

## INFORME DHT22

PROMPT PARA OBTENER DEL DOCUMENTO EJEMPLO, OTRO PROMPT, QUE PERMITA ADAPTAR DE DICHO EJEMPLO CON DOS DS18B20 A OTRO SISTEMA CON UN DHT22. SE ADJUNTA ARCHIVO EN PDF DEL PROYECTO CON 2 DS18B20:

**“Describe el proyecto tan breve como para que sirva para un prompt, en el cual se va a solicitar todos los códigos e indicaciones que permitan, en lugar de los dos ds18b20, conectar un dht22”**

RESPUESTA DE CLAUDE (PROMPT A ESCRIBIR EN UN NUEVO CHAT, ADJUNTADO EL DOC. EJEMPLO):

**Proyecto base:** Sistema de telemetría LoRaWAN con TTGO LoRa32 v1 que actualmente monitorea temperatura con 2 sensores DS18B20 conectados en paralelo al GPIO 21, transmitiendo cada 10 segundos a The Things Network (TTN) aplicación "2xds18b20", con visualización en Node-RED mediante 2 gauges independientes.

**Arquitectura actual:**

- Hardware: TTGO LoRa32-OLED v1 (ESP32, SX1276, OLED 128x64)
- Sensores: 2x DS18B20 en bus 1-Wire (GPIO 21, pull-up 4.7kΩ)
- Red: LoRaWAN US915 sub-band 2, SF7, OTAA
- Payload: 6 bytes [Temp1\_MSB, Temp1\_LSB, Temp2\_MSB, Temp2\_LSB, Status, Count]
- TTN: Decodificador JavaScript que extrae ambas temperaturas
- Visualización: Node-RED con 2 gauges (Sensor 1-Interno, Sensor 2-Externo)
- Features: Watchdog 30s, display OLED, timing preciso 10s

**Objetivo de conversión:** Reemplazar los 2 sensores DS18B20 por 1 sensor DHT22 que mida temperatura Y humedad, manteniendo la misma arquitectura LoRaWAN, TTN y Node-RED.

**Modificaciones necesarias:**

1. Código ESP32 con librería DHT22 (temperatura + humedad)
2. Nuevo formato de payload para temperatura Y humedad
3. Payload decoder JavaScript actualizado en TTN
4. Node-RED flow modificado para 2 gauges (temperatura y humedad)
5. Configuración de pin GPIO y display OLED ajustados

**EN OTRO CHAT SE ESCRIBE EL ANTERIOR PROMPT Y SE TIENE LAS SIGUIENTES CONSULTAS POR PARTE DE LA IA:**

Perfecto, veo que tienes un sistema completo y bien documentado. Antes de generar el código adaptado para el DHT22, necesito aclarar algunos detalles técnicos importantes:  
**Preguntas clave para la conversión:**

**1. GPIO del DHT22:**

- ¿En qué pin GPIO conectarás el DHT22? (Recomendaría GPIO 21 que ya tienes configurado, o prefieres otro pin)

**2. Estructura del payload:**

- ¿Quieres mantener 6 bytes o puedo optimizar?
- Propuesta: 6 bytes [Temp\_MSB, Temp\_LSB, Hum\_MSB, Hum\_LSB, Status, SensorType]
- Temperatura: int16\_t × 100 (igual que antes)
- Humedad: uint16\_t × 100 (ejemplo: 65.5% → 6550)

**3. Node-RED Dashboard:**

- Para el gauge de temperatura: ¿mismo rango 0-40°C?
- Para el gauge de humedad: ¿rango 0-100% está bien?
- ¿Quieres colores específicos para el gauge de humedad?

**4. TTN:**

- ¿Quieres usar la misma aplicación "2xds18b20" o crear una nueva?

- ¿Mantienes el mismo device o registras uno nuevo?

##### 5. Librería DHT:

- ¿Prefieres "DHT sensor library" de Adafruit (la más estable) o tienes otra preferencia?

Una vez confirmes estos detalles, te generaré:

- Código ESP32 completo con DHT22
- Payload decoder JavaScript para TTN
- Flow de Node-RED con 2 gauges
- Documentación de la conversion

#### Respuestas:

1. El mismo pi GPIO 21
2. Tu propuesta de 6 bytes
3. Si, vos seleccionas los colores adecuados
4. Nombre de la aplicación: dht22ap. Nombre del dispositivo final: dht22ed
5. La librería de Adafruit

## 1. PROGRAMA PARA ENCONTRAR DEVEUI

```
#include <Arduino.h>

// Variables globales para almacenar el DevEUI
uint8_t DEVEUI[8];

void setup() {
    Serial.begin(115200);
    delay(2000); // Esperar a que se abra el Serial Monitor

    Serial.println("\n\n=====");
    Serial.println("    OBTENER DevEUI DEL ESP32");
    Serial.println("=====\\n");

    // Obtener MAC address del ESP32
    uint64_t chipid = ESP.getEfuseMac(); // MAC único del chip

    // Generar DevEUI desde el MAC address
    // El MAC es de 6 bytes, necesitamos 8 para DevEUI
    // Añadimos 0xFFFF en medio (estándar IEEE)
    DEVEUI[0] = (chipid >> 40) & 0xFF;
    DEVEUI[1] = (chipid >> 32) & 0xFF;
    DEVEUI[2] = (chipid >> 24) & 0xFF;
    DEVEUI[3] = 0xFF; // Byte fijo
    DEVEUI[4] = 0xFE; // Byte fijo
    DEVEUI[5] = (chipid >> 16) & 0xFF;
    DEVEUI[6] = (chipid >> 8) & 0xFF;
    DEVEUI[7] = chipid & 0xFF;

    // Imprimir DevEUI en formato TTN (MSB first, big-endian)
    Serial.println("DevEUI (para copiar a TTN):");
    Serial.print("  ");
    for(int i = 0; i < 8; i++) {
        if (DEVEUI[i] < 0x10) Serial.print("0");
        Serial.print(DEVEUI[i], HEX);
    }
    Serial.println();

    // Imprimir DevEUI en formato para código C (LSB first, little-endian)
    Serial.println("\nDevEUI para código Arduino (little-endian):");
    Serial.print("  static const u1_t PROGMEM DEVEUI[8] = { ");
    for(int i = 7; i >= 0; i--) { // Orden inverso
        Serial.print("0x");
        if (DEVEUI[i] < 0x10) Serial.print("0");
        Serial.print(DEVEUI[i], HEX);
        if (i > 0) Serial.print(", ");
    }
    Serial.println(" };" );

    Serial.println("\n=====");
    Serial.println("INSTRUCCIONES:");
    Serial.println("1. Copia el DevEUI de arriba");
    Serial.println("2. Pégalo en TTN al registrar");
    Serial.println("3. Usa el array para tu código");
    Serial.println("=====\\n");
}

void loop() {
    // Nada aquí, solo queríamos obtener el DevEUI
    delay(10000);
}
```

