 2015, Amsterdam

Wienke Giezeman
Johan Stokking



Juan Félix Mateos





Long Range Wide Area Network

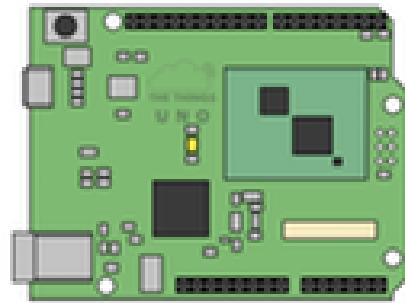
KICKSTARTER



TheThings
GATEWAY



TheThings UNO



Año 2015
**KICK
STARTER**

295.331 €

de la meta de 150.000 €

934

patrocinadores



TheThings NODE

Juan Félix Mateos

AELORA

[www.aelora.nl](http://wwwaelora.nl)



Juan Félix Mateos



BORBORA – CHILDREN AIR QUALITY

www.waag.org



Juan Félix Mateos



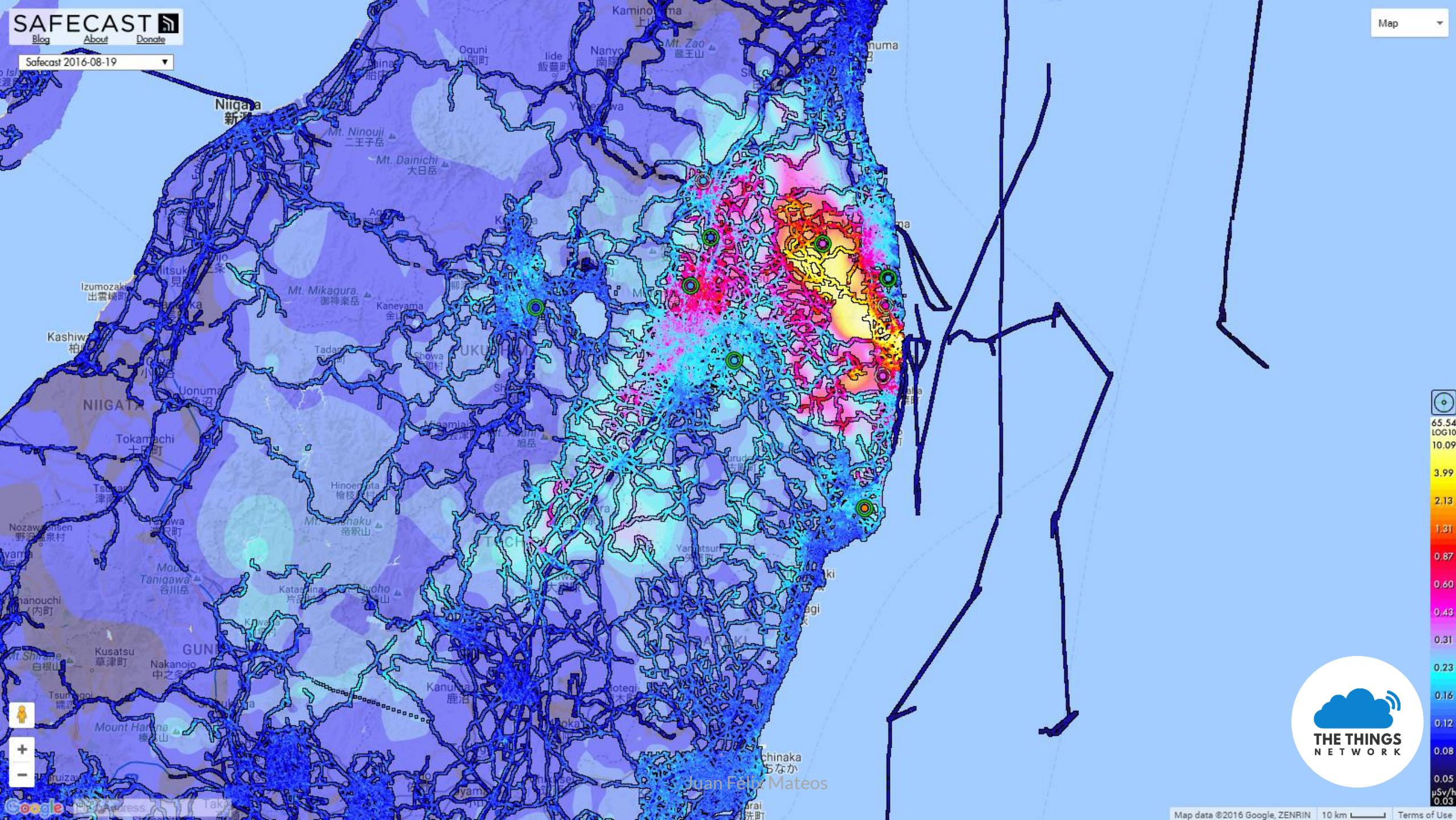
SAFECAST

www.blog.safecast.org/



Juan Félix Mateos

Safecast 2016-08-19 ▾



Juan Félix Mateos

HOOSJE BOOTJE



Juan Félix Matcos





FLOOD.NETWORK

www.flood.network/

Juan Félix Mateos



FLOOD.NETWORK



Juan Félix Mateos



SMART MOUSE TRAP

www.xignal.com



Juan Félix Mateos



THE THINGS NETWORK

CONCEPTOS TÉCNICOS



Conexiones punto a punto

Largo alcance

Bajo consumo de energía

Bajo ancho de banda

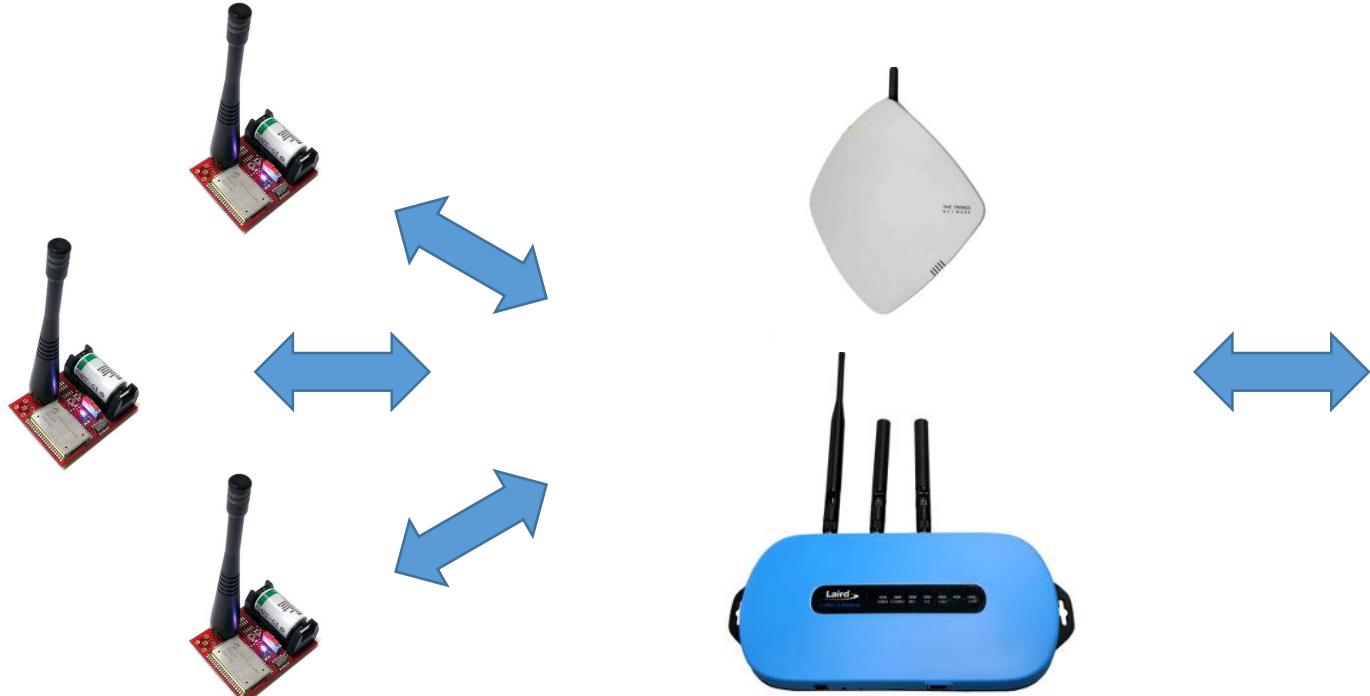
Juan Félix Mateos



Solución local

Juan Félix Mateos



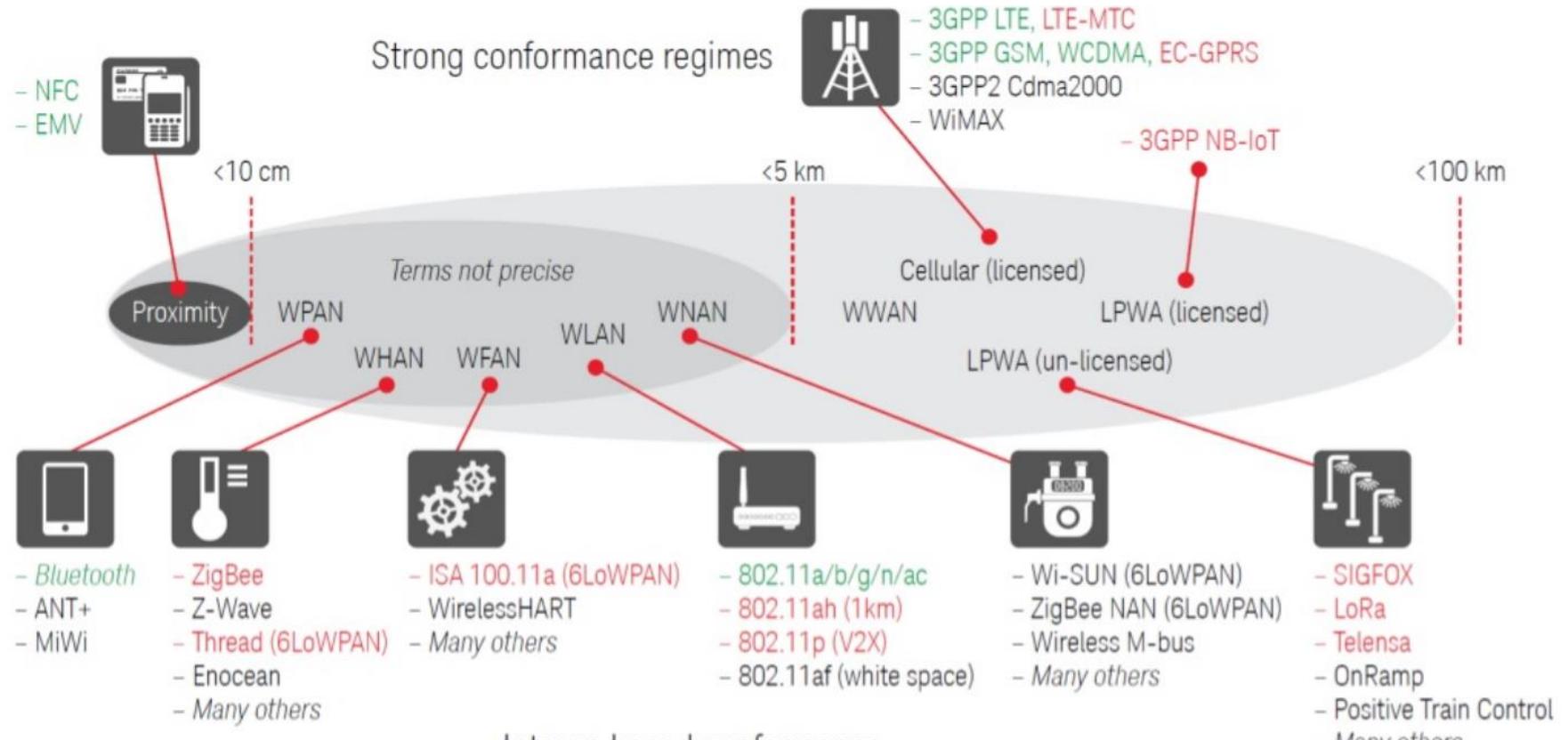


Nodo ↔ Cualquier Gateway TTN ↔ The Things Network

Solución global
Juan Félix Mateos



TECNOLOGÍAS DE CONECTIVIDAD INALÁMBRICA

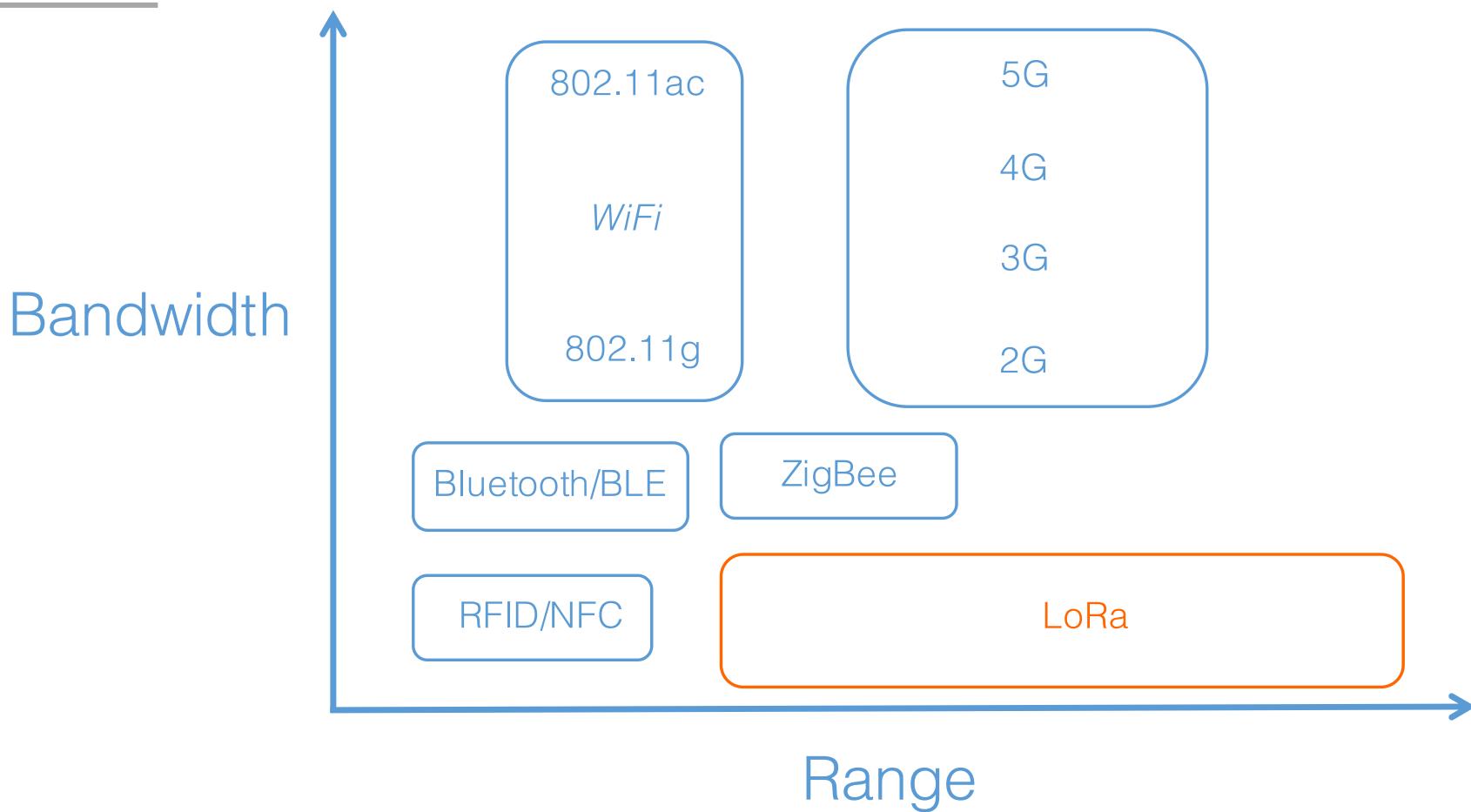


■ : > Billion units/year now
■ : Emerging

WPAN: Wireless Personal Area Network
WHAN: Wireless Home Area Network
WFAN: Wireless Field (factory) Area Network
WLAN: Wireless Local Area Network

WNAN: Wireless Neighborhood Area Network
WWAN: Wireless Wide Area Network
LPWA: Low Power Wide Area

LORA



LoRa

