

OPTIMIZACIÓN DEL PAYLOAD

MENOS ES MÁS

QUIEN SOY



Jorge :P (akirasan)

@akirasan

Maker de andar por casa, fan de Arduino, LoRaWan, RasPi, Linux, OpenSource, #Blender, #PCBArt lo que me parece en cada momento. Mantén la mente ocupada @Ripolab

© Ripollet, Barcelona, España

Twitter: @akirasan

Blog: www.akirasan.net

Instagram: @akirasan21h_

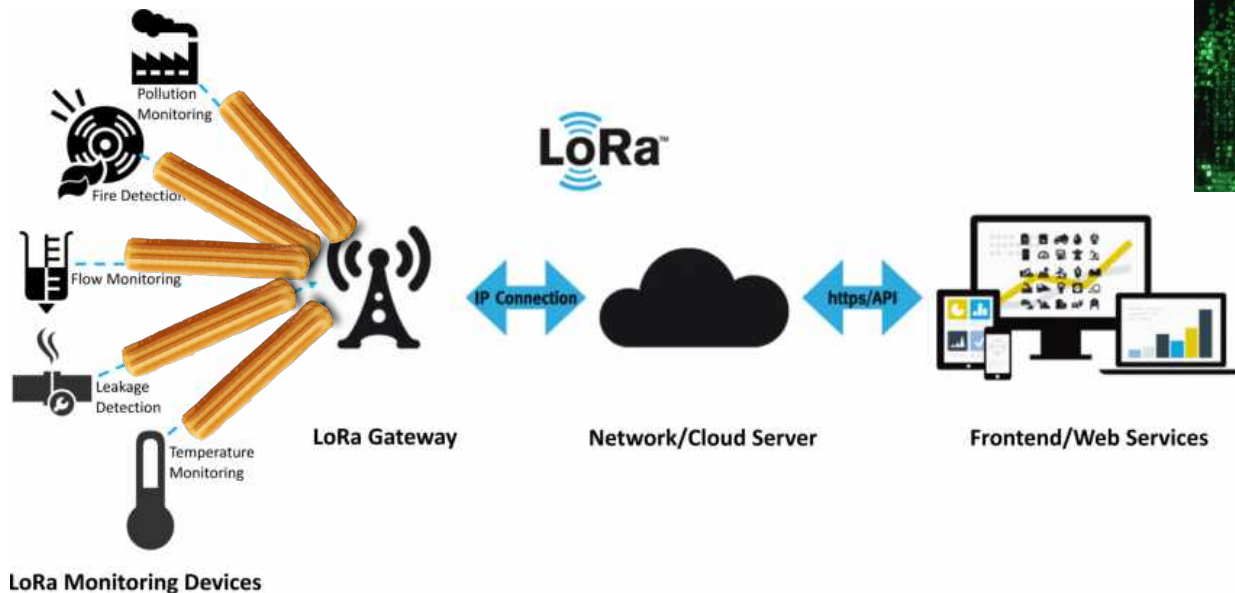
GitHub: <https://github.com/akirasan>



¿PAYLOAD?

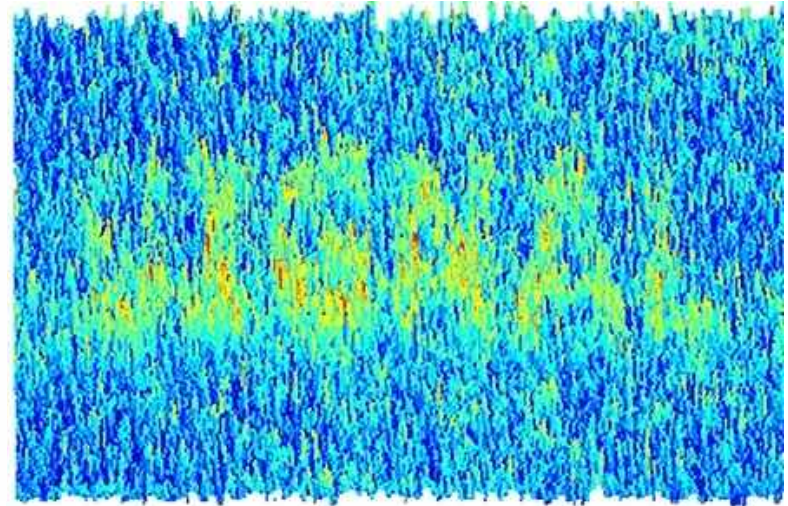
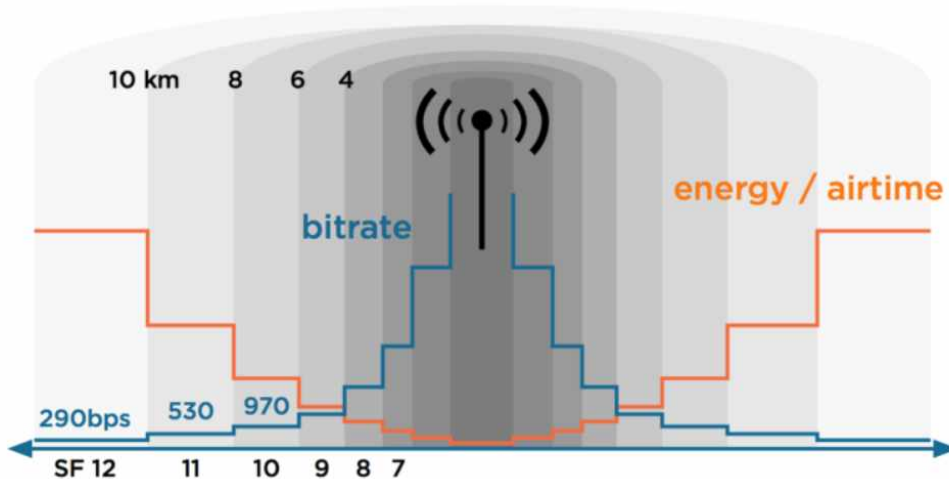
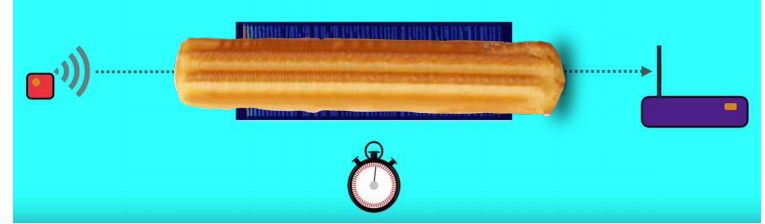
PAYLOAD

- “Churro” de bytes donde viaja la información de nuestro nodo LoRaWAN (Array de bytes).
- Algunos lo llaman Carga Útil.
- En TTN se representa en hexadecimal (HEX)