## 1. SALIDA MONITOR SERIAL

=====

OBTENER DevEUI DEL ESP32

=====

DevEUI (para copiar a TTN):

C481F1FFFEBF713C

DevEUI para código Arduino (little-endian):

```
static const u1_t PROGMEM DEVEUI[8] = { 0x3C, 0x71, 0xBF, 0xFE, 0xFF, 0xF1, 0x81, 0xC4 };
```

=====

INSTRUCCIONES:

1. Copia el DevEUI de arriba
2. Pégalo en TTN al registrar
3. Usa el array para tu código

=====

## CREAR LA APLICACIÓN dht22ap

### Create application

Within applications, you can register and manage end devices and their network data. After setting up your device fleet, use one of our many integration options to pass relevant data to your external services.

Learn more in our guide on [Adding Applications](#).

Application ID\*

dht22ap

## CREAR EL DISPOSITIVO FINAL

The screenshot shows the ThingsBoard application interface. At the top, there is a header bar with the application name "dht22ap" in blue text, a "Add label" button with a plus sign, and the text "ID: dht22ap". Below the header, there is a section titled "End devices" with a blue icon. Underneath this, there are two tabs: "Top end devices" (which is highlighted in white) and "Recently active". In the center of the screen, the message "No top devices yet" is displayed in red text, followed by the explanatory text "Your most visited and bookmarked end devices will be listed here". At the bottom, there is a large blue button with the text "+ Register end device" and a plus sign icon.

## End device type

### Input method ?

- Select the end device in the LoRaWAN Device Repository
- Enter end device specifics manually

### Frequency plan ? \*

United States 902-928 MHz, FSB 2 (used by TTN)



### LoRaWAN version ? \*

LoRaWAN Specification 1.0.3



### Regional Parameters version ? \*

RP001 Regional Parameters 1.0.3 revision A



Show advanced activation, LoRaWAN class and cluster settings

## Provisioning information

### JoinEUI ? \*

00 00 00 00 00 00 00 00

Confirm

To continue, please enter the JoinEUI of the end device so we can determine onboarding options

Clic en CONFIRM

## Provisioning information

JoinEUI ⓘ \*

00 00 00 00 00 00 00 00

Reset

This end device can be registered on the network

DevEUI ⓘ \*

C4 81 F1 FF FE BF 71 3C

Generate

0/50 used

AppKey ⓘ \*

91 DC 50 94 EA 51 C1 F4 5A 1F AF 97 44 9B 9C 13

Generate

End device ID ⓘ \*

dht22ed

## Device properties

Labels

Select...

| ▾

Add a label or [create one](#) to categorize your devices

### After registration

- View registered end device
- Register another end device of this type

Register end device

Register end device

## 2. CREACIÓN DE LA APLICACIÓN Y EL END DEVICE EN:

<https://nam1.cloud.thethings.network/console/applications>

Correo: [luisagreda@gmail.com](mailto:luisagreda@gmail.com)  
Password: 7774321Unindiac0U\$

**APLICACIÓN:** dht22ap

**END DEVICE:** dht22ed

Frequency plan:	United States 902-928 MHz, FSB 2 (used by TTN)
LoRaWAN version	LoRaWAN Specification 1.0.3
Regional Parameters version	RP001 Regional Parameters 1.0.3 revision A
Created at	Nov 16, 2025 18:56:48

**Activation information**

AppEUI	0000000000000000
DevEUI	C481F1FFFEBF713C
AppKey	91DC5094EA51C1F45A1FAF97449B9C13

**Session information**

Session start	Nov 21, 2025 07:41:00
Device address	260C24AC
NwkSKey	9FC63E059167BD3D4D0191A73FAAD049
AppSKey	0396AC93B9B31F099A793E5D42035ECD

### **3. CONVERSIÓN A FORMATO DE CÓDIGO C/C++. PROMPT A LA IA (GEMINI):**

Convierte estas credenciales LoRaWAN al formato de código C/C++ (arrays estáticos de tipo u1\_t con \$PROGMEM\$), asegurándote de que el DevEUI esté en orden de bytes little-endian (invertido):

AppEUI: 0000000000000000

DevEUI: C481F1FFFEBF713C

AppKey: 91DC5094EA51C1F45A1FAF97449B9C13

#### 4. RESPUESTA DE LA IA:

```
#include <stdint.h>

// Definición para plataformas con soporte PROGMEM (como Arduino)
// Si no usas PROGMEM, puedes cambiar la línea a: #define PROGMEM
#define PROGMEM

// u1_t es típicamente un alias para uint8_t, pero usamos uint8_t por claridad
typedef uint8_t u1_t;

// --- AppEUI (IEEE EUI-64) ---
// El AppEUI se mantiene en orden de bytes big-endian (orden de la especificación, no invertido)
static const u1_t PROGMEM APPEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

// --- DevEUI (IEEE EUI-64) ---
// **IMPORTANTE**: DevEUI se invierte a little-endian (el byte menos significativo primero)
// DevEUI original: C481F1FFFEBF713C
// DevEUI invertido: 3C 71 BF FE FF F1 81 C4
static const u1_t PROGMEM DEVEUI[8] = { 0x3C, 0x71, 0xBF, 0xFE, 0xFF, 0xF1, 0x81, 0xC4 };

// --- AppKey (Clave de Aplicación de 128 bits) ---
// El AppKey se mantiene en orden de bytes big-endian (orden de la especificación, no invertido)
static const u1_t PROGMEM APPKEY[16] = {
    0x91, 0xDC, 0x50, 0x94, 0xEA, 0x51, 0xC1, 0xF4,
    0x5A, 0x1F, 0xAF, 0x97, 0x44, 0x9B, 0x9C, 0x13
};
```