COMPARISON – main LPWAN technologies



Feature	LORAWAN	SIGFOX	LTE Cat 1	LTE M	NB - LTE
Modulation	SS chip	UNB / GFSK / BPSK	OFDMA	OFDMA	OFDMA
Rx Bandwidth	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 Output 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

Juan Félix Mateos

www.vertical-m2m.com



LORA

ALCANCE

Yes No

~1Km (urbano) / ~10Km (rural)

BAJO CONSUMO

Yes No

La baterías duran años

MILES DE DISPOSITIVOS

Yes No

Conectados a un mismo gateway

BANDA SIN LICENCIAS

Yes No

868MHz (EU) 915MHz (US)

ANCHO DE BANDA

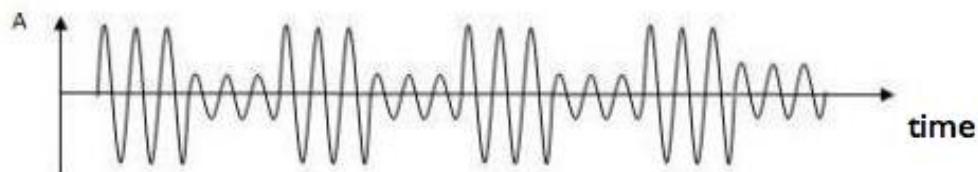
Yes No

Max 50kb/s

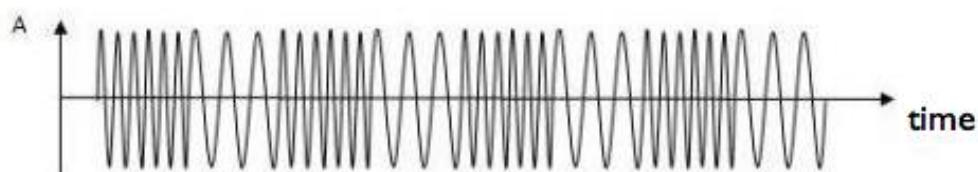
Modulación de radio

1 0 1 0 1 0 1 0

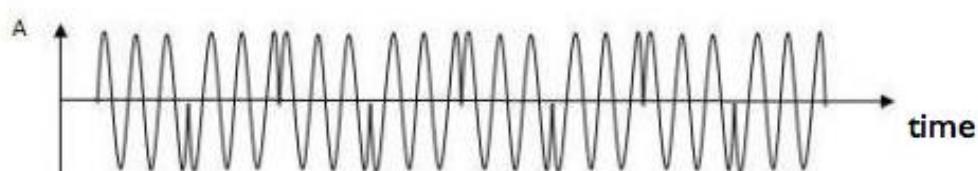
You want to transmit this
binary code



Amplitude modulation



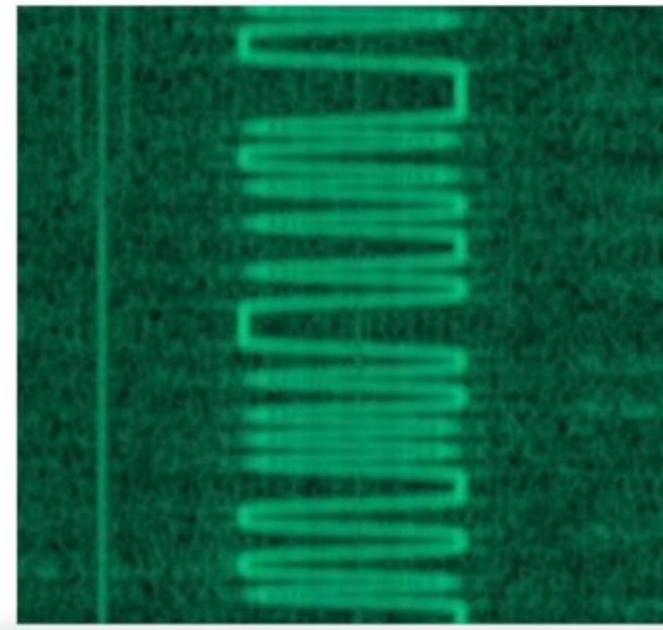
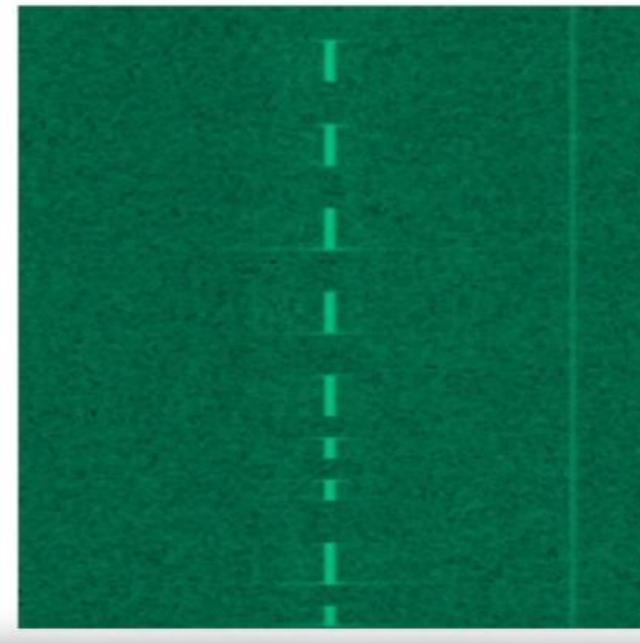
Frequency modulation



Phase modulation

Modulación:

On-Off keying VS Frequency-shift keying VS LoRa



Juan Félix Mateos

<http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html>



Modulación LoRa

Chirp Spread Spectrum

Tecnología de los 40

Utilizada en ámbitos militares → Rádar

Largo alcance y penetración

Difícil de detectar

Alta resistencia a interferencias

LoRa: Bit, Symbol, Chip, Chirp

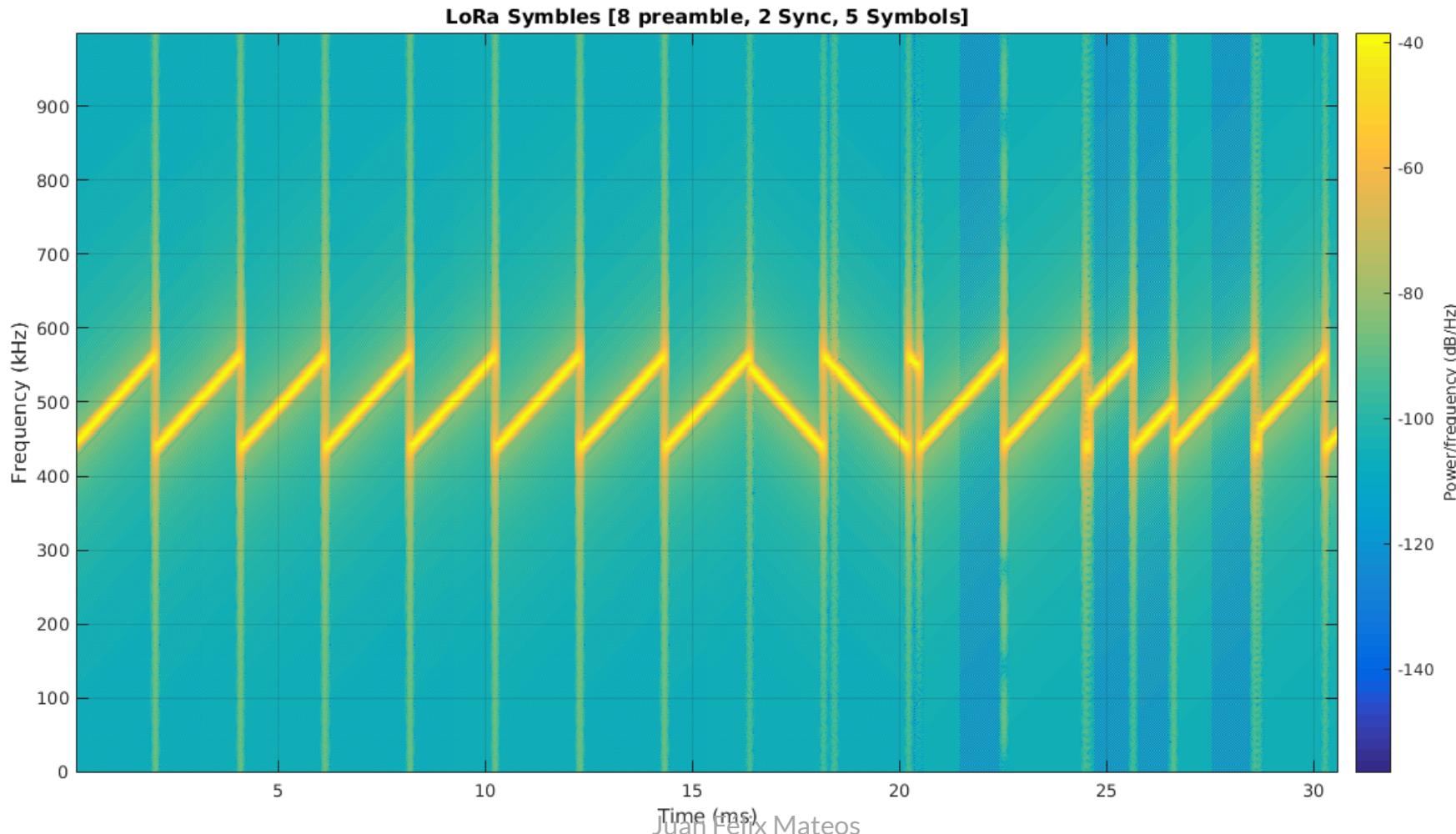
Bit: Unidad mínima de información a transmitir

Symbol: Codificación de la información para transmitirla a través de un medio determinado. Un symbol puede codificar un bit o más, y añadir información adicional como corrección de errores.