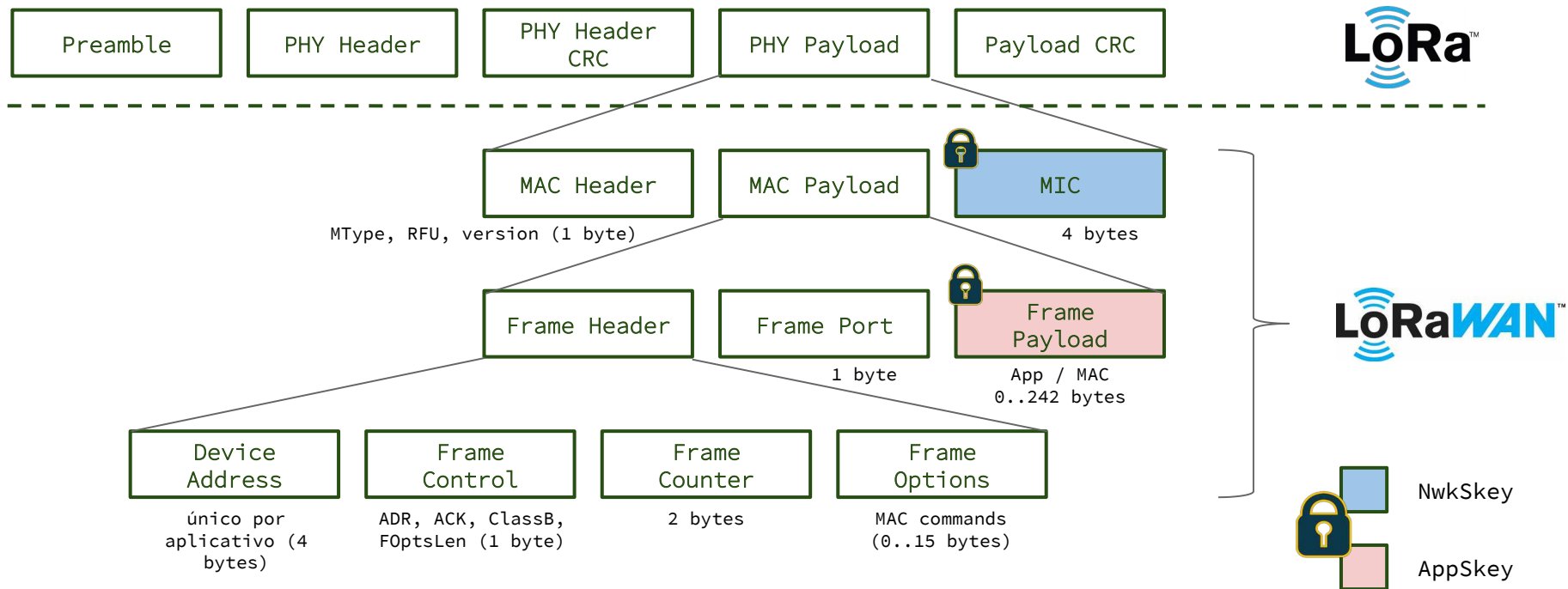


¿POR QUÉ HAY QUE OPTIMIZAR EL PAYLOAD?

- Reducir *Time on Air*.
- *Fair Play* con el resto de usuarios
- Conseguir enviar más paquetes.
- Menos consumo de energía.

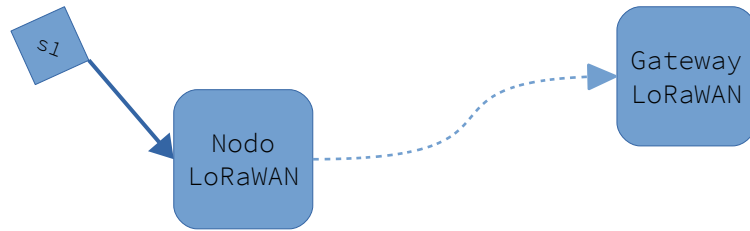


FORMATO MENSAJE LORAWAN

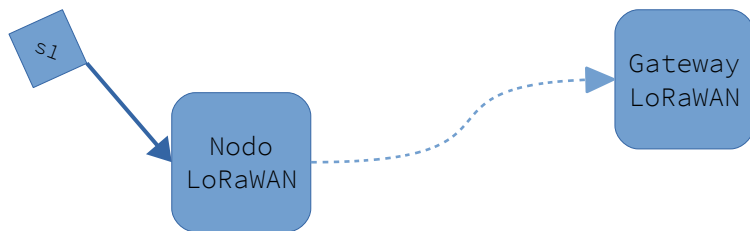


MENOS ES MAS...

¿CÓMO REDUCIR PAYLOAD?



¿CÓMO REDUCIR PAYLOAD?

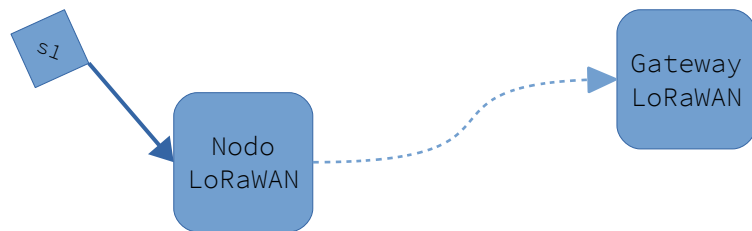


| | | | | | | | | | | | | | | | | | | | | | |
|------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| dato | e | n | c | e | n | d | e | r | | | | | | | | | | | | | |

Protocolo LoRaWAN

21 bytes: 8 bytes mensaje + 13 bytes
Protocolo LoRaWAN

¿CÓMO REDUCIR PAYLOAD?



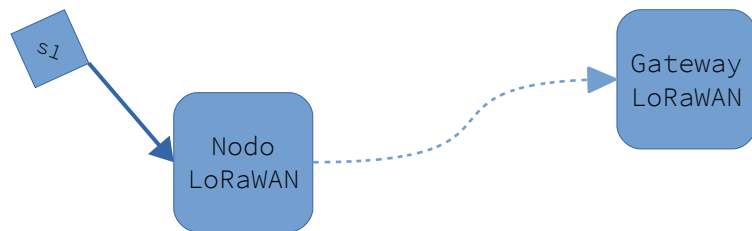
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|------|---|---|---|---|---|---|---|---|-------------------|---|----|----|----|----|----|----|----|----|----|----|----|
| dato | e | n | c | e | n | d | e | r | Protocolo LoRaWAN | | | | | | | | | | | | |

21 bytes: 8 bytes mensaje + 13 bytes Protocolo LoRaWAN

| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|------|---|---|-------------------|---|---|---|---|---|---|---|----|----|----|----|----|
| dato | o | n | Protocolo LoRaWAN | | | | | | | | | | | | |

15 bytes: 2 bytes mensaje + 13 bytes Protocolo LoRaWAN

¿CÓMO REDUCIR PAYLOAD?



| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|------|---|---|---|---|---|---|---|---|-------------------|---|----|----|----|----|----|----|----|----|----|----|----|
| dato | e | n | c | e | n | d | e | r | Protocolo LoRaWAN | | | | | | | | | | | | |