RESUMEN:

```
static const u1_t PROGMEM APPEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
static const u1_t PROGMEM DEVEUI[8] = { 0x3C, 0x71, 0xBF, 0xFE, 0xFF, 0xF1, 0x81, 0xC4 };
static const u1_t PROGMEM APPKEY[16] = {
    0x91, 0xDC, 0x50, 0x94, 0xEA, 0x51, 0xC1, 0xF4,
    0x5A, 0x1F, 0xAF, 0x97, 0x44, 0x9B, 0x9C, 0x13
};
```

## 5. CÓDIGO DEL TTGO

```
/*
 * TTGO LoRa32-OLED + DHT22 Temperature & Humidity Telemetry
 * OPTIMIZADO PARA TRANSMISIÓN EXACTA CADA 10 SEGUNDOS
 * CON OPTIMIZACIÓN DE ENERGÍA DESDE POWERBANK
 * Aplicación: dht22ap - Dispositivo: dht22ed
 *
 * CONVERSIÓN v1.0 - DHT22 SENSOR:
 * - Soporte para temperatura Y humedad
 * - Identificación del tipo de sensor
 * - Payload de 6 bytes con status del sensor
 * - Display OLED mostrando temperatura y humedad
 * - Manejo robusto de errores
 * - Watchdog para recuperación automática
 */

#include <Arduino.h>

// ----- CONFIGURACIÓN LMIC (CRÍTICO: ANTES DE INCLUDES) -----
#define CFG_sx1276_radio 1
#define CFG_us915 1

// ----- LIBRERÍAS NECESARIAS -----
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <DHT.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>
#include <esp_task_wdt.h>

// ----- CONFIGURACIÓN HARDWARE -----
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define LED_PIN 25

// ----- CONFIGURACIÓN DHT22 -----
#define DHTPIN 21          // GPIO para DHT22
#define DHTTYPE DHT22        // Tipo de sensor DHT

// Watchdog timeout (30 segundos)
#define WDT_TIMEOUT 30

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RST);
DHT dht(DHTPIN, DHTTYPE);

// ----- CONFIGURACIÓN LORAWAN -----
// IMPORTANTE: Estos valores deben coincidir con TTN
// Credenciales de dht22ap / dht22ed
static const u1_t PROGMEM APPEUI[8] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
static const u1_t PROGMEM DEVEUI[8] = { 0x3C, 0x71, 0xBF, 0xFE, 0xFF, 0xF1, 0x81, 0xC4 };
static const u1_t PROGMEM APPKEY[16] = {
    0x91, 0xDC, 0x50, 0x94, 0xEA, 0x51, 0xC1, 0xF4,
    0x5A, 0x1F, 0xAF, 0x97, 0x44, 0x9B, 0x9C, 0x13
};
```

```

void os_getArtEui(u1_t* buf) { memcpy_P(buf, APPEUI, 8); }
void os_getDevEui(u1_t* buf) { memcpy_P(buf, DEVEUI, 8); }
void os_getDevKey(u1_t* buf) { memcpy_P(buf, APPKEY, 16); }

const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 33, 32},
};

// ----- VARIABLES GLOBALES OPTIMIZADAS -----
static uint8_t mydata[6]; // 6 bytes: [Temp_MSB, Temp_LSB, Hum_MSB, Hum_LSB, Status, SensorType]
static osjob_t sendjob;

// TIMING PRECISO
const unsigned long TX_INTERVAL_MS = 10000; // 10 segundos exactos
unsigned long lastTransmissionTime = 0;
bool forceTransmission = false;

// Variables de estado - DHT22
float currentTemp = 0.0;
float currentHum = 0.0;
bool sensorValid = false;
int tx_count = 0;
String network_status = "Iniciando...";
bool joined = false;
uint8_t last_event = 0;
int consecutiveSensorErrors = 0;

// Variables para monitoreo de timing
unsigned long actualIntervals[10] = {0};
int intervalIndex = 0;

// Variables para ahorro de energía
unsigned long lastDisplayUpdate = 0;
const unsigned long DISPLAY_UPDATE_INTERVAL = 5000; // Actualizar display cada 5s
bool displayOn = true;

// ----- FUNCIONES DE GESTIÓN DE ENERGÍA -----
void displaySleep() {
    if (displayOn) {
        display.ssd1306_command(SSD1306_DISPLAYOFF);
        displayOn = false;
    }
}

void displayWake() {
    if (!displayOn) {
        display.ssd1306_command(SSD1306_DISPLAYON);
        displayOn = true;
    }
}

// ----- FUNCIONES DE TIMING PRECISO -----
void calculateTiming() {
    if (lastTransmissionTime > 0) {
        unsigned long actualInterval = millis() - lastTransmissionTime;
        actualIntervals[intervalIndex] = actualInterval;
        intervalIndex = (intervalIndex + 1) % 10;
    }
}

```

```

    Serial.printf("⌚ Intervalo real: %lu ms (objetivo: %lu ms)\n", actualInterval,
TX_INTERVAL_MS);

    // Calcular promedio de últimos 10 intervalos
    unsigned long sum = 0;
    int count = 0;
    for (int i = 0; i < 10; i++) {
        if (actualIntervals[i] > 0) {
            sum += actualIntervals[i];
            count++;
        }
    }

    if (count > 0) {
        Serial.printf("📊 Promedio últimos %d: %.1f s\n", count, sum / (count * 1000.0));
    }
}

bool isTimeToTransmit() {
    unsigned long currentTime = millis();

    if (lastTransmissionTime == 0 || forceTransmission) {
        return true;
    }

    return (currentTime - lastTransmissionTime) >= TX_INTERVAL_MS;
}

unsigned long getTimeUntilNext() {
    if (lastTransmissionTime == 0) return 0;

    unsigned long elapsed = millis() - lastTransmissionTime;
    if (elapsed >= TX_INTERVAL_MS) return 0;

    return (TX_INTERVAL_MS - elapsed) / 1000; // En segundos
}

// ----- FUNCIONES DE SENSOR DHT22 -----
void initializeSensor() {
    dht.begin();
    Serial.println("\n🌡 Sensor DHT22 inicializado");
    Serial.printf("  Pin GPIO: %d\n", DHTPIN);
    Serial.println("  Tipo: DHT22 (AM2302)");
    Serial.println("  Rango Temp: -40°C a +80°C");
    Serial.println("  Rango Hum: 0% a 100%");

    // Lectura inicial de prueba
    delay(2000); // DHT22 necesita 2s después de encendido

    float testTemp = dht.readTemperature();
    float testHum = dht.readHumidity();

    if (!isnan(testTemp) && !isnan(testHum)) {
        Serial.println("  ✓ Sensor respondiendo correctamente");
        Serial.printf("  Lectura inicial: %.1f°C, %.1f%\n", testTemp, testHum);
    } else {
        Serial.println("  ⚠️ ADVERTENCIA: Error en lectura inicial");
    }
}