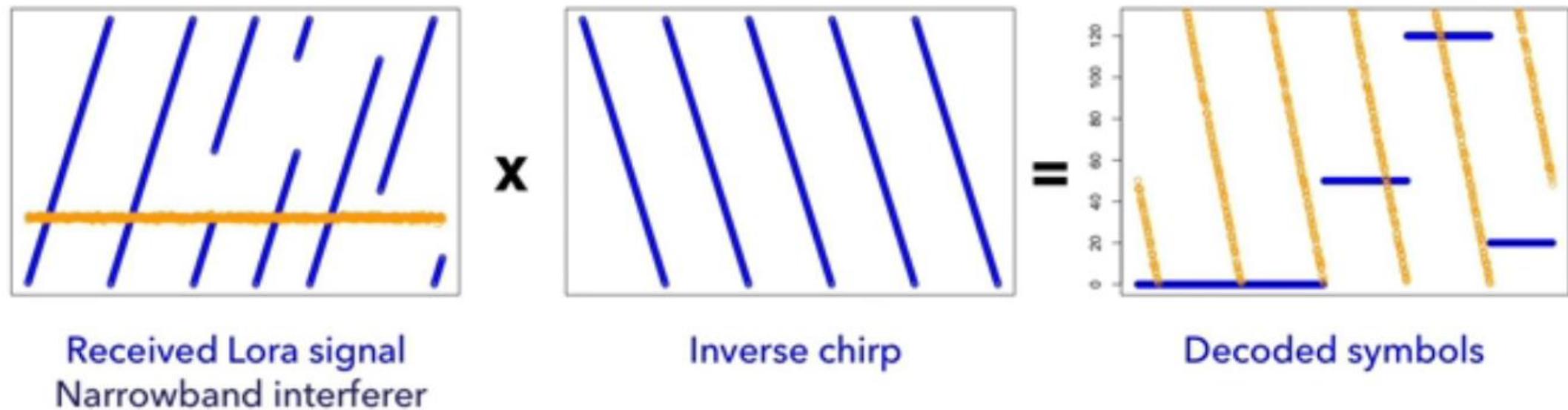
Chip: Unidad mínima de codificación en Spread Spectrum: Se necesitan varios chips para codificar un symbol.

Chirp: Es un Symbol en la terminología de Spread Spectrum.

LoRa Frame



LoRa Demodulación

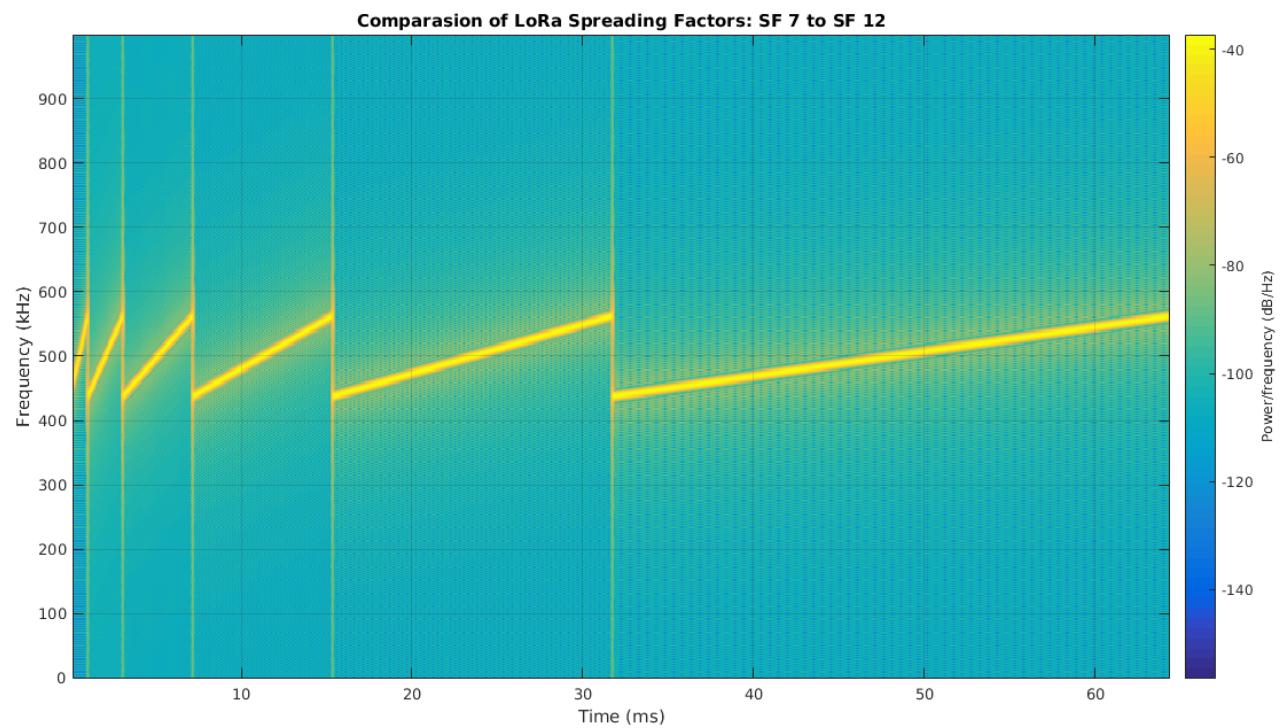


LoRa: Spread Factor

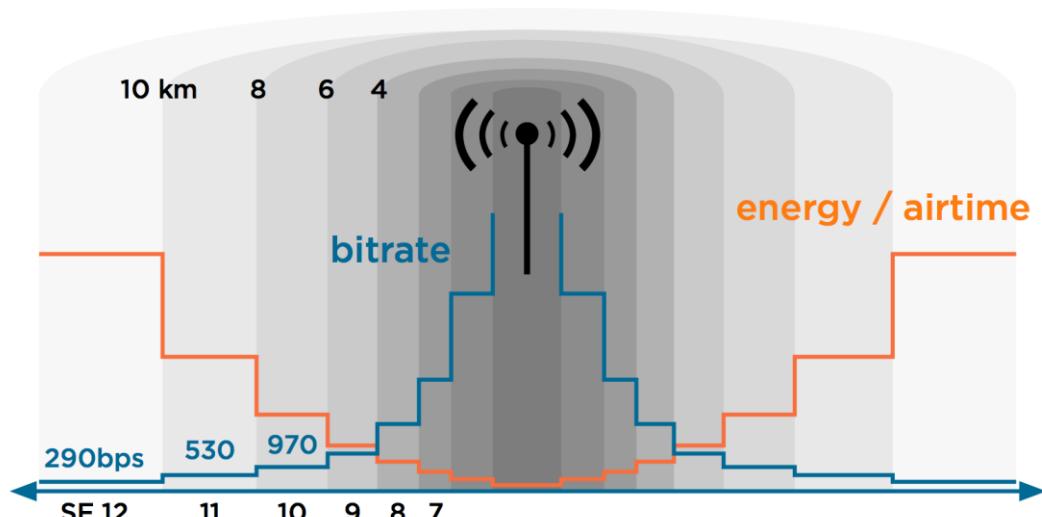
Velocidad vs Alcance vs Consumo

A mayor velocidad, menor alcance

A mayor velocidad, menor
consumo



LoRa: Spread Factor



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

LoRa Spreading Factors (125kHz bw)

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/>

LoRa: Duty Cycle – 36 segundos por hora

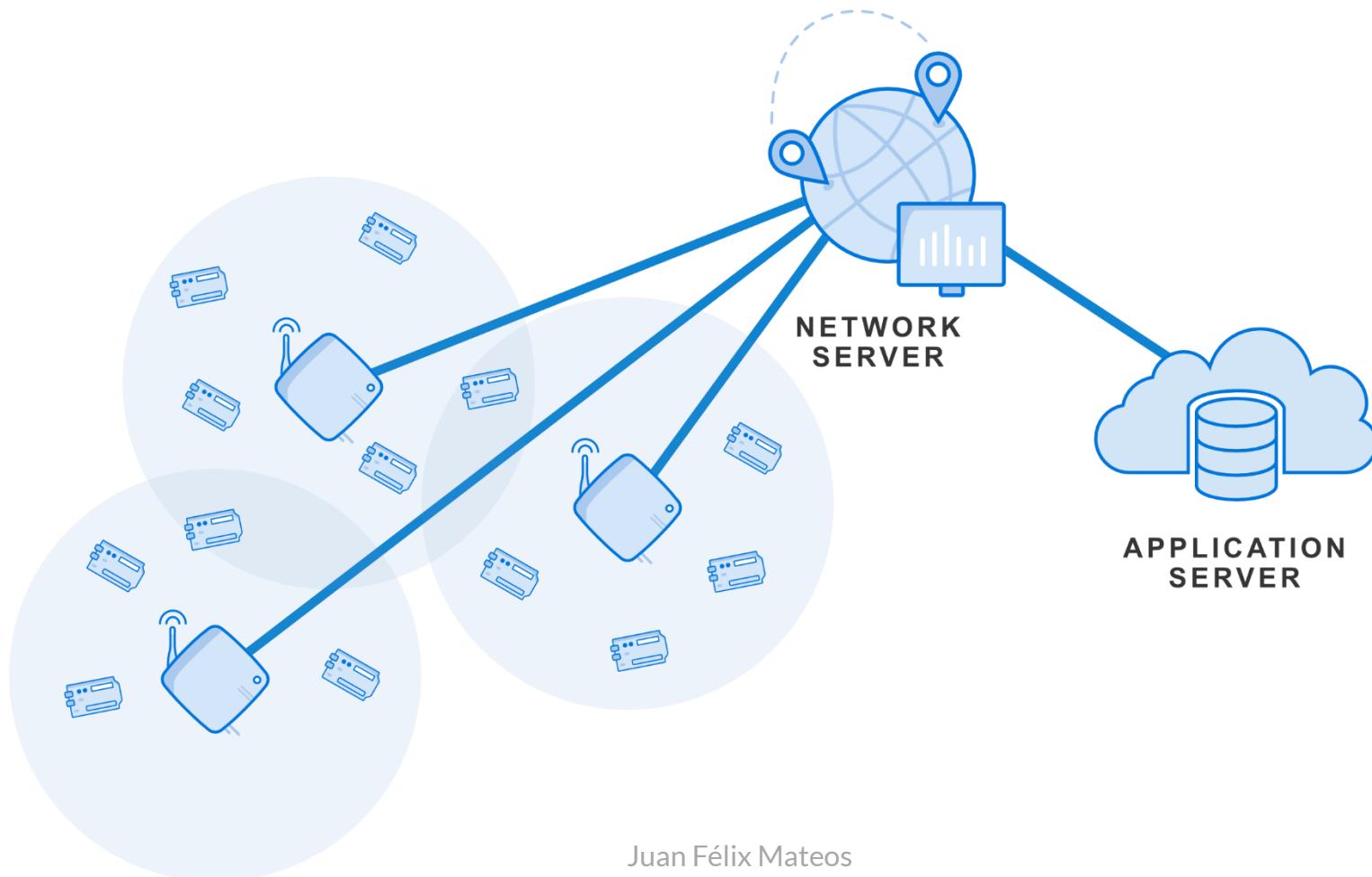
Sección 7.2.3 de la norma ETSI EN300.220

- g (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%

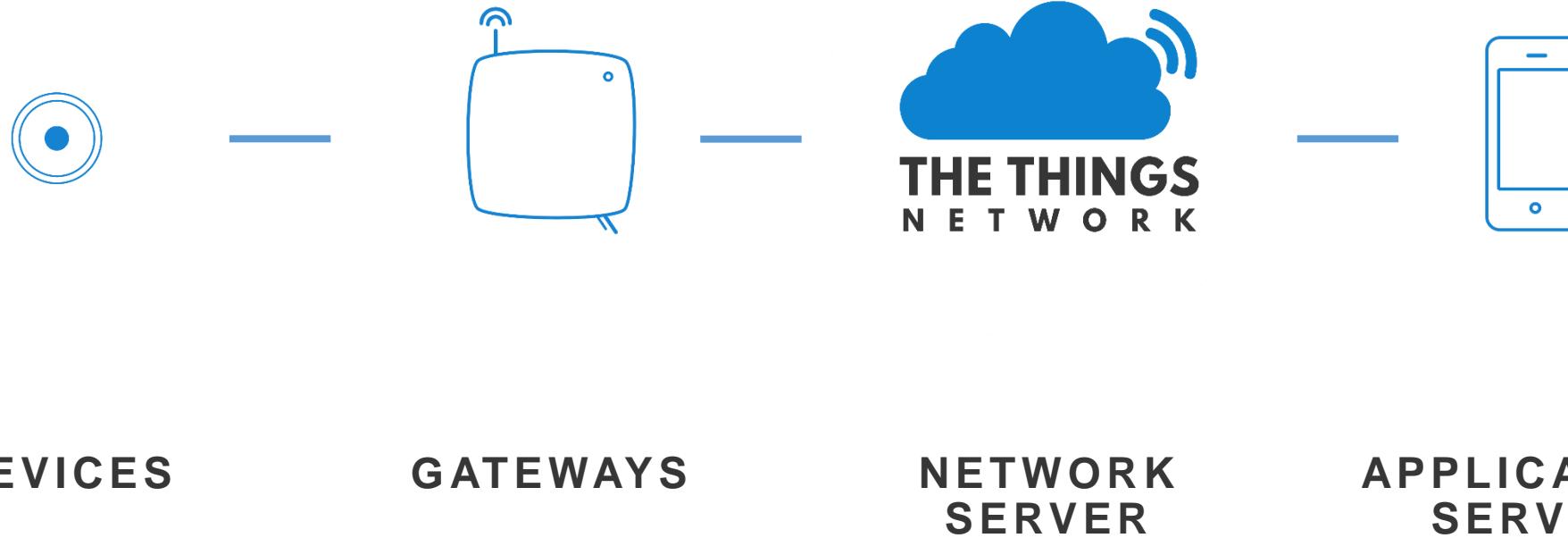
The Things Network: política de acceso justo