21 bytes: 8 bytes mensaje + 13 bytes Protocolo LoRaWAN

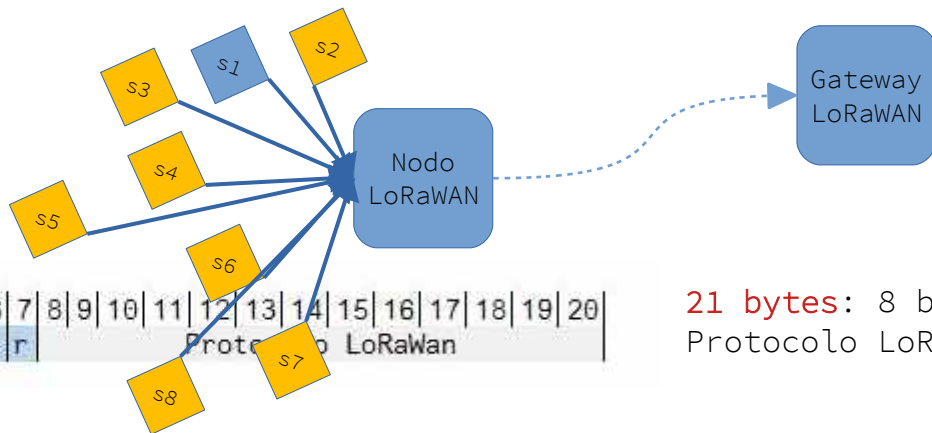
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|------|---|---|-------------------|---|---|---|---|---|---|---|----|----|----|----|----|
| dato | o | n | Protocolo LoRaWAN | | | | | | | | | | | | |

15 bytes: 2 bytes mensaje + 13 bytes Protocolo LoRaWAN

| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|------|---|-------------------|---|---|---|---|---|---|---|---|----|----|----|----|
| dato | 1 | Protocolo LoRaWAN | | | | | | | | | | | | |

14 bytes: 1 bytes mensaje + 13 bytes Protocolo LoRaWAN

¿CÓMO REDUCIR PAYLOAD? WTF!!!



21 bytes: 8 bytes mensaje + 13 bytes Protocolo LoRaWAN

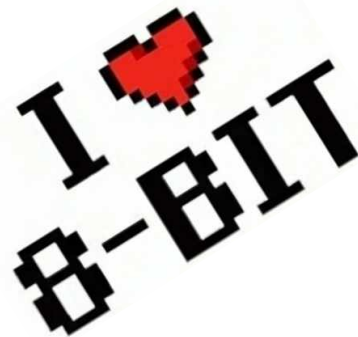
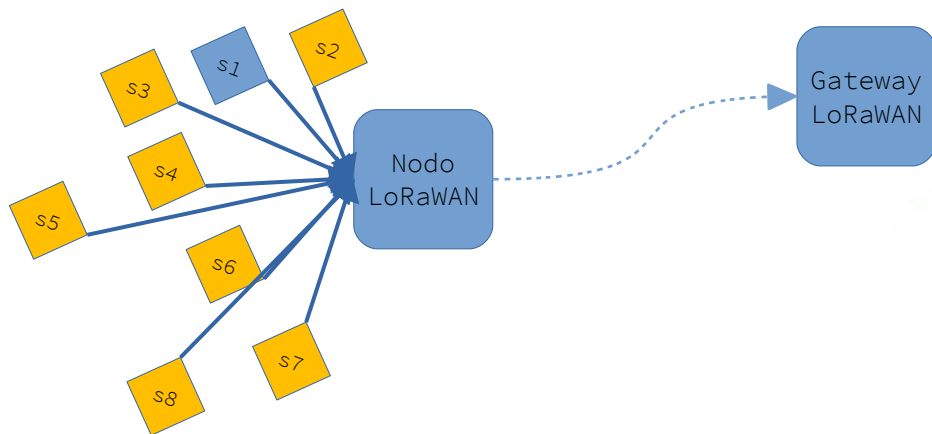
byte dato | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| o | n | | | | | | | | | | | | | |
Protocolo LoRaWAN

15 bytes: 2 bytes mensaje + 13 bytes Protocolo LoRaWAN

byte dato | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| 1 | | | | | | | | | | | | | |
Protocolo LoRaWAN

14 bytes: 1 bytes mensaje + 13 bytes Protocolo LoRaWAN

¿CÓMO REDUCIR PAYLOAD? PENSANDO...



| | | | | | | | | | | | | | | |
|------|----|---|---|---|---|---|---|---|---|---|----|----|----|----|
| byte | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
| dato | 78 | | | | | | | | | | | | | |

Protocolo LoRaWAN

14 bytes: 1 bytes mensaje + 13 bytes
Protocolo LoRaWAN

| | | | | | | | | | |
|-------|---|---|---|---|---|---|---|---|------|
| bit | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| valor | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | = 78 |

s1 s2 s3 s4 s5 s6 s7 s8

Trabajar a nivel de BITS

IMPACTO EN NUESTRO TIME-ON-AIR

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 |

10 bytes vs SF vs Time-On-Air

<https://www.loratools.nl/#/>

FORMATOS DE PAYLOAD EN THE THINGS NETWORK

TIPOS DE FORMATOS

- Cayenne LPP
- Custom

PAYLOAD FORMATS

Payload Format

The payload format sent by your devices

Cayenne LPP

Custom

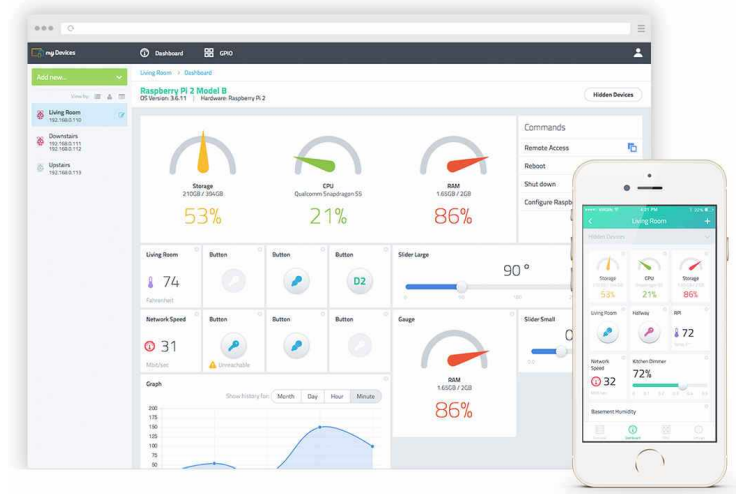
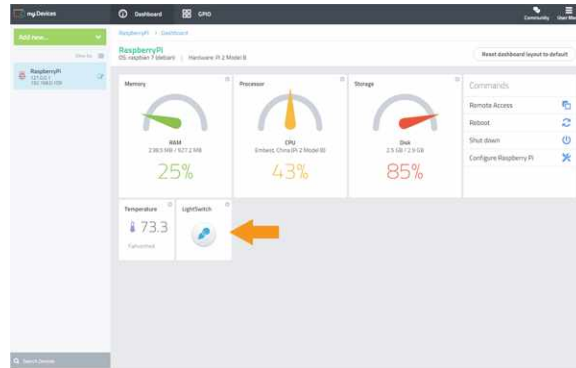
Cayenne LPP

Cancel

no changes to save

CAYENNE LPP (LOW POWER PAYLOAD)

- Plataforma IoT myDevices
- Fácil y cómodo
- Diferentes datos de sensores en un único mensaje
- Cada dato se identifica con una cabecera:
 - Data Channel: Identificación única del sensor
 - Data Type: Tipo de dato
- Dashboards