```

```

void readSensorData() {
    // Leer temperatura y humedad
    currentTemp = dht.readTemperature();
    currentHum = dht.readHumidity();

    // Validar lecturas
    if (isnan(currentTemp) || isnan(currentHum)) {
        consecutiveSensorErrors++;
        Serial.printf("🔴 DHT22: Error lectura (errores consecutivos: %d)\n",
consecutiveSensorErrors);

        sensorValid = false;
        currentTemp = -999.0;
        currentHum = -999.0;

        // Reintentar inicialización si hay demasiados errores
        if (consecutiveSensorErrors > 5) {
            Serial.println("🕒 DHT22: Reintentando inicialización...");
            dht.begin();
            consecutiveSensorErrors = 0;
        }
        return;
    }

    // Validar rangos del DHT22
    // Temperatura: -40°C a +80°C (típicamente)
    // Humedad: 0% a 100%
    if (currentTemp < -40.0 || currentTemp > 80.0 ||
        currentHum < 0.0 || currentHum > 100.0) {
        consecutiveSensorErrors++;
        Serial.printf("🔴 DHT22: Valores fuera de rango (T:%.1f°C, H:%.1f%%)\n", currentTemp,
currentHum);
        sensorValid = false;
        return;
    }

    // Lectura válida
    consecutiveSensorErrors = 0;
    sensorValid = true;

    Serial.printf("🌡️ DHT22: %.1f°C, %.1f%% RH (OK)\n", currentTemp, currentHum);
}

void prepareTempHumData() {
    // Leer datos del sensor
    readSensorData();

    // Si hay errores, usar valores por defecto
    float tempToSend = sensorValid ? currentTemp : -999.0;
    float humToSend = sensorValid ? currentHum : -999.0;

    // Convertir a formato de transmisión (int16_t × 100 y uint16_t × 100)
    int16_t tempInt = (int16_t)(tempToSend * 100);
    uint16_t humInt = (uint16_t)(humToSend * 100);

    // Construir payload de 6 bytes
    mydata[0] = (tempInt >> 8) & 0xFF; // Temp MSB
    mydata[1] = tempInt & 0xFF;           // Temp LSB
    mydata[2] = (humInt >> 8) & 0xFF;   // Hum MSB
}

```

```

mydata[3] = humInt & 0xFF;           // Hum LSB

// Status byte: bit 0 = sensor válido
mydata[4] = sensorValid ? 0x01 : 0x00;

// Identificador: 0x22 para DHT22
mydata[5] = 0x22;

Serial.printf("LCD Payload: [%02X %02X %02X %02X %02X %02X]\n",
              mydata[0], mydata[1], mydata[2], mydata[3], mydata[4], mydata[5]);
}

// ----- FUNCIONES DE PANTALLA - DHT22 -----
void updateTempHumDisplay() {
    display.clearDisplay();
    display.setTextColor(WHITE);

    // ===== TEMPERATURA =====
    // Etiqueta pequeña
    display.setTextSize(1);
    display.setCursor(0, 0);
    display.print("TEMPERATURA:");

    // Temperatura GRANDE
    display.setTextSize(2);
    display.setCursor(0, 12);
    if (sensorValid) {
        display.printf("%.1f", currentTemp);
        display.setTextSize(1);
        display.print("°");
        display.setTextSize(2);
        display.print("C");
    } else {
        display.setTextSize(2);
        display.print("ERROR");
    }

    // ===== HUMEDAD =====
    // Etiqueta pequeña
    display.setTextSize(1);
    display.setCursor(0, 38);
    display.print("HUMEDAD:");

    // Humedad GRANDE
    display.setTextSize(2);
    display.setCursor(0, 50);
    if (sensorValid) {
        display.printf("%.1f", currentHum);
        display.setTextSize(2);
        display.print("%");
    } else {
        display.setTextSize(2);
        display.print("ERROR");
    }

    display.display();
}

void updateInfoDisplay() {
    display.clearDisplay();
    display.setTextColor(WHITE);
}

```

```

display.setTextSize(1);

display.setCursor(0, 0);
display.println("== STATOS DHT22 ==");

display.setCursor(0, 12);
display.printf("TX Total: %d", tx_count);

display.setCursor(0, 22);
display.printf("Next TX: %lus", getTimeUntilNext());

display.setCursor(0, 32);
display.printf("Pin: %d | DHTPIN", DHTPIN);

display.setCursor(0, 42);
display.printf("Sensor:%s", sensorValid ? "OK" : "ER");

display.setCursor(0, 52);
display.printf("Ev:%d Err:%d", last_event, consecutiveSensorErrors);

display.display();
}

void updateDisplay() {
    displayWake();
    updateTempHumDisplay(); // Solo mostrar temperatura y humedad
}

// ----- TRANSMISIÓN OPTIMIZADA -----
void do_send(osjob_t* j) {
    esp_task_wdt_reset();

    if (LMIC.opmode & OP_TXRXPEND) {
        Serial.println("TX ocupado - reintento en 500ms");
        os_setTimedCallback(&sendjob, os_getTime() + ms2osticks(500), do_send);
        return;
    }

    if (!joined) {
        Serial.println("No conectado - reintento en 1s");
        os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(1), do_send);
        return;
    }

    if (!forceTransmission && !isTimeToTransmit()) {
        unsigned long msUntilNext = TX_INTERVAL_MS - (millis() - lastTransmissionTime);
        Serial.printf(" Esperando %lu ms para timing exacto\n", msUntilNext);
        os_setTimedCallback(&sendjob, os_getTime() + ms2osticks(msUntilNext), do_send);
        return;
    }

    calculateTiming();

    digitalWrite(LED_PIN, HIGH);

    prepareTempHumData();

    LMIC_setTxData2(1, mydata, sizeof(mydata), 0);

    lastTransmissionTime = millis();
    forceTransmission = false;
}