- 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

LoRaWAN

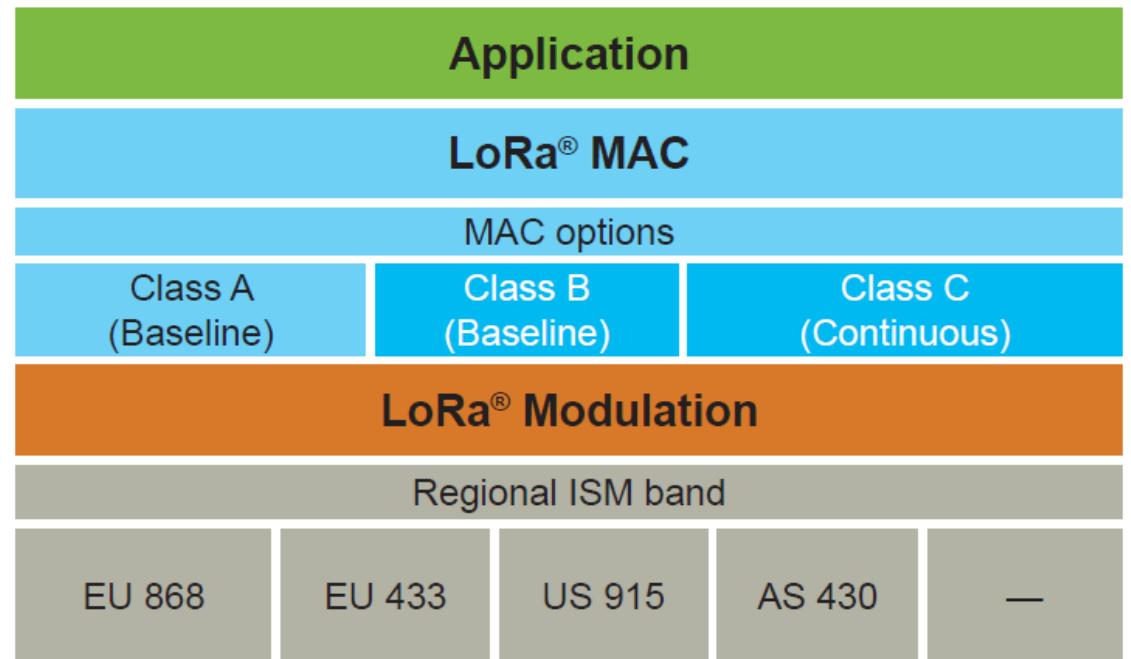


¿Cómo funciona?



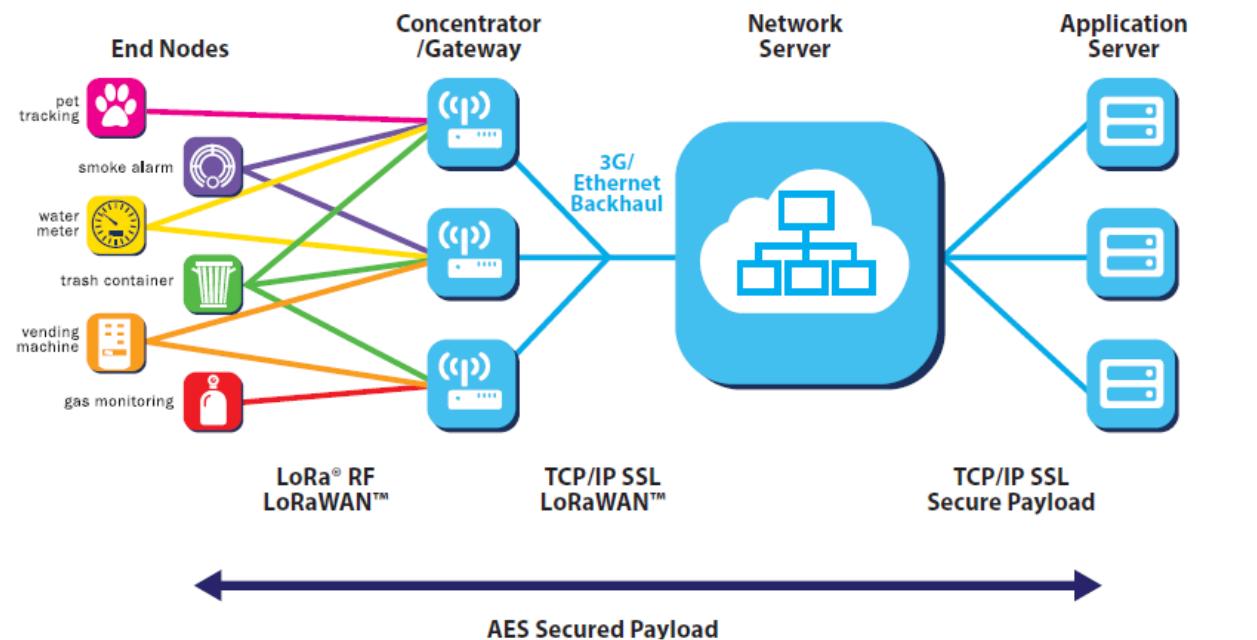
LoRaWAN

- ❑ LoRa → Capa física de largo alcance
- ❑ LoRaWAN → Arquitectura de protocolo y capa de red
- ❑ El protocolo y la arquitectura tienen una gran influencia sobre
 - Consumo de batería
 - Capacidad de la red
 - QoS
 - Aplicaciones finales sobre la estructura



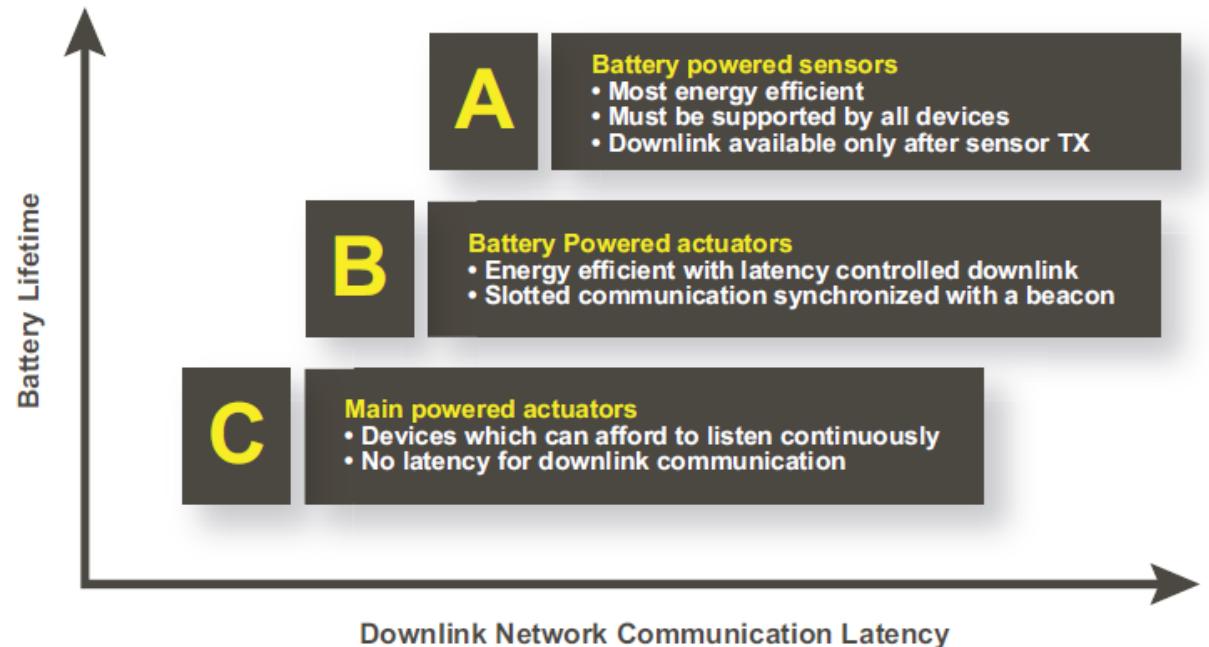
LoRaWAN

- En LoRaWAN no existe asociación entre nodo y gateway
- La inteligencia y la complejidad se desplazan al plano del servidor, que es el que lida con duplicidades, seguridad, planificación de notificaciones,
- No existe el concepto de handover

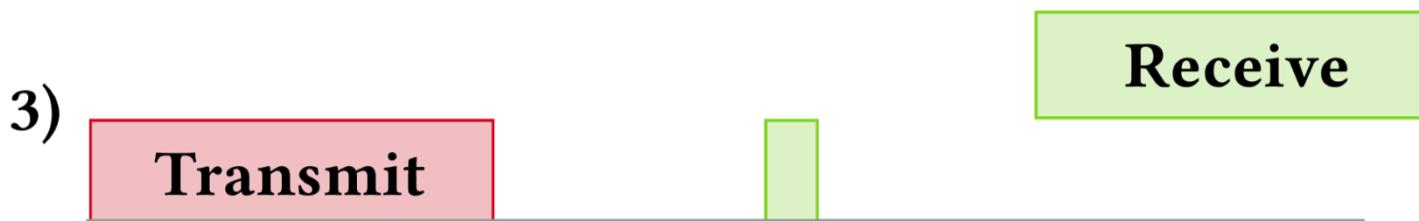
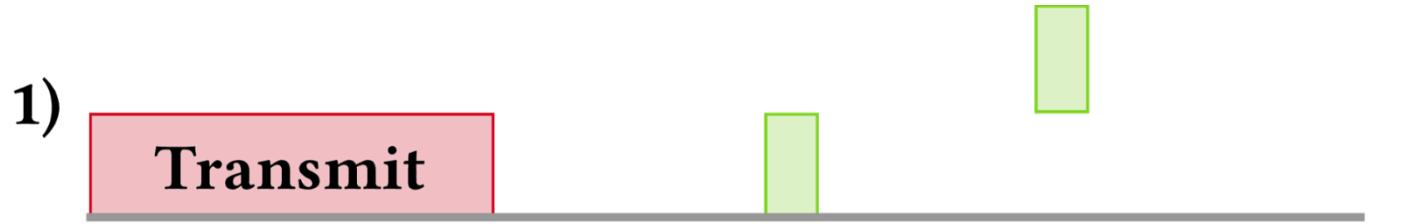


LoRaWAN

- Tres clases de dispositivos en función del uso del espectro:
 - Clase A: Sensores
 - Clase B: Actuadores
 - Clase C: Actuadores críticos



LoRaWAN: Uplink y Downlink en clase A



Receive Delay 1

Juan Felix Mateos
Receive Delay 2

LoRaWAN: Identificación única de los componentes

GatewayEUI (64 bits, único).