CAYENNE LPP (LOW POWER PAYLOAD)

API

Propósito general

```
addDigitalInput(uint8_t channel, uint8_t value);  
addDigitalOutput(uint8_t channel, uint8_t value);
```

```
addAnalogInput(uint8_t channel, float value);  
addAnalogOutput(uint8_t channel, float value);
```

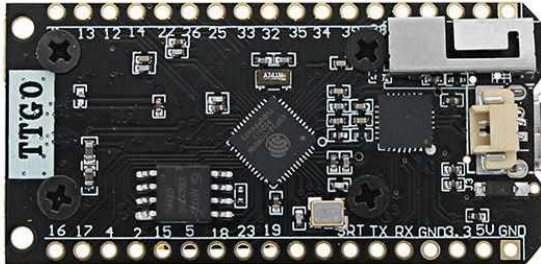
Específicos

```
addLuminosity(uint8_t channel, uint16_t lux);  
addPresence(uint8_t channel, uint8_t value);  
addTemperature(uint8_t channel, float celsius);  
addRelativeHumidity(uint8_t channel, float rh);  
addAccelerometer(uint8_t channel, float x, float y, float z);  
addBarometricPressure(uint8_t channel, float hpa);  
addGyrometer(uint8_t channel, float x, float y, float z);  
addGPS(uint8_t channel, float latitude, float longitude, float meters);
```

CAYENNE LPP (LOW POWER PAYLOAD)



EJEMPLO Cayenne



TTGO LORA32 868/915Mhz SX1276 ESP32

Operating voltage: 3.3V to 7V

Operating temperature range: -40 ° C to + 90 ° C

Support for Sniffer software protocol analysis, Station, SoftAP, Bluetooth and Wi-Fi Direct modes

Data rates: 150 Mbps ' 11n HT40., 72 Mbps ' 11n HT20, 54 Mbps ' 11g, 11 Mbps ' 11b

transmit power: 19.5 dBm ' 11b, 16.5 dBm ' 11g, 15.5 dBm ' 11n

receiver sensitivity up to -98 dBm

UDP sustained throughput of 135 Mbps

CUSTOM

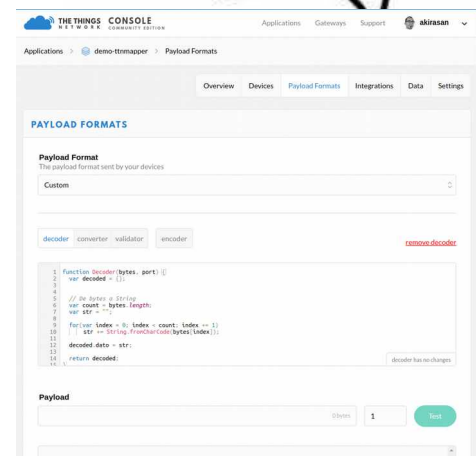
Yo me lo guiso, yo me lo como



Codificar: Definimos qué y cómo enviamos nuestros “churros” de bytes

```
/* Editor: Programa: Herencia: Ayuda
read_tty_herando  LMC_Accesorio | LMC_Accesorio | LMC_Accesorio
const unsigned TX_INTERVAL = 60;

/* RQII no buques 000 ***** */
void enviar_datos(job_t *) {
    // Check if there is not a current TX/RX job running
    if (LMC_opmode < 0 || TXRX_BUSY) {
        Serial.println(F("TX/RX busy, not sending"));
        return;
    }
    // Preparamos datos para enviar PAYLOAD
    // *****
    Serial.println("enviando...");
    Serial.println(F("n -> Tamaño de nuestro payload: ")); Serial.println(sizeof(payload) - 1); Serial.println(F("n -> payload HEX: ")); for (byte i = 0; i < sizeof(payload) - 1; i++) {
        Serial.print(payload[i], HEX);
        Serial.print(" ");
    }
    Serial.println(F("n -> payload HEX: ")); for (byte i = 0; i < sizeof(payload) - 1; i++) {
        Serial.print(payload[i], HEX);
        Serial.print(" ");
    }
    Serial.println(F("n -> payload BIN: ")); for (byte i = 0; i < sizeof(payload) - 1; i++) {
        Serial.print(payload[i], BIN);
        Serial.print(" ");
    }
    LMC_sendData2(1, payload, sizeof(payload) - 1, 0);
    Serial.println(F("n -> Paquete listo para enviar"));
}
}
```



Decodificar: Damos sentido a los “churros” de bytes

CUSTOM



EJEMPLO “hola mundo”

BYTE A BYTE... BIT A BIT

LOS TÍPICOS TIPOS DE DATOS ("TEÓRICOS")

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

ENVIAR NÚMEROS GRANDES

Técnica de la Indexación (enviar la variación)

Podemos tener un número grande, pero si la variación que sufre es pequeña, podemos enviar únicamente esa variación.

Ejemplo: Valores entre 3400 y 3600

- Sin indexar 2 bytes
- Con indexación 1 byte

Código de encode

```
int myVal = 3450;  
const int myBase = 3400;  
byte payload[] = { myVal - myBase };
```



Código de decode

```
var myBase = 3400;  
decoded.myVal = bytes[0] + myBase;
```

1. Restamos valor leído con valor mínimo conocido.
2. Enviamos el resultado de la resta.

1. Al valor recibido le sumamos el valor mínimo conocido

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

ENVIAR NÚMEROS GRANDES

Técnica del Redondeo

Si podemos permitirnos cierto margen de error, podríamos enviar nuestro valor con una aproximación válida.

Ejemplo: Valor a enviar 300

- Sin redondeo 2 bytes
- Con redondeo 1 byte (con margen de error)

Código de encode

```
int myVal = 300;  
int errorMargin = 2;  
byte payload[] = { round(myVal / errorMargin) };
```

1. Dividimos el valor por 2.
2. Enviamos parte entera división.

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

Código de decode

```
var errorMargin = 2;  
decoded.myVal = bytes[0] * errorMargin;
```

1. Al valor recibido lo multiplicamos por 2.

ENVIAR NÚMEROS GRANDES +1 BYTE

Números de 2 bytes (0..65.535) o mas!!!

Tenemos un valor entre 0 y 65.535 que internamente se guarda en 2 bytes. Tenemos que partirlo para enviarlo en bytes individuales.

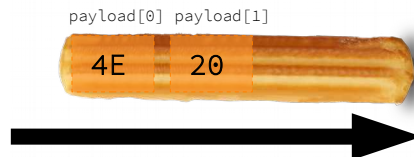
Ejemplo: Valor a enviar 20.000 → HEX 4E 20 → BIN 01001110 00100000

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

Código de encode

```
int myVal = 20000;  
byte payload[];  
payload[0] = highByte(myVal);  
payload[1] = lowByte(myVal);
```

```
var myVal = 20000;  
var bytes = [];  
bytes[0] = (myVal & 0xFF00) >> 8;  
bytes[1] = (myVal & 0x00FF);
```



Código de decode

```
decoded.myVal = (bytes[0] << 8)  
+ bytes[1];
```

ENVIAR NÚMEROS GRANDES +1 BYTE

Números mas grandes...de 4 bytes

Tenemos un valor entre 0 y 4.294.967.295 que internamente se guarda en 4 bytes. Tenemos que partirlo para enviarlo en bytes individuales.