```

```

tx_count++;

Serial.printf("TX #%"tx_count" enviado\n", tx_count);

delay(30);
digitalWrite(LED_PIN, LOW);

updateDisplay();
}

// ----- EVENTOS LMIC OPTIMIZADOS -----
void onEvent(ev_t ev) {
    last_event = ev;
    esp_task_wdt_reset();

    switch (ev) {
        case EV_JOINING:
            Serial.println("⌚ Uniendo a TTN dht22ap...");
            network_status = "Joining...";
            break;

        case EV_JOINED:
            Serial.println("✅ CONECTADO A dht22ap!");
            network_status = "Connected!";
            joined = true;

            // Configuración de canales US915
            Serial.println("🔧 Configurando canales US915...");
            for (int i = 0; i < 72; i++) {
                if (i != 8 && i != 16 && i != 24 && i != 32 &&
                    i != 40 && i != 48 && i != 56 && i != 64) {
                    LMIC_disableChannel(i);
                }
            }

            LMIC_setLinkCheckMode(0);
            LMIC_setDrTxpow(DR_SF7, 14);
            LMIC_setAddrMode(0);
            LMIC_setClockError(MAX_CLOCK_ERROR * 2 / 100);

            Serial.println("⚙️ Optimizaciones aplicadas: ADR OFF, SF7 fijo");

            forceTransmission = true;
            lastTransmissionTime = 0;
            os_setTimedCallback(&sendjob, os_getTime() + sec2osticks(2), do_send);
            break;

        case EV_JOIN_FAILED:
            Serial.println("❌ Error de conexión");
            network_status = "Join Failed";
            joined = false;
            break;

        case EV_TXCOMPLETE:
        {
            Serial.println("✓ TX completado");
            network_status = "TX OK";

            if (LMIC.txrxFlags & TXRX_ACK) {
                Serial.println("✉️ ACK recibido");
            }
        }
    }
}

```

```

    }

    if (LMIC.dataLen) {
        Serial.printf("⬇️ Downlink: %d bytes\n", LMIC.dataLen);
    }

    unsigned long msUntilNext = TX_INTERVAL_MS;
    os_setTimedCallback(&sendjob, os_getTime() + ms2osticks(msUntilNext), do_send);

    Serial.printf("⌚ Próxima TX programada en %lu ms\n", msUntilNext);
}
break;

case EV_RESET:
    Serial.println("♻️ LMIC Reset");
    network_status = "Reset";
    break;

default:
    Serial.printf("ℹ️ Evento: %d\n", (unsigned) ev);
    break;
}

updateDisplay();
}

// ----- SETUP OPTIMIZADO -----
void setup() {
    Serial.begin(115200);
    delay(1000);

    Serial.println("\n=====");
    Serial.println(" LoRaWAN DHT22 v1.0");
    Serial.println(" TEMPERATURA + HUMEDAD");
    Serial.println(" Transmisión: 10.0s exactos");
    Serial.println(" ADR OFF, SF7 fijo, Clock 2%");
    Serial.println("=====\\n");

    // Configurar watchdog
    esp_task_wdt_init(WDT_TIMEOUT, true);
    esp_task_wdt_add(NULL);
    Serial.println("🐕 Watchdog activado (30s timeout)");

    // Configurar LED
    pinMode(LED_PIN, OUTPUT);
    digitalWrite(LED_PIN, LOW);

    // Configurar DHT22
    initializeSensor();

    // Configurar OLED
    pinMode(OLED_RST, OUTPUT);
    digitalWrite(OLED_RST, LOW);
    delay(5);
    digitalWrite(OLED_RST, HIGH);

    Wire.begin(OLED_SDA, OLED_SCL);

    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) {
        Serial.println("✖️ Error OLED");
    }
}

```

```

        for(;;);
    }

display.clearDisplay();
display.setTextColor(WHITE);
display.setTextSize(1);
display.setCursor(0, 0);
display.println("DHT22 v1.0");
display.setCursor(0, 16);
display.println("LoRaWAN Temp+Hum");
display.setCursor(0, 32);
display.println("Cada 10.0s exactos");
display.setCursor(0, 48);
display.println("Conectando TTN...");
display.display();

// Test LED
for (int i = 0; i < 3; i++) {
    digitalWrite(LED_PIN, HIGH);
    delay(100);
    digitalWrite(LED_PIN, LOW);
    delay(100);
}

// Configuración LMIC
os_init();
LMIC_reset();
LMIC_selectSubBand(1);
LMIC_setClockError(MAX_CLOCK_ERROR * 2 / 100);
LMIC_setLinkCheckMode(0);
LMIC_setAddrMode(0);

Serial.println("📡 Iniciando conexión optimizada...");
LMIC_startJoining();

do_send(&sendjob);
}

// ----- LOOP OPTIMIZADO -----
void loop() {
    esp_task_wdt_reset();
    os_runloop_once();
    yield();
}