DevEUI (64 bits, único)

AppEUI (64 bits, único)

LoRaWAN: Información requerida por el nodo

- **Device Address** (32 bits, puede no ser único)
 - 7 MSB: Network Identifier (0x26, 0x27 = The Things Network)
 - 25 LSB: Network Address
- **Network Session Key**
- **Application Session Key (ABP)**
- **Frame counter (uplink y downlink)**

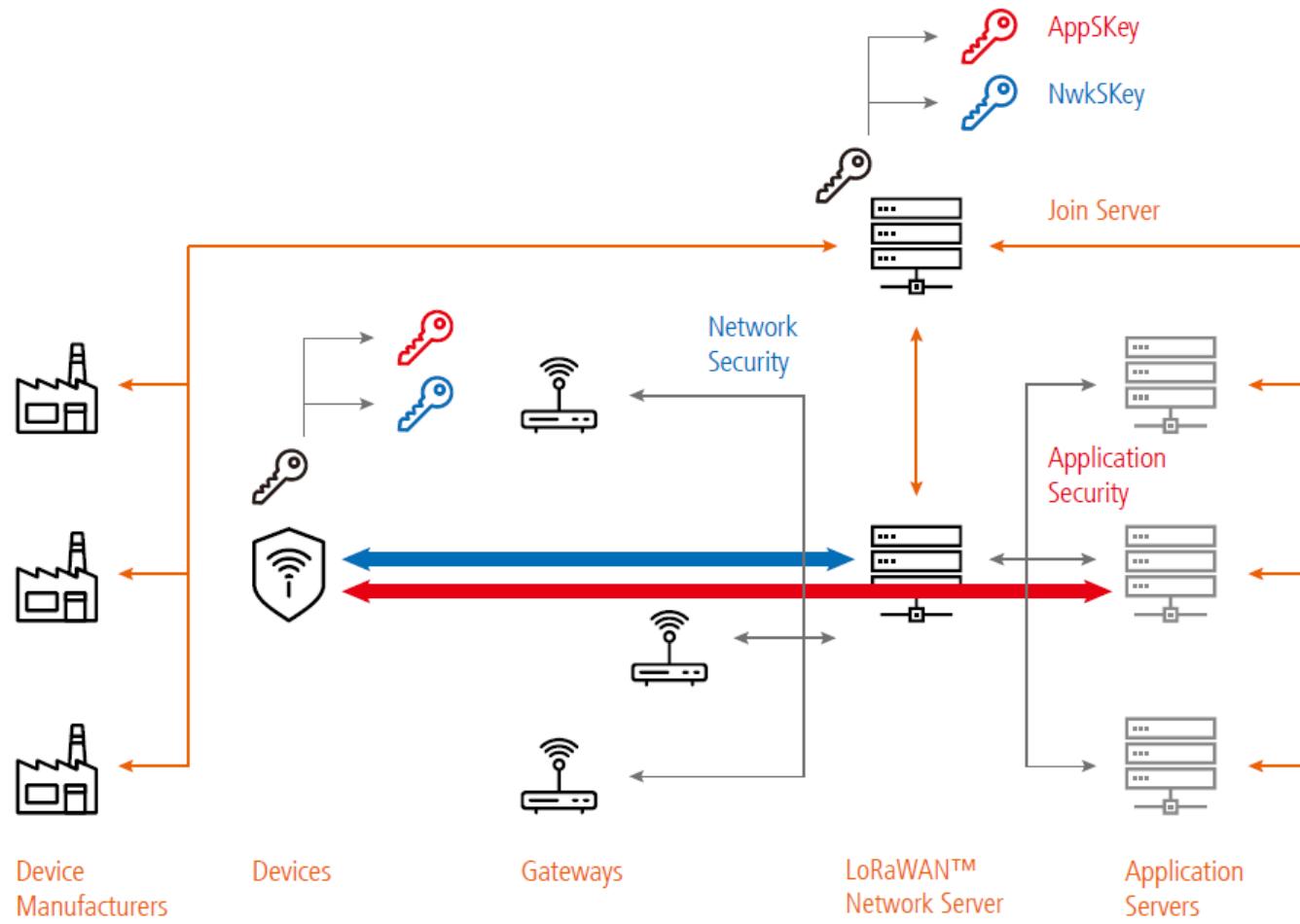
LoRaWAN: Seguridad

Autenticidad

Integridad

Token

Encriptación



LoRaWAN: Personalización y activación OTAA vs ABP

ABP: Activación por personalización

- Devide Address, Network Session Key y Application Session Key programados “a fuego” en el nodo

OTAA: Activación por el aire

- El nodo envía el DeviceEUI y el ApplicationEUI, y a partir de la respuesta recibida y haciendo uso de la ApplicationKey, deriva el Devide Address, Network Session Key y Application Session Key.

LoRaWAN: Formato de un mensaje

Radio PHY layer:

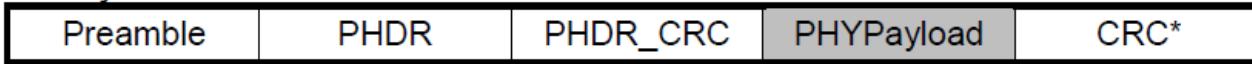


Figure 5: Radio PHY structure (CRC* is only available on uplink messages)

PHYPayload:



or



or



Figure 6: PHY payload structure

MACPayload:



Figure 7: MAC payload structure

FHDR:



Figure 8: Frame header structure



THE THINGS NETWORK

APLICACIONES CIUDADANAS

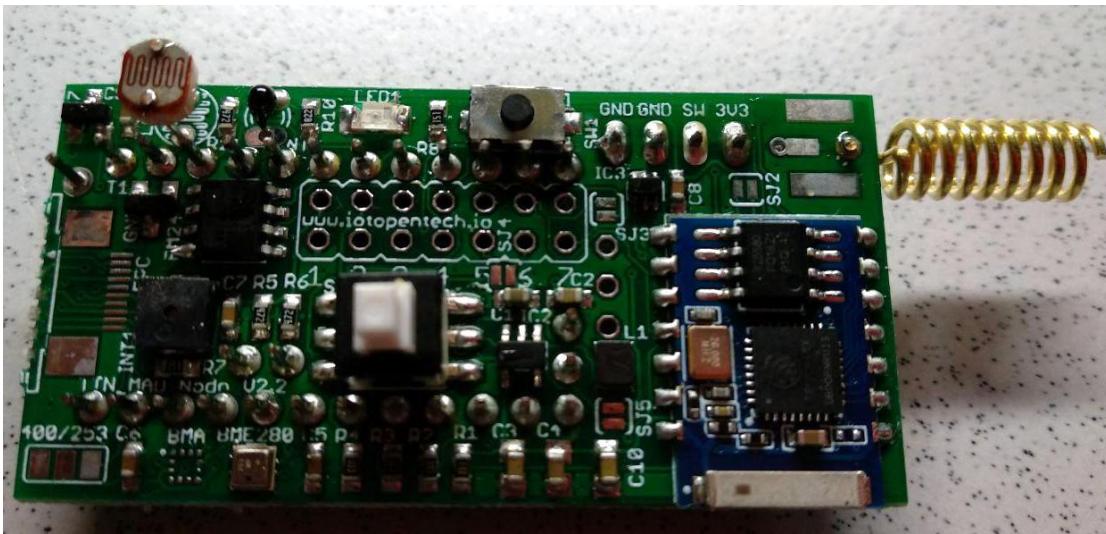


¿Qué es el nodo TTN MAD v2.2?

Es un nodo LoRaWAN diseñado con los siguientes requerimientos:

- Fácil de montar
- Económico (monetaria y energéticamente)
 - Ampliable
 - Muy flexible.

Nodo TTN MAD v2.2



Juan Félix Mateos

La consola de The Things Network

Vamos a crear una aplicación nueva y, dentro de ella, un dispositivo que se una a la red por ABP.



Crear una aplicación

The image consists of three panels illustrating the steps to create a new application in The Things Network Console:

- Panel 1:** Shows the main dashboard with a large "APPLICATIONS" icon and a red circle labeled "1".
- Panel 2:** Shows the "Applications" list page with a red arrow pointing to a green "Add Application" button, and a red circle labeled "2".
- Panel 3:** Shows the "Add Application" form. A red circle labeled "3" highlights the "Application ID" field containing "jfmateos_hackmad". A red circle labeled "4" highlights the "Add application" button at the bottom right.

Juan Félix Mateos

Añadir un dispositivo

The image shows two screenshots illustrating the process of adding a device to a TTN application.

Left Screenshot (Application Overview):

- Shows the application details: Application ID **jfmateos_hackmad**, Description, Created 34 seconds ago, Handler **ttn-handler-eu (current handler)**.
- Shows the **APPLICATION EUIS** section with a copyable EUI: **70 B3 D5 7E D0 02 40 BF**.
- Shows the **DEVICES** section with a count of **0 registered devices**.
- An arrow labeled **1** points to the **register device** button at the bottom right of the devices table.

Right Screenshot (Register Device Form):

- Shows the **REGISTER DEVICE** form.
- Step **2**: **Device ID** field contains **jfmateos_hackmad_01**.
- Step **3**: **Device EUI** field contains **70 B3 D5 7E D0 02 40 BF**. A note says "this field will be generated". An arrow points from the left screenshot's EUI to this field.
- Step **4**: **Register** button at the bottom right.

Configurar el dispositivo para activarse por ABP

The screenshot illustrates the configuration process for a device to support Activation by Beacon (ABP). It consists of two main parts: the Device Overview page and the Settings page.

Device Overview Page (Left):

- Path: Applications > jfmateos_hackmad > Devices > jfmateos_hackmad_01
- Overview tab selected.
- Device ID: jfmateos_hackmad_01
- Activation Method: OTAA (highlighted with a yellow box).
- Device EUI: 00 00 00 00 00 00 00 00
- Application EUI: 70 B3 D5 7E D0 02 40 BF
- App Key: A series of dots (...)

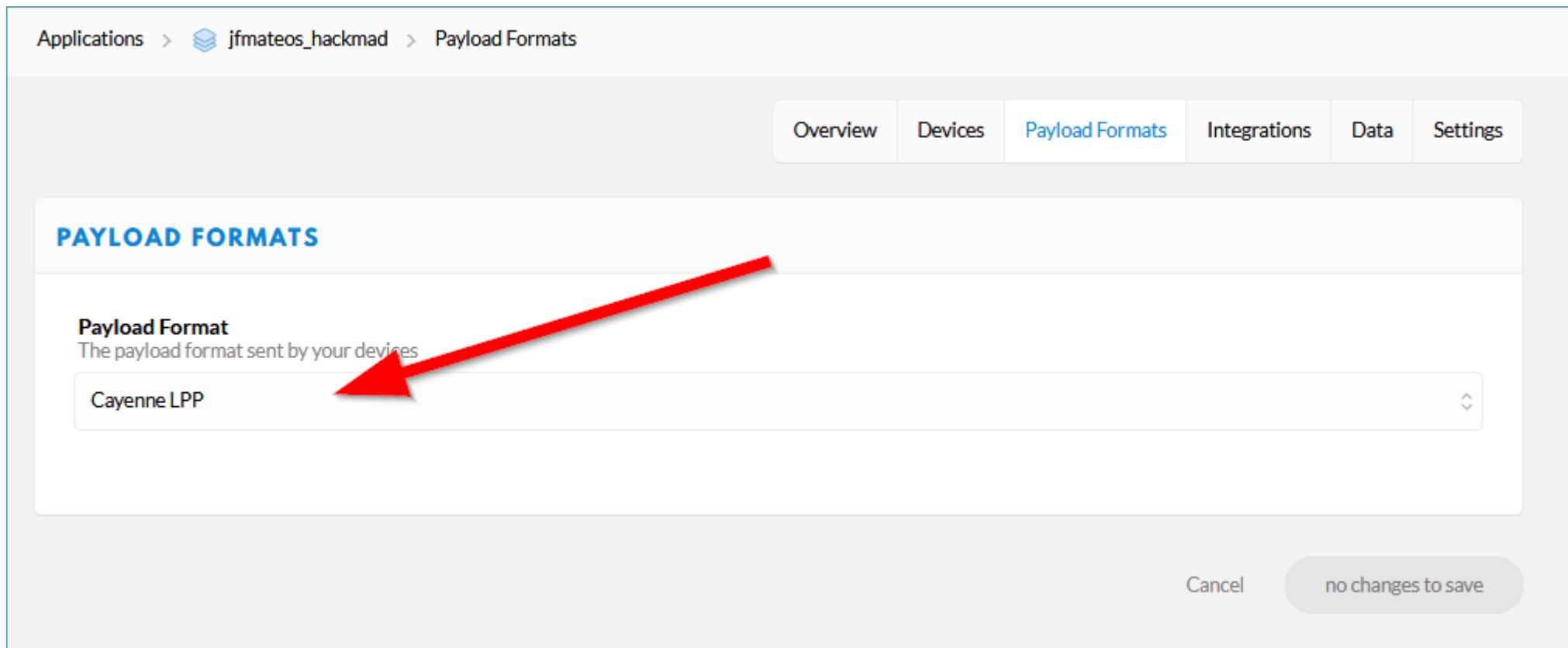
Settings Page (Right):

- Path: Applications > jfmateos_hackmad > Devices > jfmateos_hackmad_01 > Settings
- Activation Method tab selected (highlighted with a yellow box).
- Activation Method dropdown: OTAA (disabled) and ABP (selected, highlighted with a blue box).
- Device Address: The device address will be assigned by the network server.
- Network Session Key: Network Session Key will be generated.
- App Session Key: App Session Key will be generated.
- Frame Counter Width: Set to 1 (highlighted with a red circle labeled 1).
- Frame Counter Checks: A checkbox labeled "Frame Counter Checks" is checked (highlighted with a red circle labeled 2).
- Information: Disabling frame counter checks drastically reduces security and should only be used for development purposes.
- Buttons at the bottom: Cancel (disabled) and Save (highlighted with a green box labeled 4).