Ejemplo: Valor a enviar 200.000

Código de encode

```
long lng = 200000L;
byte payload[4];
payload[0] = (byte) ((lng & 0xFF000000) >> 24);
payload[1] = (byte) ((lng & 0x00FF0000) >> 16);
payload[2] = (byte) ((lng & 0x0000FF00) >> 8);
payload[3] = (byte) ((lng & 0x000000FF));
```

1. Desplazamiento de bits a la derecha.
2. Asignamos 1 byte a una posición del array de bytes.

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

Código de decode

```
decoded.myVal = ((long)(bytes[0]) << 24)
+ ((long)(bytes[1]) << 16)
+ ((long)(bytes[2]) << 8)
+ ((long)(bytes[3]));
```

1. Sumamos cada posición del payload, desplazando a la izquierda el byte a su posición.

ENVIAR NÚMEROS DECIMALES

Números float

Un número decimal normalmente se guarda en 4 bytes. Si es posible, podemos reducir su tamaño en el momento del envío.

Ejemplo: Valor a enviar 1,22

- Sin conversión 4 bytes
- Con conversión 1 byte

Código de encode

```
float myVal = 1.22;  
byte payload[4];  
payload[0] = round(myVal * 100);
```



Código de decode

```
decoded.myVal = bytes[0] / 100;
```

1. Eliminamos los decimales **multiplicando** por potencia de 10.
2. Redondeamos el valor (quitamos decimales 0)

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

1. Dividimos por la misma potencia que en origen.

ENVIAR NÚMEROS NEGATIVOS

La diferencia es 1 bit

La única diferencia con la representación de los tipos de datos sin signo, es el primer bit (el de mas a la izquierda).

Aplicamos las mismas estrategias que en resto de tipos.

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

CUSTOM

EJEMPLO “números negativos”



UNA HISTORIA BASADA EN
HECHO REALES...

TTN MAPPER: PROYECTO COLABORATIVO

Objetivo: Generar un mapa global de cobertura de LoRaWan para The Things Network

- Datos abiertos (Open Data Commons)
- OpenSource (pendiente...)



URL: <https://ttnmapper.org>

Twitter: @ttnmapper

Github: <https://github.com/ttnmapper>



JP Meijers
@jpmeijers

¿CÓMO PODEMOS CONTRIBUIR?

- **Método 1.** Usando un nodo+aplicación TTN y una aplicación para móvil
- **Método 2.** Usando un nodo con GPS y una aplicación conectada a TTNMapper

NODO + APP TTN + APP MÓVIL

- Método fácil de implementar
- Aplicación disponible para Android/iOS
- Siempre juntos: Nodo + Aplicación TTN
- Vincular la app móvil con Aplicación TTN
- No es necesario enviar datos, solo un paquete

App Store Preview

This app is only available on the App Store for iOS devices.



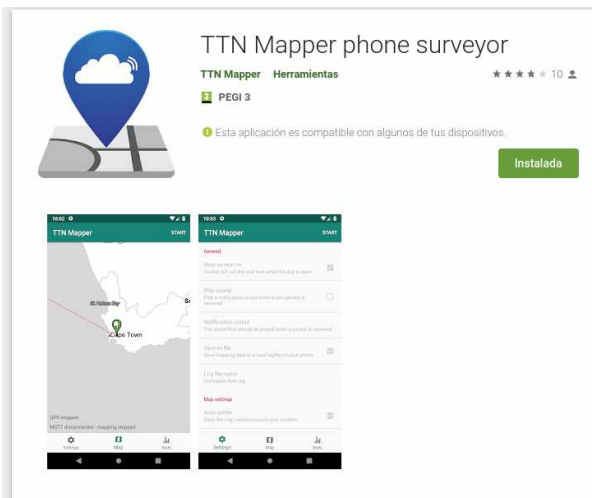
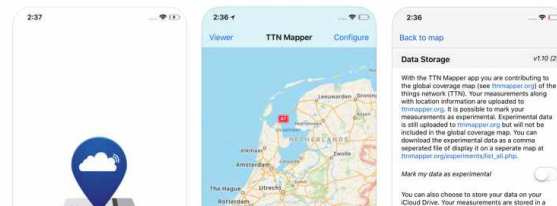
TTN Mapper

Timothy Sealy

★★★★ 1.0, 1 Rating

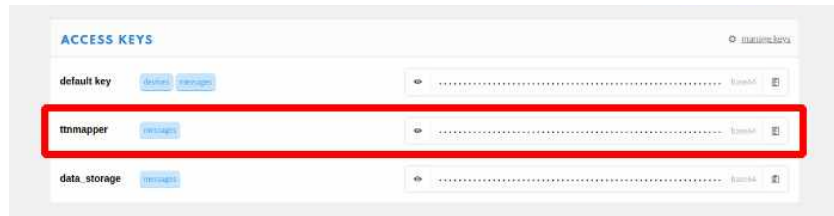
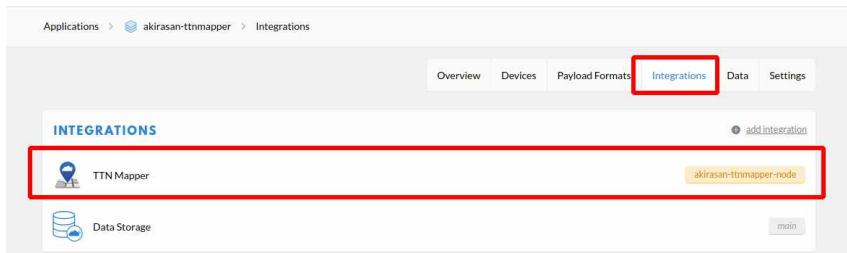
Free

iPhone Screenshots



NODO CON GPS + APP TTN + INTEGRACIÓN TTNMAPPER

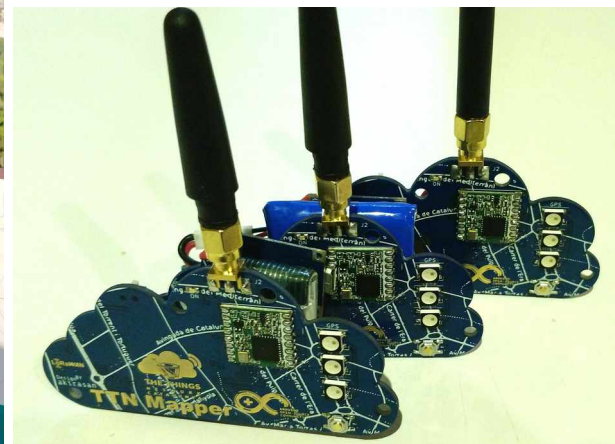
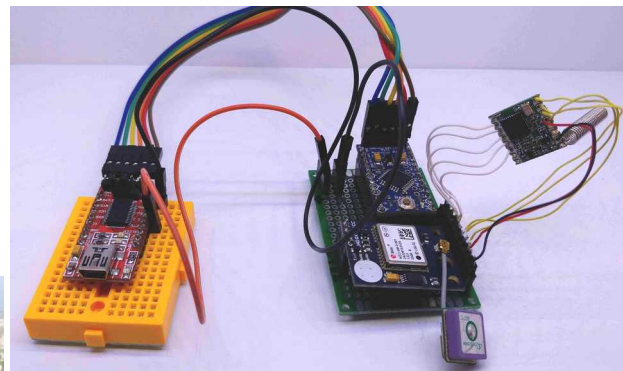
- Nodo con GPS integrado
- Aplicación integrada con TTNMapper
- Gestionar información GPS:
 - Objeto **JSON** con las keys:
 - "latitude"
 - "longitude"
 - "altitude"
 - "hdop" o "accuracy" o "sats" (calidad de los datos)
- Enviar en formato Cayenne LPP o Custom