```

## 6. PLATFORMIO.INI

```
; PlatformIO Project Configuration File
;
; Sistema de Telemetría DHT22 + LoRaWAN
; Hardware: TTGO LoRa32-OLED v1
; Aplicación TTN: dht22ap
; Dispositivo: dht22ed

[env:ttgo-lora32-v1]
platform = espressif32
board = ttgo-lora32-v1
framework = arduino

; Configuración del monitor serial
monitor_speed = 115200
monitor_filters =
    esp32_exception_decoder
    time

; Opciones de compilación
build_flags =
    -D ARDUINO_LMIC_PROJECT_CONFIG_H_SUPPRESS
    -D CFG_us915=1
    -D CFG_sx1276_radio=1
    -D LMIC_PRINTF_TO=Serial
    -D DISABLE_PING
    -D DISABLE_BEACONS
    -D USE_ORIGINAL_AES
    -D hal_init=LMICHAL_init

; Librerías necesarias
lib_deps =
    ; Display OLED
    adafruit/Adafruit SSD1306@^2.5.7
    adafruit/Adafruit GFX Library@^1.11.3
    adafruit/Adafruit BusIO@^1.14.1

    ; Sensor DHT22
    adafruit/DHT sensor library@^1.4.4
    adafruit/Adafruit Unified Sensor@^1.1.6

    ; LoRaWAN
    mcci-catena/MCCI LoRaWAN LMIC library@^4.1.1

; Configuración de particiones (opcional, para más espacio)
board_build.partitions = default.csv

; Opciones de upload
upload_speed = 921600

; Configuración adicional para debugging (opcional)
; build_type = debug
; debug_tool = esp-prog
; debug_init_break = tbreak setup
```

**NOTA.-** Se ha eliminado la línea:

```
upload_port = AUTO
```

## 7. PAYLOAD UPLINK

```
// Decoder JavaScript para DHT22 (temperatura + humedad) vía LoRaWAN
// Para aplicación "dht22ap" en TTN
// Versión 1.0 - Sensor DHT22

function decodeUplink(input) {
    // Obtener los bytes recibidos
    var bytes = input.bytes;

    // Verificar que hay datos
    if (!bytes || bytes.length === 0) {
        return {
            errors: ["No hay datos en el payload"]
        };
    }

    // Verificar formato básico (esperamos 6 bytes)
    if (bytes.length < 6) {
        return {
            data: {
                raw: bytes,
                warning: "Payload menor a 6 bytes"
            }
        };
    }

    // ====== TEMPERATURA ======
    // Reconstruir el valor de temperatura de 16 bits (MSB, LSB)
    var tempRaw = (bytes[0] << 8) | bytes[1];

    // Determinar si el valor es negativo (complemento a 2)
    if (tempRaw & 0x8000) {
        tempRaw = -(0x10000 - tempRaw);
    }

    // Convertir a valor decimal (se dividió por 100 al enviar)
    var temperature = tempRaw / 100;
    temperature = parseFloat(temperature.toFixed(1));

    // ====== HUMEDAD ======
    // Reconstruir el valor de humedad de 16 bits (MSB, LSB)
    var humRaw = (bytes[2] << 8) | bytes[3];

    // Convertir a valor decimal
    var humidity = humRaw / 100;
    humidity = parseFloat(humidity.toFixed(1));

    // ====== STATUS BYTE ======
    var statusByte = bytes[4];
    var sensorValid = (statusByte & 0x01) !== 0;

    // ====== IDENTIFICADOR ======
    var sensorType = bytes[5];
    var sensorName = "";

    if (sensorType === 0x22) {
        sensorName = "DHT22";
    } else {
        sensorName = "Unknown";
    }
}
```

```

// ===== VALIDACIÓN DHT22 =====
// Rango válido DHT22: Temp: -40°C a +80°C, Hum: 0% a 100%
var tempInRange = (temperature >= -40 && temperature <= 80);
var humInRange = (humidity >= 0 && humidity <= 100);

// Si el sensor marca error (-999), considerarlo inválido
if (temperature < -900 || humidity < -900) {
    sensorValid = false;
    tempInRange = false;
    humInRange = false;
}

// ===== CONSTRUIR RESPUESTA =====
var warnings = [];

if (!sensorValid || !tempInRange) {
    warnings.push("Temperatura: Error o fuera de rango");
}
if (!sensorValid || !humInRange) {
    warnings.push("Humedad: Error o fuera de rango");
}

return {
    data: {
        // Datos principales
        temperature: temperature,
        temperature_valid: sensorValid && tempInRange,
        temperature_status: (sensorValid && tempInRange) ? "OK" : "ERROR",

        humidity: humidity,
        humidity_valid: sensorValid && humInRange,
        humidity_status: (sensorValid && humInRange) ? "OK" : "ERROR",

        // Información general
        temperature_unit: "°C",
        humidity_unit: "%",
        sensor_type: sensorName,
        sensor_id: sensorType,

        // Metadata
        bytes_received: bytes.length,
        status_byte: statusByte,

        // Información de red (si está disponible)
        rssi: input.metadata ? input.metadata.rssi : null,
        snr: input.metadata ? input.metadata.snr : null
    },
    warnings: warnings,
    errors: []
};

}

```

## 8. MQTT

The screenshot shows the MQTT configuration page for a device named "dht22ap". At the top, there are buttons for "Add label" and "Edit", and status indicators showing "Last activity just now" and "0 End devices". Below the header, the title "MQTT" is displayed. A descriptive text explains that MQTT is a publish/subscribe messaging protocol designed for IoT, and it's mentioned that every application on TTS automatically exposes an MQTT endpoint. It also states that to connect to the MQTT server, a new API key needs to be created, which will function as the connection password. Existing API keys can also be used if they have the necessary rights granted. There is a "Further resources" section with links to the "MQTT server" and the "Official MQTT website". The main configuration area is titled "Connection information" and includes fields for the "MQTT server host". Under "Public address", the value is "nam1.cloud.thethings.network:1883". Under "Public TLS address", the value is "nam1.cloud.thethings.network:8883". Below this, the "Connection credentials" section contains fields for "Username" (set to "dht22ap@ttn") and a "Generate new API key" button. The entire configuration page has a dark theme.