Juan Félix Mateos

THE THINGS NETWORK

Activar el formato de carga de pago Cayenne LPP



The screenshot shows a web interface for managing payload formats in a Cayenne application. The top navigation bar includes 'Applications' (with icon), 'jfmateos_hackmad', and 'Payload Formats'. Below the navigation is a horizontal menu with tabs: 'Overview', 'Devices', 'Payload Formats' (which is blue, indicating it's selected), 'Integrations', 'Data', and 'Settings'. A large central box is titled 'PAYLOAD FORMATS'. Inside, a section labeled 'Payload Format' describes the payload format sent by devices. A dropdown menu contains the option 'Cayenne LPP', which is highlighted with a large red arrow pointing to it. At the bottom right of the central box are 'Cancel' and 'no changes to save' buttons.

Programa en Arduino para el nodo TTN MAD v2.2

Vamos a crear un programa que envíe un frame con la tensión de la batería cada 30 segundos.

Librerías de Arduino

Instalar las siguientes librerías en el IDE de Arduino:

- MCCI LoRaWAN LMIC Library by IBM, Matthijs Kooijman, Terry Moore, ChaeHee Won, Frank Rose. Versión 2.3.2.
- Low-Power by Rocket Scream Electronics. Versión 1.6.0.
- CayenneLPP by Electronics Cats. Versión 1.0.1

Configurar la librería MCCI LoRaWAN LMIC Library

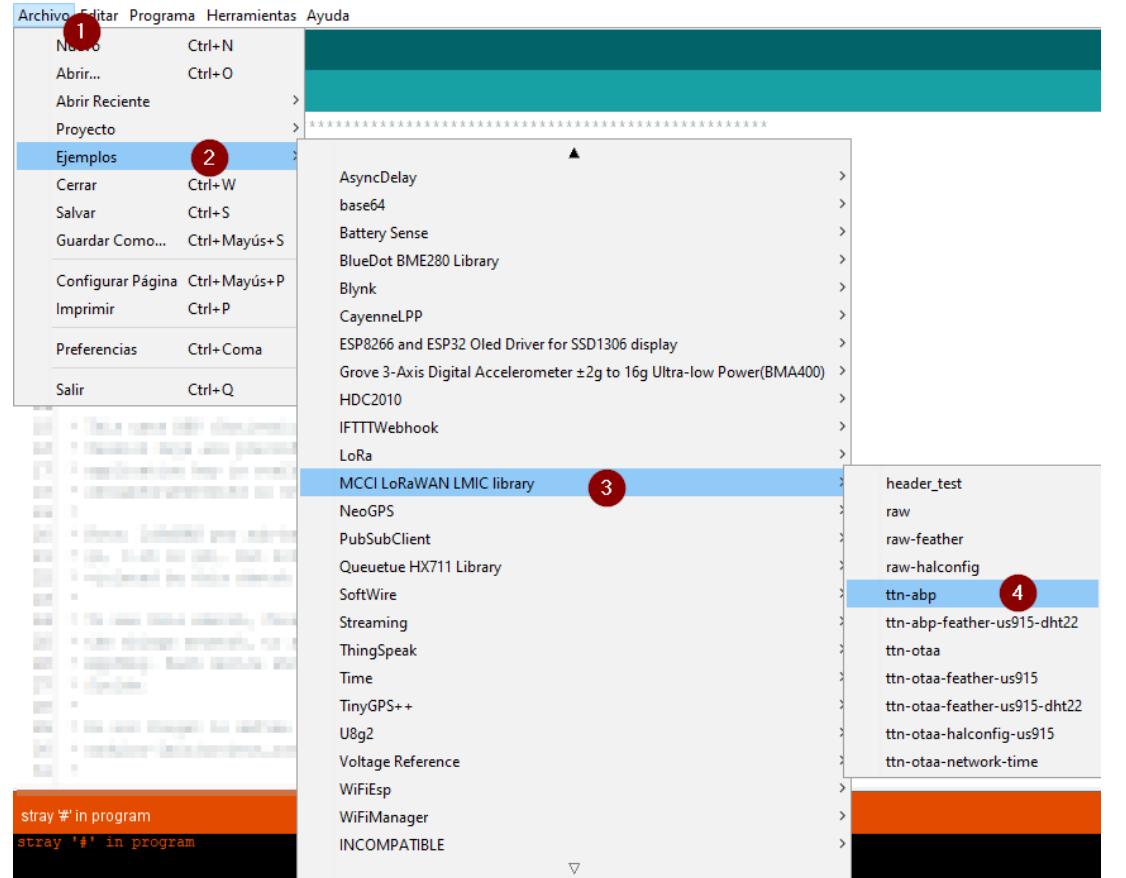
Para reducir al máximo los requisitos de memoria de la librería vamos a eliminar las funcionalidades que no necesitamos modificando el siguiente archivo:

MCCI_LoRaWAN_LMIC_library\project_config\lmic_project_config.h

```
// 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      /* for as923-JP */  
  
//##define CFG_in866 1  
  
#define CFG_sx1276_radio 1  
  
//##define LMIC_USE_INTERRUPTS  
  
#define DISABLE_PING  
  
#define DISABLE_BEACONS
```

Punto de partida

- Partimos del ejemplo ttn-abp de la librería MCCI LoRaWAN LMIC Library



Indicar las credenciales del nodo

- Copiamos el NwkSkey (msb), AppSkey (msb) y DevAddr (expresar en hexadecimal anteponiendo 0x) de la consola de TTN en los lugares correspondientes del código.

The screenshot shows the Arduino IDE interface with the sketch `tt_nmad_v2_2_basico`. The left sidebar shows the project structure: `applications > jfmateos_hackmad > Devices > jfmateos_hackmad_01`. The main area is divided into two sections: **DEVICE OVERVIEW** and **CODE EDITOR**.

DEVICE OVERVIEW:

- Application ID: `jfmateos_hackmad`
- Device ID: `jfmateos_hackmad_01`
- Activation Method: `ABP`
- Device EUI: `00 95 A7 08 96 89 D5 0D`
- Application EUI: `70 B3 D5 7E D1 02 40 BF`
- Device Address: `26 01 11 38`
- Network Session Key: `{ 0x65, 0xBB, 0xE5, 0xF5 }`
- App Session Key: `{ 0xFB, 0x78, 0x52, 0x47 }`
- Status: `never seen`
- Frames up: 0 [reset frame counters](#)

CODE EDITOR:

```
tt_nmad_v2_2_basico
...
// We want to be able to compile these scripts. In fact,
// COMPILER_REGRESSION_TEST, and in that case we define
// working but innocuous value.
//
#ifndef COMPILER_REGRESSION_TEST
#define FILLMEIN 0
#else
// warning "You must replace the values marked FILLMEIN"
#define FILLMEIN (#dont edit this, edit the lines that follow)
#endif

// LoRaWAN NwkSKey, network session key
static const PROGMEM ul_t NWSKEY[16] = { 0x65, 0xBB, 0xE5, 0xF5 };
// LoRaWAN AppSKey, application session key
static const ul_t PROGMEM APPSKEY[16] = { 0xFB, 0x78, 0x52, 0x47 };
// LoRaWAN end-device address (DevAddr)
// See http://thethingsnetwork.org/wiki/AddressSpace
// The library converts the address to network byte order
static const ul_t DEVADDR = 0x26011138; // <<-- Change this to your device address
// These callbacks are only used in over-the-air activation
// Left empty here for now, can leave them out completely
<<-- Change this to your device address

Compiled
El Sketch usa 278284 bytes (55%) del espacio de almacenamiento
Las variables Globales usan 28240 bytes (34%) de la memoria

```

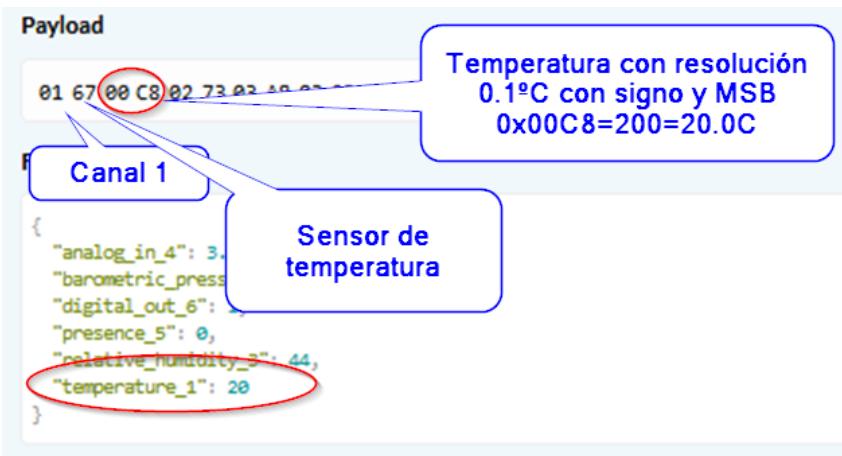
Modificar los pines que controlan el RFM95W

- Configurar la sección //Pin mapping como se indica

```
// Pin mapping  
  
const lmic_pinmap lmic_pins = {  
  
    .nss = 10,  
  
    .rxtx = LMIC_UNUSED_PIN,  
  
    .rst = 9,  
  
    .dio = {2, 7, LMIC_UNUSED_PIN},  
  
};
```

Cayenne LPP: Tipos de datos

- Cada sensor/actuador tiene asociado un tipo de dato



Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB Longitude : 0.0001 ° Signed MSB Altitude : 0.01 meter Signed MSB

Incluir las librerías lowpower y cayenne

```
37 #include <lmic.h>
38 #include <hal/hal.h>
39 #include <CDT.h>
40 #include "LowPower.h"
41 #include <CayenneLPP.h>
42 CayenneLPP lpp(4);
43 //
```

Insertar la función readVCC

<https://gist.github.com/IoTopenTech>

```
long readVcc() {  
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) |  
    _BV(MUX1);  
    delay(2);  
    ADCSRA |= _BV(ADSC);  
    while (bit_is_set(ADCSRA, ADSC));  
    long result = ADCL;  
    result |= ADCH << 8;  
    result = 1126400L / result; // Back-calculate AVcc in mV  
    return result;  
}  
  
86     .rxtx = LMIC_UNUSED_PIN,  
87     .rst = 9,  
88     .dio = {2, 7, LMIC_UNUSED_PIN},  
89 };  
90 long readVcc () {  
91     ADMUX = _BV (REFS0) | _BV (MUX3) | _BV (MUX2) | _BV (MUX1) ;  
92     delay (2) ;  
93     ADCSRA |= _BV (ADSC) ;  
94     while ( bit_is_set (ADCSRA, ADSC) ) ;  
95  
96     long result = ADCL;  
97     result |= ADCH << 8;  
98     result = 1126400L / result; // Back-calculate AVcc in mV  
99     return result;  
100 }  
101 void onEvent (ev_t ev) {  
102     Serial.print (os_getTime ()) ;  
103     Serial.print (": ");
```

Modificar el código de la función do_send

```
// Prepare upstream data transmission at the
next possible time.
lpp.reset();
lpp.addAnalogInput(1, readVcc() / 1000.F);
//LMIC_setTxData2(1, mydata,
sizeof(mydata)-1, 0);
LMIC_setTxData2(1, lpp.getBuffer(),
lpp.getSize(), 0);
Serial.println(F("Packet queued"));
```

```
181 }
182
183 void do_send(osjob_t* j){
184     // Check if there is not a current TX/RX job running
185     if (LMIC.opmode & OP_TXRXPEND) {
186         Serial.println(F("OP_TXRXPEND, not sending"));
187     } else {
188         // Prepare upstream data transmission at the next possible time.
189         lpp.reset();
190         lpp.addAnalogInput(1, readVcc() / 1000.F);
191         //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);
192         LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(), 0);
193         Serial.println(F("Packet queued"));
194     }
195     // Next TX is scheduled after TX COMPLETE event.
196 }
197
198 void setup() {
```