PROYECTO NUBE TTNMAPPER

PROYECTO NUBE TTNMAPPER

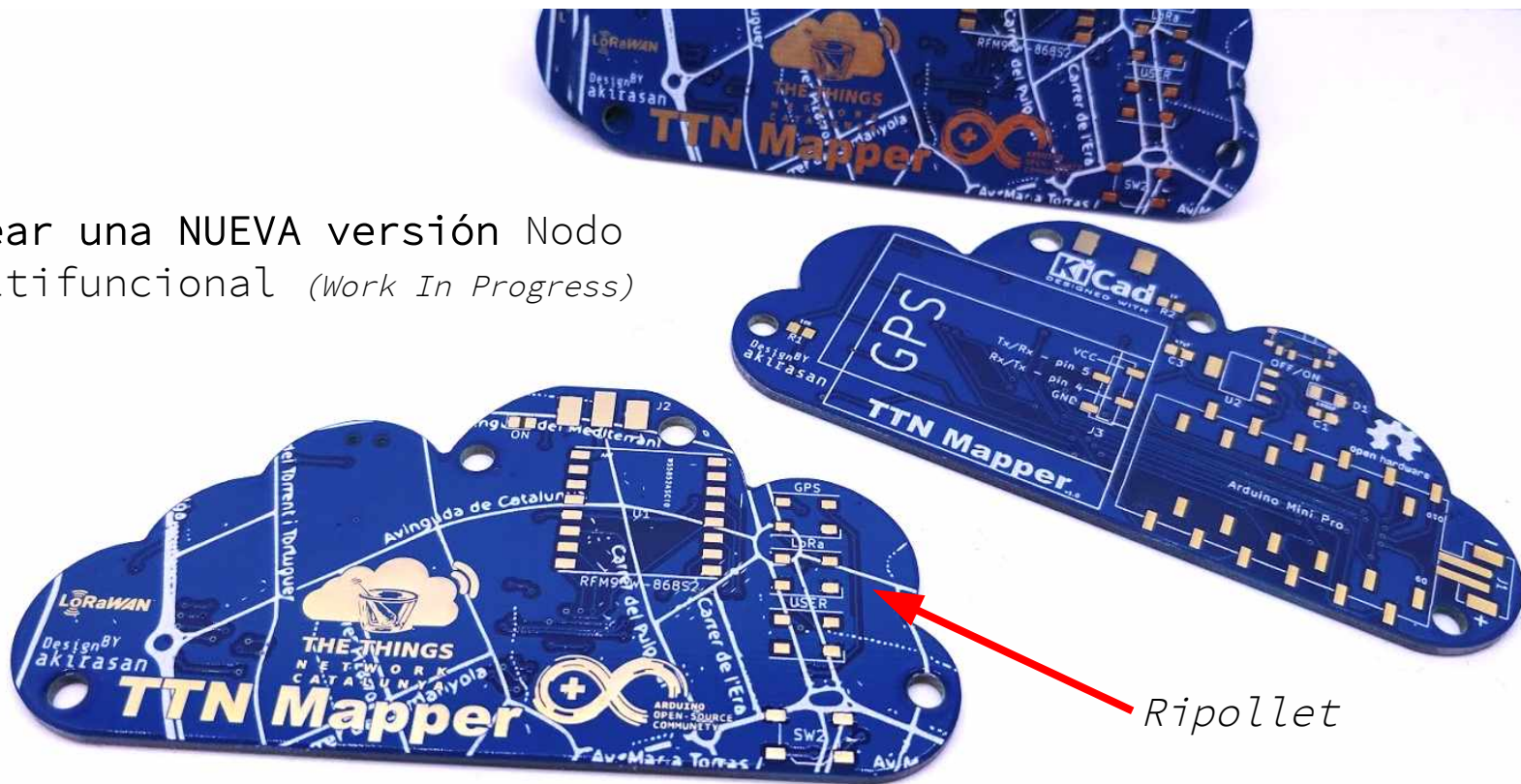
- Nodo TTN con GPS
 - Aplicación en TTN integrada con TTNMapper
 - Gestionar información GPS
 - 100% Opensource
-
- Diseño con Inkscape y KiCAD
 - Arduino Mini Pro 3.3V
 - Módulo GPS Ublox NEO-6M
 - Módulo RFM95W 868Mhz



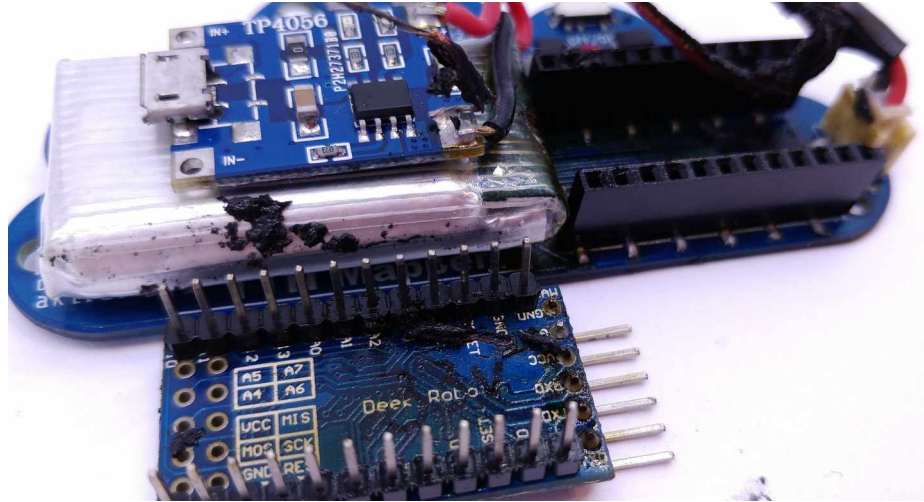
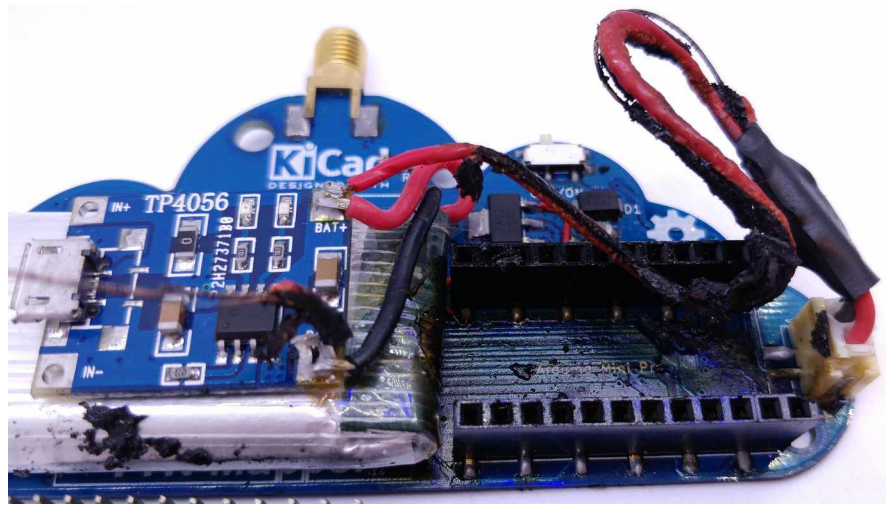
<https://github.com/akirasan/LoRaWAN-tracker-TTNmapper>