**Generate new API key**

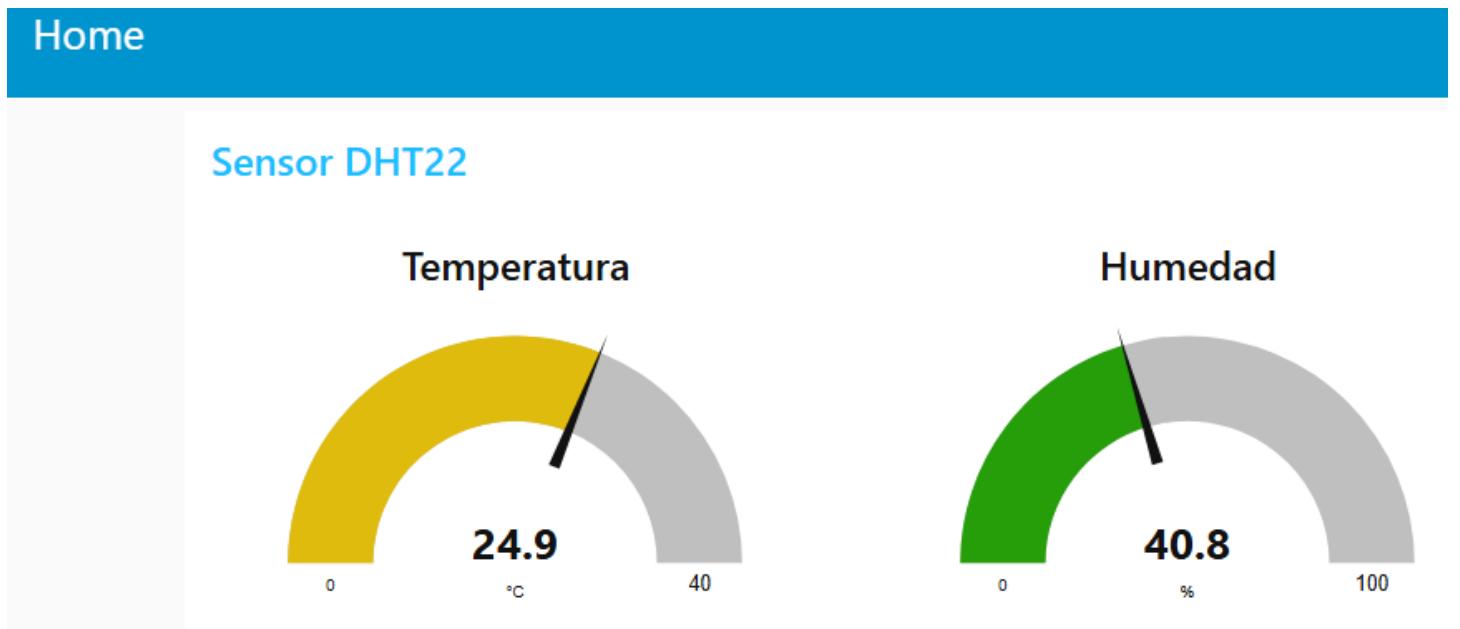
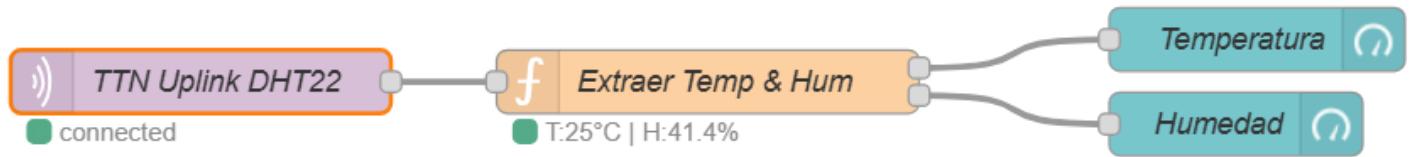
NNXS .Q6T3MBZ276C5F2FN6MHDS55763PE5VFHOHC5QRA .VCS32FQ43Y7YEAIJFLXXRLV2D05BCVW2I5C2D5II  
PKGBD7EIQ

Username: dht22ap@ttn

Public TLS address: nam1.cloud.thethings.network:8883

Topic: v3/dht22ap@ttn/devices/dht22ed/up

## 9. NODE-RED



```
[{"id": "flow_dht22", "type": "tab", "label": "TTN DHT22", "disabled": false, "info": "Dashboard\npara sensor DHT22 (temperatura + humedad) vía\nLoRaWAN"}, {"id": "mqtt_in_dht22", "type": "mqtt in", "z": "flow_dht22", "name": "TTN Uplink\nDHT22", "topic": "v3/dht22ap@ttn/devices/dht22ed/up", "qos": "0", "datatype": "auto-\ndetect", "broker": "broker_ttn_dht22", "nl": false, "rap": true, "rh": 0, "inputs": 0, "x": 130, "y": 8\n0, "wires": [{"function_extract_data"}]}, {"id": "function_extract_data", "type": "function", "z": "flow_dht22", "name": "Extraer Temp & Hum", "func": "// Extraer datos del mensaje de\nTTN\\nvar decoded = msg.payload.uplink_messagedecoded_payload;\\n\\n// Temperatura\\nvar\ntemp = decoded.temperature;\\nvar tempValid = decoded.temperature_valid;\\n\\n//\nHumedad\\nvar hum = decoded.humidity;\\nvar humValid = decoded.humidity_valid;\\n\\n// Crear\nmensajes separados para cada gauge\\nvar msg1 = {\\n    payload: tempValid ? temp : null,\\n    topic: \"temperature\\n\"};\\n\\nvar msg2 = {\\n    payload: humValid ? hum : null,\\n    topic: \"humidity\\n\"};\\n\\n// Mostrar en el estado del nodo\\nnode.status({\\n    fill:\n    \"green\",\\n    shape: \"dot\",\\n    text: \"T:\\\" + (tempValid ? temp + \"°C\\\" : \"ERR\\\")\n    + \" | H:\\\" + (humValid ? hum + \"%\\\" : \"ERR\\\")\\n\"});\\n\\nreturn [msg1,\nmsg2];", "outputs": 2, "timeout": "", "noerr": 0, "initialize": "", "finalize": "", "libs": [], "x": 37\n0, "y": 80, "wires": [{"gauge_temperature"}, {"gauge_humidity"}]}, {"id": "gauge_temperature", "type": "ui_gauge", "z": "flow_dht22", "name": "Temperatura", "group": "group_dht22", "order": 1, "width": 6, "height": 4, "gtype": "gage", "title": "Temperatura", "label": "°C", "format": "{{value}}", "min": 0, "max": 40, "colors": ["#0095e8", "#e6e600", "#ca3838"], "seg1": "", "seg2": "", "diff": false, "className": "", "x": 630, "y": 60, "wires": []}, {"id": "gauge_humidity", "type": "ui_gauge", "z": "flow_dht22", "name": "Humedad", "group": "group_dht22", "order": 2, "width": 6, "height": 4, "gtype": "gage", "title": "Humedad", "label": "%", "format": "{{value}}", "min": 0, "max": 100, "colors": ["#ca3838", "#00b500", "#0095e8"], "seg1": "", "seg2": "", "diff": false, "className": "", "x": 620, "y": 100, "wires": []}, {"id": "broker_ttn_dht22", "type": "mqtt-broker", "name": "TTN\nMQTT\nBroker", "broker": "nam1.cloud.thethings.network", "port": 8883, "tls": "", "clientid": "", "autoConnect": true, "useTLS": true, "protocolVersion": 4, "keepalive": 60, "cleansession": true, "autoUnsubscribe": true, "birthTopic": "", "birthQos": 0, "birthRetain": false, "birthPayload": "", "birthMsg": {}, "closeTopic": "", "closeQos": 0, "closeRetain": false, "closePayload": "", "closeMsg": {}, "willTopic": "", "willQos": 0, "willRetain": false, "willPayload": "", "willMsg": {}, "userProps": "", "sessionExpiry": ""}, {"id": "group_dht22", "type": "ui_group", "name": "Sensor\nDHT22", "tab": "tab_home", "order": 1, "disp": true, "width": 24, "collapse": false, "className": ""}, {"id": "tab_home", "type": "ui_tab", "name": "Home", "icon": "dashboard", "disabled": false, "hidden": false}, {"id": "fe4f27cd00390b23", "type": "global-config", "env": [], "modules": {"node-red-dashboard": "3.6.6"}}]
```