Modificar el código del estado EV_TXCOMPLETE

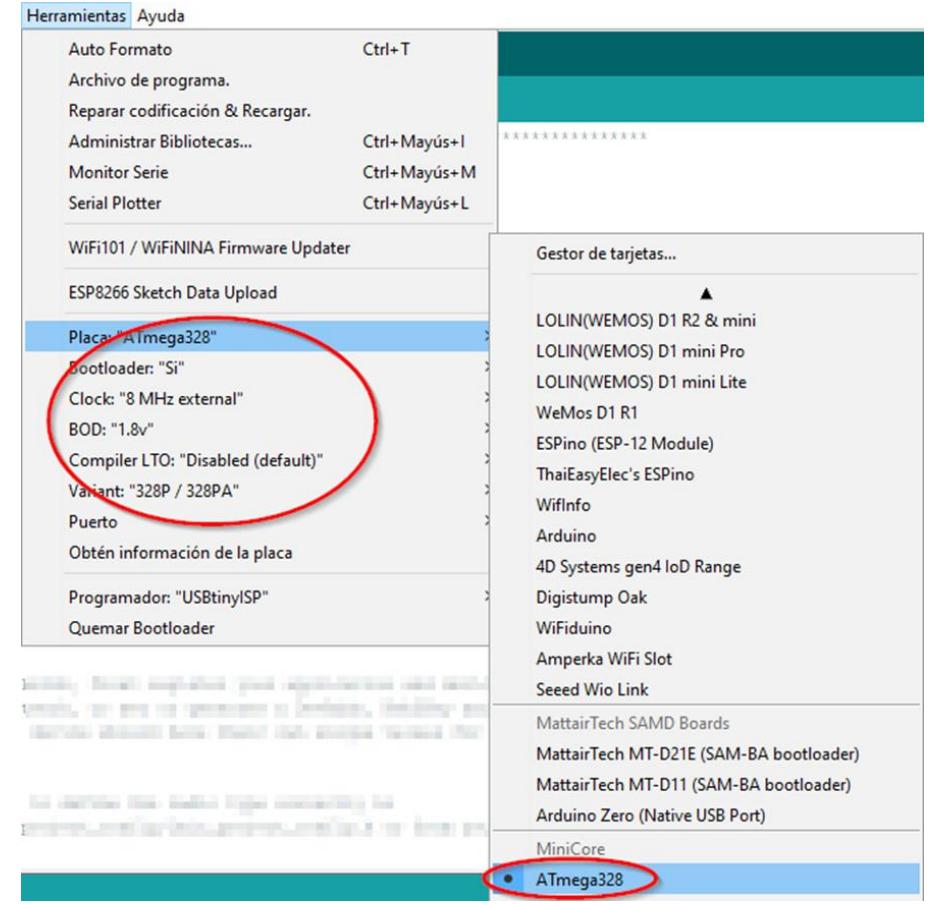
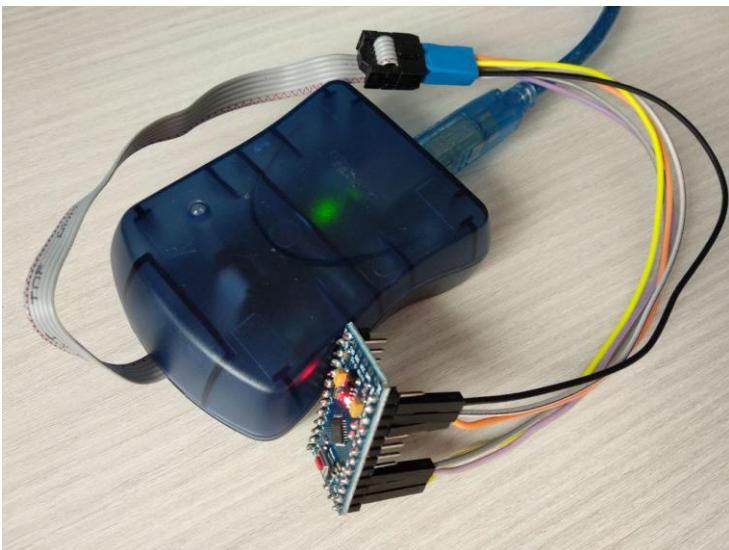
```
Serial.println("Me voy a echar  
una siestecita");  
delay(1000); // Para que  
terminen de imprimirse los  
mensajes en el terminal  
for (byte contador = 0;  
contador < 3; contador++) {  
  
LowPower.powerDown(SLEEP_8S,  
ADC_OFF, BOD_OFF);  
}  
os_setCallback (&sendjob,  
do_send);
```

```
137 case EV_TXCOMPLETE:
138     Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
139     if (LMIC.txrxFlags & TXRX_ACK)
140         Serial.println(F("Received ack"));
141     if (LMIC.dataLen) {
142         Serial.println(F("Received "));
143         Serial.println(LMIC.dataLen);
144         Serial.println(F(" bytes of payload"));
145     }
146     // Schedule next transmission
147     //os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL), do_send);
148     Serial.println("Me voy a echar una siestecita");
149     delay(1000); // Para que terminen de imprimirse los mensajes en el terminal
150     for (byte contador = 0; contador < 3; contador++) {
151         LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
152     }
153     os_setCallback (&sendjob, do_send);
154     break;
155 case EV_LOST_TSYNC:
```

Sustituir el bootloader por el de MiniCore (opcional)

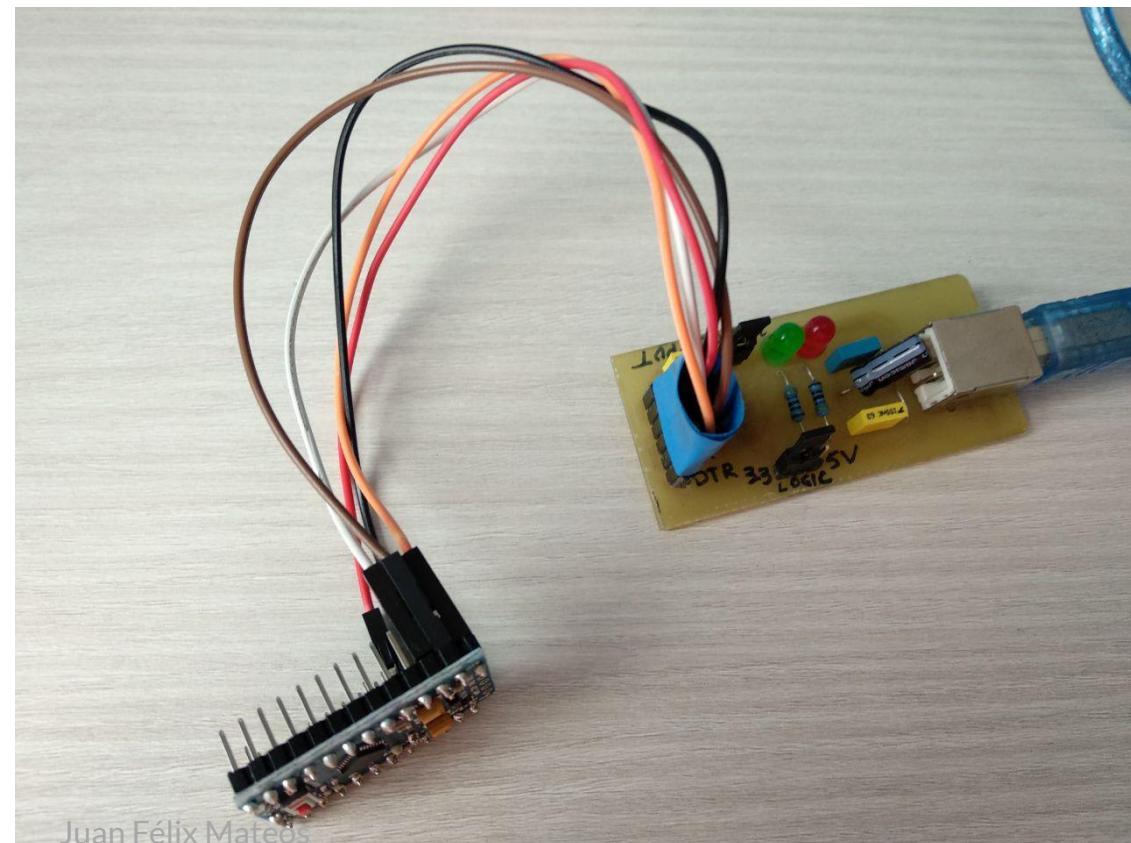
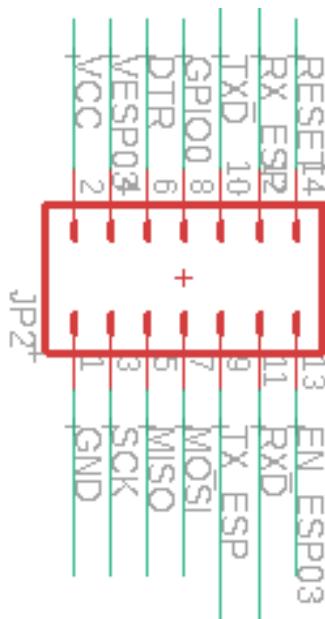
Sustituir el bootloader del Arduino Pro Mini por la versión de MiniCore (<https://github.com/MCUduude/MiniCore>) usando un programador (por ejemplo, el USBtinyISP). Para conectar el programador al Arduino se usarán los siguientes pines:

- VCC
- GND
- RST
- 11: MOSI
- 12: MISO
- 13: SCK



Programar el Arduino Pro Mini

Cargar el programa en el Arduino Pro Mini
usando un conversor USB → Serial



Juan Félix Mateos

Puesta en funcionamiento

Insertar las baterías.

Pulsar el botón de encendido.

Los frames empezarán a aparecer en la consola de The Things Network.

APPLICATION DATA					
time	counter	port	dev id:	payload:	analog_in_1:
▲ 00:02:06	19	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 00:01:34	18	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 00:01:01	17	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 00:00:27	16	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 23:59:55	15	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 23:59:21	14	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 23:58:48	13	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 23:58:15	12	1	ifmateos_hackmad_01	01 02 01 50	3.36
▲ 23:57:42	11	1	ifmateos_hackmad_01	01 02 01 50	3.36

BME280: Incluir las librerías y ampliar la carga de pago

```
36  
37 #include <lmic.h>  
38 #include <hal/hal.h>  
39 #include <Wire.h> // Circulo rojo  
40 #include <SPI.h>  
41 #include "LowPower.h"  
42 #include <Adafruit_Sensor.h>  
43 #include <Adafruit_BME280.h>  
44 //En el archivo Adafruit_BME280.h hay que cambiar la siguiente linea  
45 // #define BME280_ADDRESS (0x76)  
46 #define SEALEVELPRESSURE_HPA (1013.25)  
47 Adafruit_BME280 bme; // I2C  
48  
49 #include <CayenneLPP.h>  
50 CayenneLPP lpo(16); //Sólo voy a pasar la tensión y el BME280  
51
```

BME280: Añadir la temperatura, presión y humedad

```
if (bme.begin()) {  
  
    bme.setSampling(Adafruit_BME280::MODE_FORCED,  
                    Adafruit_BME280::SAMPLING_X1, //  
temperature  
                    Adafruit_BME280::SAMPLING_X1, //  
pressure  
                    Adafruit_BME280::SAMPLING_X1, //  
humidity  
                    Adafruit_BME280::FILTER_OFF );  
    lpp.addTemperature(2, bme.readTemperature());  
    lpp.addBarometricPressure(3,  
bme.readPressure() / 100.0F);  
    lpp.addRelativeHumidity(4, bme.readHumidity());  
}
```

```
200     lpp.reset();  
201     lpp.addAnalogInput(1, readVcc() / 1000.F);  
202  
203     //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);  
204     if (bme.begin()) {  
205         bme.setSampling(Adafruit_BME280::MODE_FORCED,  
206                         Adafruit_BME280::SAMPLING_X1, // temperature  
207                         Adafruit_BME280::SAMPLING_X1, // pressure  
208                         Adafruit_BME280::SAMPLING_X1, // humidity  
209                         Adafruit_BME280::FILTER_OFF );  
210         lpp.addTemperature(2, bme.readTemperature());  
211         lpp.addBarometricPressure(3, bme.readPressure() / 100.0F);  
212         lpp.addRelativeHumidity(4, bme.readHumidity());  
213     }  
214     LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(), 0);  
215     //LMIC_setTxData2(1, mydata, sizeof(mydata)-1, 0);  
216     Serial.println(F("Packet queued"));  
217 }
```

BME280: Comprobar en la consola

Applications >  jfmateos_orense > Data

Overview Devices Payload Formats Integrations **Data** Settings

APPLICATION DATA || pause clear

Filters uplink downlink activation ack error

time	counter	port	devid	payload	analog_in_1	barometric_pressure_3	relative_humidity_4	temperature_2
▲ 20:24:27	118	1	jfmateos_orense_01	01 02 01 1C 02 67 00 D4 03 73 25 11 04 68 5D	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								
◀								
▲ 20:22:13	x	x historical	jfmateos_orense_01	01 02 01 1C 02 67 00 D4 03 73 25 11 04 68 5D	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								
▲ 20:21:07	x	x historical	jfmateos_orense_01	01 02 01 1C 02 67 00 D4 03 73 25 12 04 68 5C	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								
▲ 20:20:00	x	x historical	jfmateos_orense_01	01 02 01 1C 02 67 00 D4 03 73 25 11 04 68 5C	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								
▲ 20:18:53	x	x historical	jfmateos_orense_01	01 02 01 1C 02 67 00 D5 03 73 25 11 04 68 5C	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								
▲ 20:17:46	x	x historical	jfmateos_orense_01	01 02 01 1C 02 67 00 D5 03 73 25 11 04 68 5C	analog_in_1: 2.84	barometric_pressure_3: 948.9	relative_humidity_4: 46.5	temperature_2: 21.2
◀								

Juan Félix Mateos

LED: Downlink

```
222     lpp.addBarometricPressure(4, bme.readPressure() / 100.0F);
223     lpp.addRelativeHumidity(5, bme.readHumidity());
224     }
225     lpp.addDigitalOutput(6, 1);
226     LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(), 0);
227     Serial.println(F("Packet queued"));
228   }
229   // Next TX is scheduled after TX_COMPLETE event.
230 }
231
232 void setup() {
233   pinMode(4, OUTPUT);
234   digitalWrite(4, LOW);
235   //   pinMode(13, OUTPUT);
236   while (!Serial); // wait for Serial to be initialized
```