PROYECTO NUBE TTNMAPPER

Crear una NUEVA versión Nodo multifuncional *(Work In Progress)*



PROYECTO NUBE TTNMAPPER

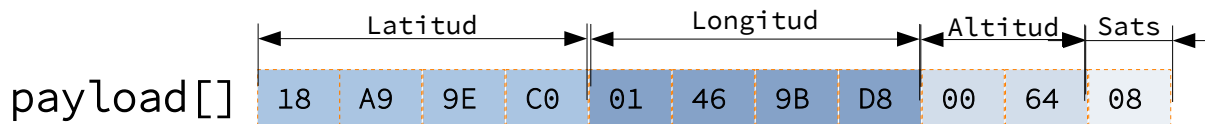


¿CÓMO ENVIAMOS DATOS
COMPLEJOS DE UN GPS?

FORMANDO UN CHURRO DE DATOS

Enviar múltiples datos en el mismo payload:

- *Latitud*
- *Longitud*
- *Altitud*
- *"hdop"* o *"accuracy"* o *"sats"*



TRABAJO CON BITS – TAMAÑO DE LOS DATOS

Codificar datos de un GPS

La *Longitud* tiene un valor entre -180° y 180° con una precisión de 7 decimales.

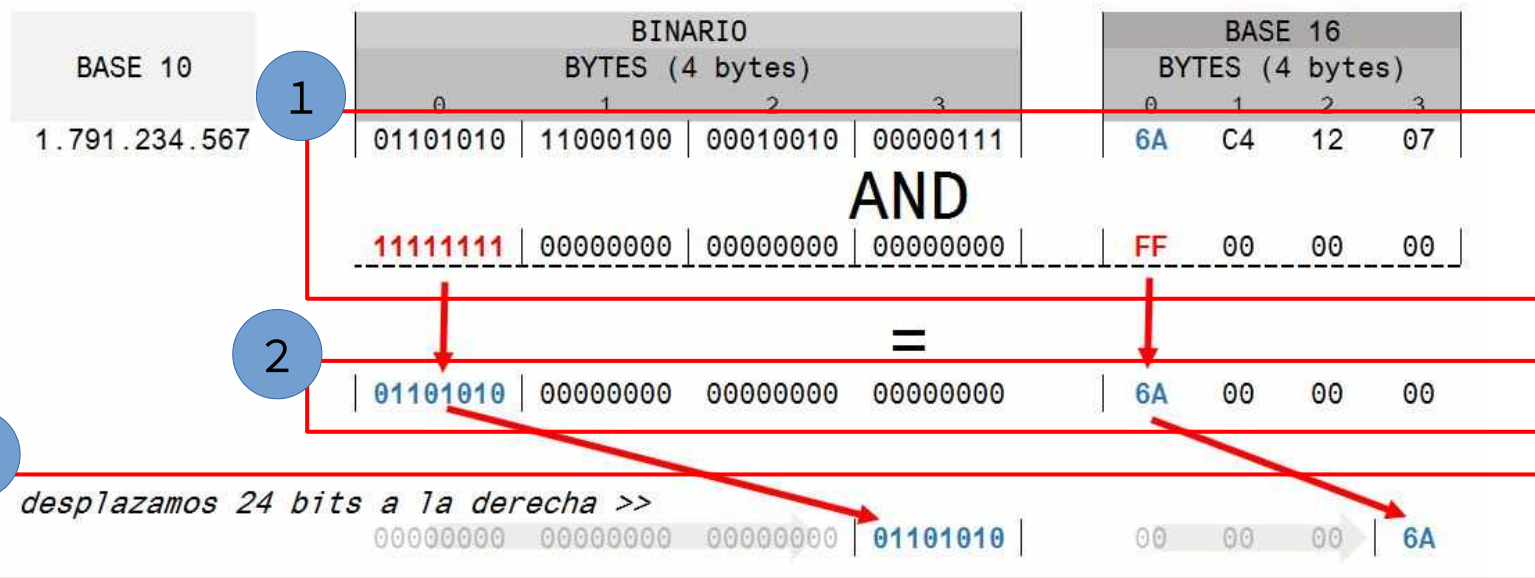
Ejemplo: 179,1234567

Eliminando decimales = 1.791.234.567

| TIPO DE DATO | TAMAÑO EN MEMORIA | RANGO DE VALORES |
|------------------|-------------------|---|
| byte | 1 byte / 8 bits | 0..255 |
| char (con signo) | 1 byte / 8 bits | (7 bits) -128..127 |
| word | 2 bytes / 16 bits | 0.. 65.535 |
| int (con signo) | 2 bytes / 16 bits | (15 bits) -32.768.. 32.767 |
| unsignedlong | 4 bytes / 32 bits | 0.. 4.294.967.295 |
| long | 4 bytes / 32 bits | (31 bits) -2,147,483,648..2,147,483,647 |