LED: Downlink

```
37 #include <hal/hal.h>
38 #include <Adafruit_Sensor.h>
39 #include <Adafruit_BME280.h>
40 #include <CayenneLPP.h>
41 CayenneLPP lpp(22);
42 //En el archivo Adafruit_BME280.h hay que cambiar la siguiente linea
43 //    #define BME280_ADDRESS (0x76)
```

LED: Downlink

```
152     case EV_TXCOMPLETE:
153         Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
154         if (LMIC.txrxFlags & TXRX_ACK)
155             Serial.println(F("Received ack"));
156         if (LMIC.dataLen) {
157             //Serial.println(F("Received "));
158             //Serial.println(LMIC.dataLen);
159             //Serial.println(F(" bytes of payload"));
160             //El downlink tendrá el formato canal (06), tipo de valor (00=digital), valor (0 o 100=0x64), FF (ACK)
161             if (LMIC.frame[LMIC.dataBeg + 2] == 100) {
162                 digitalWrite(4, HIGH);
163             } else {
164                 digitalWrite(4, LOW);
165             }
166         }
```

LED: Downlink

Probar a encender y apagar el LED simulando los siguientes downlinks:

- 06 01 64 → Encender
- 06 01 00 → Apagar

DOWNLINK

Scheduling

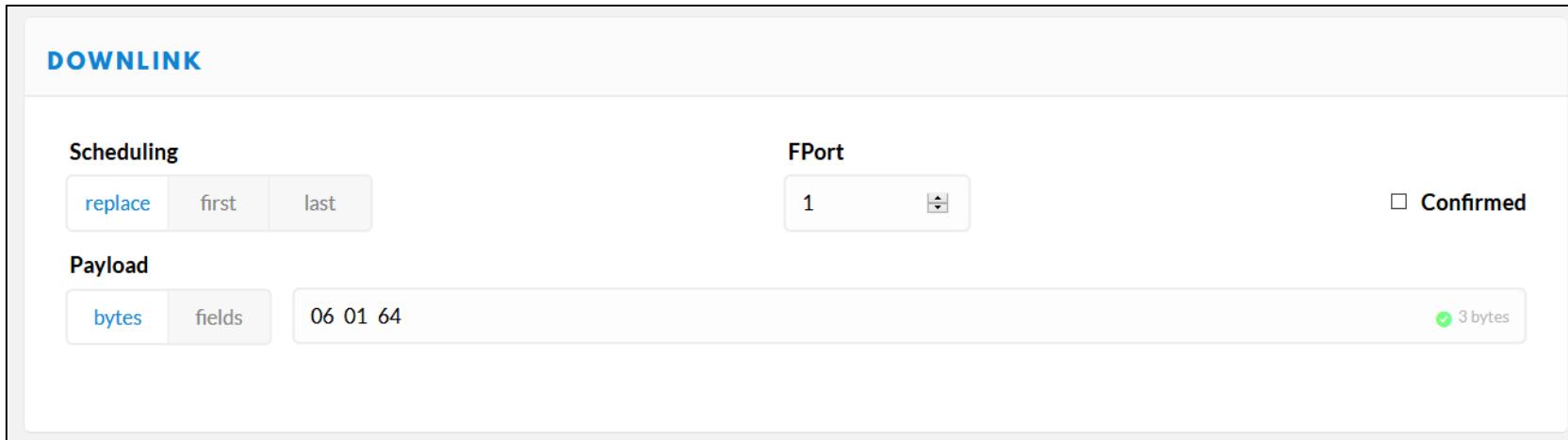
FPort

1

Confirmed

Payload

06 01 64 ✓ 3 bytes



OTAA + HALL + Interrupciones + Heartbeat

Vamos a programar el nodo para que, cada vez que se detecte un cambio en el estado del sensor HALL, envíe un mensaje y, adicionalmente, también envíe un heartbeat cada minuto.

Partimos del ejemplo MCC1 LoRaWAN Library LMIC Library:

- `ttn-otaa.ino`

OTAA + HALL + Interrupciones + Heartbeat

Dar de alta un dispositivo OTAA

The image shows two screenshots of The Things Network Console. The left screenshot displays the 'Devices' page for an application named 'jfmateos_hackmad'. A red arrow labeled '1' points to the 'Devices' tab in the navigation bar. Another red arrow labeled '2' points to the 'register device' button at the top right of the device list. The right screenshot shows the 'REGISTER DEVICE' form. A red arrow labeled '3' points to the 'Device ID' field containing 'jfmateos_hackmad_02'. A red arrow labeled '4' points to the 'Device EUI' field, which is currently empty. A red arrow labeled '5' points to the 'Register' button at the bottom right of the form.

OTAA + HALL + Interrupciones + Heartbeat

Configurar el nodo en Arduino

¡PRECAUCIÓN! El AppEUI y DeviceEUI deben ir en **little-endian** format, pero el AppKey debe ir en **big-endian** format

DEVICE OVERVIEW

Application ID jfmateos_hackmad
Device ID jfmateos_hackmad_02
Activation Method OTAA

Device EUI <=> lsb { 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }
Application EUI <=> lsb { 0xBF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }
App Key <=> msb { 0x90, 0x00, 0x00 }

```
ttn-otaa$  
49 # define FILLMEIN (#dont edit this, edit the lines that use FILLMEIN)  
50 #endif  
51  
52 // This EUI must be in little-endian format, so least-significant-byte  
53 // first. When copying an EUI from ttncctl output, this means to reverse  
54 // the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,  
55 // 0x70.  
56 static const ul_t APPKEY[8]={ 0xBF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };  
57 void os_getAppKey (ul_t* buf) { memcpy_P(buf, APPKEY, 8);}  
58  
59 // This should also be in little endian format, see above.  
60 static const ul_t DEVEUI[8]={ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };  
61 void os_getDevEui (ul_t* buf) { memcpy_P(buf, DEVEUI, 8);}  
62  
63 // This key should be in big endian format (or, since it is not really a  
64 // number but a block of memory, endianness does not really apply). In  
65 // practice, a key taken from ttncctl can be copied as-is.  
66 static const ul_t APPKEY[16] = { 0x90, 0x00, 0x00 };  
67 void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}
```

Modificar los pines que controlan el RFM95W

- Configurar la sección //Pin mapping como se indica

```
// Pin mapping  
  
const lmic_pinmap lmic_pins = {  
  
    .nss = 10,  
  
    .rxtx = LMIC_UNUSED_PIN,  
  
    .rst = 9,  
  
    .dio = {2, 7, LMIC_UNUSED_PIN},  
  
};
```

Incluir las librerías lowpower y cayenne

```
34 #include <lmic.h>
35 #include <hal/hal.h>
36 #include <SPI.h>
37 #include "LowPower.h"
38 #include <CayenneLPP.h>
39 CayenneLPP lpp(7);
```

Insertar la función readVCC

<https://gist.github.com/IoTopenTech>

```
long readVcc() {  
  
    ADMUX = _BV(REFS0) | _BV(MUX3) | _BV(MUX2) |  
    _BV(MUX1);  
  
    delay(2);  
  
    ADCSRA |= _BV(ADSC);  
  
    while (bit_is_set(ADCSRA, ADSC));  
  
    long result = ADCL;  
  
    result |= ADCH << 8;  
  
    result = 1126400L / result; // Back-calculate AVcc in mV  
  
    return result;  
  
}
```

```
86    .rxtx = LMIC_UNUSED_PIN,  
87    .rst = 9,  
88    .dio = {2, 7, LMIC_UNUSED_PIN},  
89 };  
90 long readVcc () {  
91     ADMUX = _BV (REFS0) | _BV (MUX3) | _BV (MUX2) | _BV (MUX1) ;  
92     delay (2) ;  
93     ADCSRA |= _BV (ADSC) ;  
94     while (bit_is_set (ADCSRA, ADSC)) ;  
95  
96     long result = ADCL;  
97     result |= ADCH << 8;  
98     result = 1126400L / result; // Back-calculate AVcc in mV  
99     return result;  
100 }  
101 void onEvent (ev_t ev) {  
102     Serial.print (os_getTime ()) ;  
103     Serial.print (": ") ;
```

Crear las variables necesaria para la lógica del programa

```
37 #include "LOWPOWER.h"
38 #include <CayenneLPP.h>
39 CayenneLPP lpp(7);
40
41 volatile boolean envioEnCurso = false;
42 boolean puertaAbierta;
43 volatile boolean interrumpido = false;
44 byte minutosHeartbeat = 1;
45
```

Modificar la función setup()

```
238 LMIC_reset();  
239  
240 // Start job (sending automatically starts OTAA too)  
241 LMIC_setClockError(MAX_CLOCK_ERROR * 1 / 100); //Para mejorar la recepción de los downlinks  
242 //Conecto en el pin 3 un interruptor a GND normalmente abierto  
243 pinMode(3, INPUT);  
244 delay(100);  
245 // Start job  
246 puertaAbierta = digitalRead(3);  
247 enviarMensaje();  
248 //do_send(&sendjob);  
249 }  
250
```

La ISR (Interrupt Service Routine) enviarMensaje()

```
104     result = 1126400L / result; // Back-  
105     return result;  
106 }  
107  
108 void enviarMensaje() {  
109     envioEnCurso = true;  
110     interrumpido = true;  
111     os_setCallback (&sendjob, do_send);  
112 }  
113  
114 void onEvent (ev_t ev) {  
115     Serial.print(os_getTime());  
116     Serial.print(": ");  
117 }
```

Modificar el estado EV_TXCOMPLETE

```
166 case EV_TXCOMPLETE:
167     Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
168     if (LMIC.txrxFlags & TXRX_ACK)
169         Serial.println(F("Received ack"));
170     if (LMIC.dataLen) {
171         Serial.print(F("Received "));
172         Serial.print(LMIC.dataLen);
173         Serial.println(F(" bytes of payload"));
174     }
175     // Schedule next transmission
176     //os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(TX_INTERVAL), do_send);
177     envioEnCurso = false;
178     break;
179 case EV_LOST_TSYNC:
```

Incluir la carga de pago en el método do_send

```
213 void do_send(osjob_t* j) {
214     // Check if there is not a current TX/RX job running
215     if (LMIC.opmode & OP_TXRXPEND) {
216         Serial.println(F("OP_TXRXPEND, not sending"));
217     } else {
218         // Prepare upstream data transmission at the next possible time.
219         //LMIC_setTxData2(1, mydata, sizeof(mydata) - 1, 0);
220         lpp.reset();
221         lpp.addAnalogInput(1, readVcc() / 1000.F);
222         lpp.addDigitalInput(2, puertaAbierta);
223
224         LMIC_setTxData2(1, lpp.getBuffer(), lpp.getSize(), 0);
225         Serial.println(F("Packet queued"));
226     }
227     // Next TX is scheduled after TX_COMPLETE event.
228 }
```

El método loop()

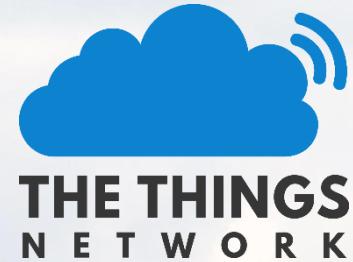
```
void loop() {
    //Esperamos a que concluya el envío
    while (envioEnCurso) {
        os_runloop_once();
    }
    delay(100);
    interrumpido = false;
```

```
    if (puertaAbierta == digitalRead(3)) {
        //La puerta no ha cambiado de estado
        attachInterrupt(digitalPinToInterrupt(3), enviarMensaje, (puertaAbierta == 1 ? FALLING : RISING));
        for (byte minutos = 0; minutos < minutosHeartbeat; minutos++) {
            if (interrumpido == true) {
                break;
            }
            for (byte contador = 0; contador < 7; contador++) {
                if (interrumpido == true) {
                    break;
                }
                LowPower.powerDown(SLEEP_8S, ADC_OFF, BOD_OFF);
            }
        }
    }
```

El método loop()

```
if (!interrumpido){  
    enviarMensaje();  
}  
else {  
    puertaAbierta = !puertaAbierta;  
}  
else {  
    //La puerta ha cambiado de estado durante el  
    envío del mensaje de estado anterior  
    puertaAbierta = !puertaAbierta;  
    enviarMensaje();  
}
```

- Si se está enviando un mensaje → Atender la máquina de estados
- Si no, comprobar si ha cambiado el estado de la puerta.
- Si sí ha cambiado, enviar mensaje.
- Si no ha cambiado, dormir en intervalos de 8 segundos mientras no se detecte un cambio de estado de la puerta o no se alcance el tiempo máximo de heartbeat.
- Si se alcanza el tiempo de heartbeat, enviarlo
- Si se detecta un cambio de estado de la puerta, salir del sueño y notificarlo.



YOU ARE THE NETWORK
LET'S BUILD THIS THING TOGETHER!

Juan Félix Mateos

juanfelixmateos@gmail.com

www.thethingsnetwork.org