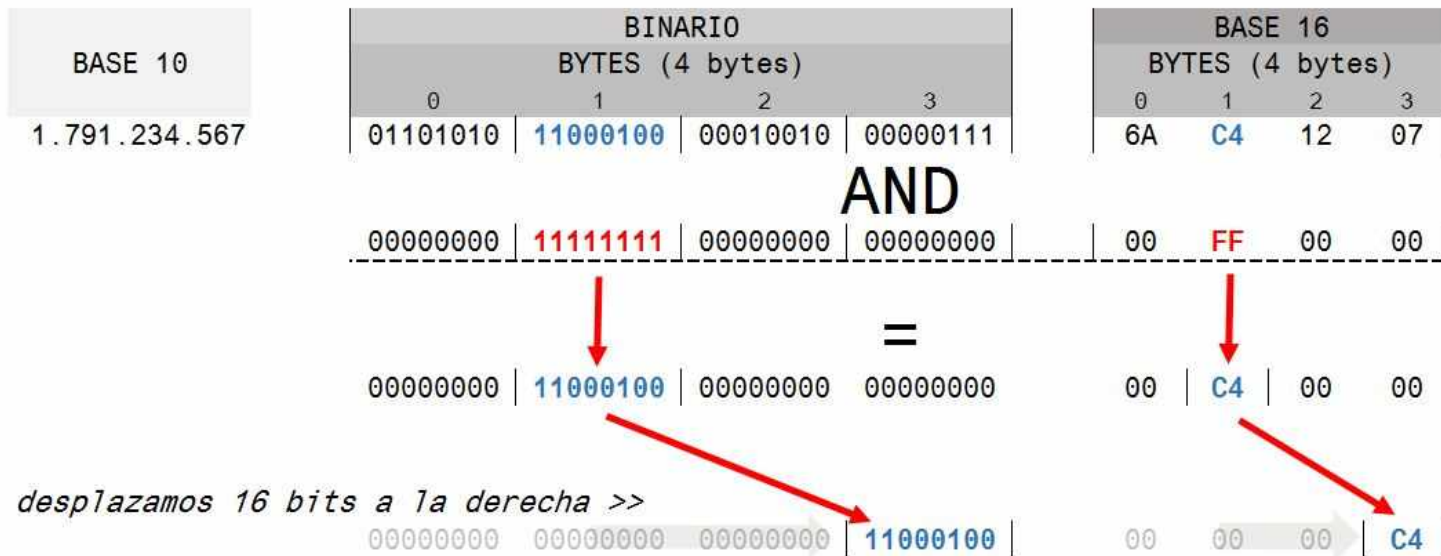
| BASE 10 | BINARIO (32 bits) | BYTES (4 bytes) | | | |
|----------------|-----------------------------------|-----------------|----------|----------|----------|
| | | 0 | 1 | 2 | 3 |
| 1.791.234.567 | 011010101100010000001001000000111 | 01101010 | 11000100 | 00010010 | 00000111 |
| -1.791.234.567 | 111010101100010000001001000000111 | 11101010 | 11000100 | 00010010 | 00000111 |

TRABAJO CON BITS – TROCEANDO BYTE A BYTE



| BASE 10 | BINARIO (1 byte) | BASE 16 (1 byte) | payload |
|---------|------------------|------------------|---------|
| 106 | 01101010 | 6A | 0 |

TRABAJO CON BITS – TROCEANDO BYTE A BYTE



| BASE 10 | BINARIO (1 byte) | BASE 16 (1 byte) |
|---------|------------------|------------------|
| 196 | 11000100 | C4 |

| payload | |
|---------|----|
| 0 | 1 |
| 6A | C4 |

TRABAJO CON BITS – TROCEANDO BYTE A BYTE



```
static uint8_t payload[4];  
float f_longitud = 179.1234567;
```

```
long longitud = f_longitud * 10000000;
```

```
Serial.println(longitud);
```

```
// [0..3] 4 bytes
```

```
payload[0] = (byte) ((longitud & 0xFF000000) >> 24 );
```

```
payload[1] = (byte) ((longitud & 0x00FF0000) >> 16 );
```

```
payload[2] = (byte) ((longitud & 0x0000FF00) >> 8 );
```

```
payload[3] = (byte) ((longitud & 0X000000FF));
```

```
// payload = 6AC41207
```



```
1 function Decoder(bytes, port) {  
2   // Decode an uplink message from a buffer  
3   // (array) of bytes to an object of fields.  
4   var decoded = {};  
5   longitud_decode = ((bytes[0]) << 24)  
6   + ((bytes[1]) << 16)  
7   + ((bytes[2]) << 8)  
8   + ((bytes[3]));  
9   decoded.longitud_gps = longitud_decode / 10000000;  
10  return decoded;  
11 }
```

| payload | | | |
|---------|----|----|----|
| 0 | 1 | 2 | 3 |
| 6A | C4 | 12 | 07 |

CUSTOM



EJEMPLO GPS

¿Y AHORA QUE?

REFERENCIAS JAVASCRIPT



THE THINGS
NETWORK

CONSOLE
COMMUNITY EDITION



JavaScript



DARIUS FOROUX

Operator precedence

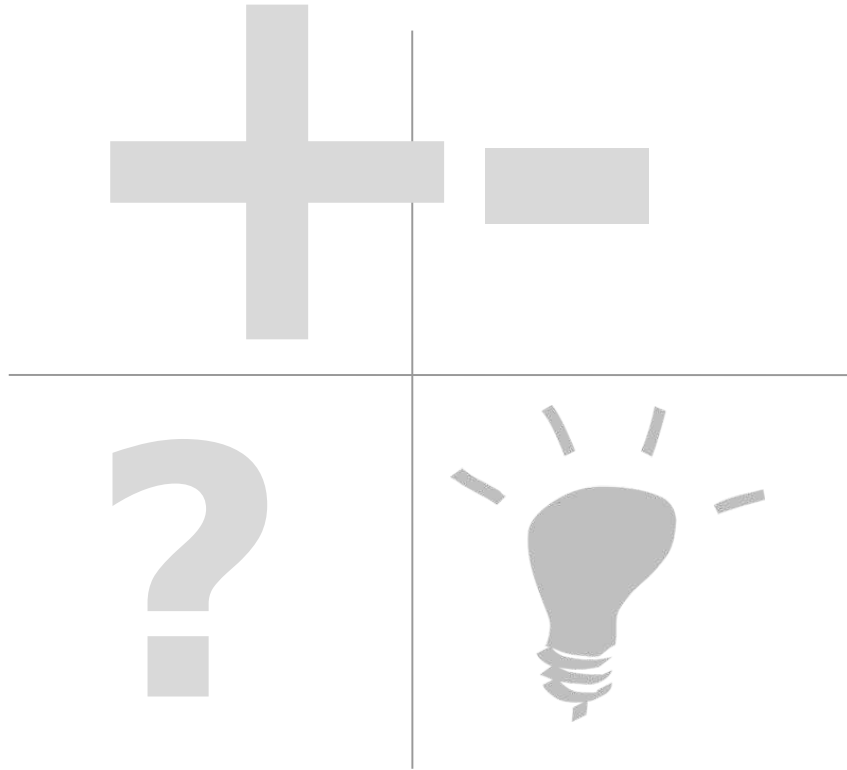
| Level | Operators | Notes |
|-------|--|--|
| 1 | () [] . | call, member (including typeof and void) |
| 2 | ! - - ++ -- | negation, increment |
| 3 | * / % | multiply/divide |
| 4 | + - | addition/subtraction |
| 5 | << >> >>> | bitwise shift |
| 6 | < <= > >= | relational |
| 7 | == != | equality |
| 8 | & | bitwise AND |
| 9 | ^ | bitwise XOR |
| 10 | | bitwise OR |
| 11 | && | logical AND |
| 12 | | logical OR |
| 13 | ?: | conditional |
| 14 | = += -= *= /= %= <<= >>= >>>= &= ^= = | assignment |
| 15 | , | comma |

RECETA PARA UN BUEN "CHURRO DE BYTES"

- Inventarnos un código basado en bits
- En mente el *Time on Air* y *Spreading Factor*
- "Texto" PROHIBIDO
- Descomponer en bytes: números grandes, negativos, decimales,...
- Enriquecer la información en la aplicación:
decoder(), converter(), validator()



GRACIAS



@akirasan
<http://akirasan.net>