## 10. SALIDA MONITOR SERIAL

```
1:07:43.903 > =====
11:07:43.906 > LoRaWAN DHT22 v1.0
11:07:43.908 > TEMPERATURA + HUMEDAD
11:07:43.910 > Transmisión: 10.0s exactos
11:07:43.913 > ADR OFF, SF7 fijo, Clock 2%
11:07:43.915 > =====
11:07:43.918 >
11:07:43.919 > 🐶 Watchdog activado (30s timeout)
11:07:43.922 >
11:07:43.922 > 🌡 Sensor DHT22 inicializado
11:07:43.925 > Pin GPIO: 21
11:07:43.926 > Tipo: DHT22 (AM2302)
11:07:43.930 > Rango Temp: -40°C a +80°C
11:07:43.932 > Rango Hum: 0% a 100%
11:07:45.931 > ✓ Sensor respondiendo correctamente
11:07:45.935 > Lectura inicial: 25.0°C, 42.3%
11:07:46.594 > 🚀 Iniciando conexión optimizada...
11:07:46.599 > 🛡️ No conectado - reinicio en 1s
11:07:46.602 > 🌐 Uniendo a TTN dht22ap...
11:07:46.621 > 📊 Evento: 17
11:07:47.595 > 🛡️ TX ocupado - reinicio en 500ms
11:07:48.095 > 🛡️ TX ocupado - reinicio en 500ms
11:07:48.595 > 🛡️ TX ocupado - reinicio en 500ms
11:07:49.095 > 🛡️ TX ocupado - reinicio en 500ms
11:07:49.594 > 🛡️ TX ocupado - reinicio en 500ms
11:07:50.094 > 🛡️ TX ocupado - reinicio en 500ms
11:07:50.595 > 🛡️ TX ocupado - reinicio en 500ms
11:07:51.094 > 🛡️ TX ocupado - reinicio en 500ms
11:07:51.595 > 🛡️ TX ocupado - reinicio en 500ms
11:07:52.095 > 🛡️ TX ocupado - reinicio en 500ms
11:07:52.136 > ✅ ¡CONECTADO A dht22ap!
11:07:52.139 > 🛡️ Configurando canales US915...
11:07:52.142 > 🛡️ Optimizaciones aplicadas: ADR OFF, SF7 fijo
11:07:54.145 > 🌡 DHT2 2: 25.0°C, 42.0% RH (OK)
11:07:54.148 > 📦 Payload: [09 C4 10 68 01 22]
11:07:54.151 > 📊 Evento: 17
11:07:54.173 > 🚢 TX #1 enviado
11:08:00.312 > ✓ TX completado
11:08:00.315 > ⏱️ Próxima TX programada en 10000 ms
```

SALIDA MONITOR SERIAL 1:07:43.903 > =====

11:07:43.906 > LoRaWAN DHT22 v1.0

11:07:43.908 > TEMPERATURA + HUMEDAD

11:07:43.910 > Transmisión: 10.0s exactos

11:07:43.913 > ADR OFF, SF7 fijo, Clock 2%

11:07:43.915 > =====

11:07:43.918 >

11:07:43.919 > 🐶 Watchdog activado (30s timeout)

11:07:43.922 >

11:07:43.922 > 🌡 Sensor DHT22 inicializado

11:07:43.925 > Pin GPIO: 21

11:07:43.926 > Tipo: DHT22 (AM2302)

11:07:43.930 > Rango Temp: -40°C a +80°C

11:07:43.932 > Rango Hum: 0% a 100%

11:07:45.931 > ✓ Sensor respondiendo correctamente

11:07:45.935 > Lectura inicial: 25.0°C, 42.3%

11:07:46.594 > 🚀 Iniciando conexión optimizada...

11:07:46.599 > 🚧 No conectado - reinicio en 1s

11:07:46.602 > 🔄 Uniendo a TTN dht22ap...

11:07:46.621 > 📖 Evento: 17

11:07:47.595 > 🚧 TX ocupado - reinicio en 500ms

11:07:48.095 > 🚧 TX ocupado - reinicio en 500ms

11:07:48.595 > 🚧 TX ocupado - reinicio en 500ms

11:07:49.095 > 🚧 TX ocupado - reinicio en 500ms

11:07:49.594 > 🚧 TX ocupado - reinicio en 500ms

11:07:50.094 > 🚧 TX ocupado - reinicio en 500ms

11:07:50.595 > 🚧 TX ocupado - reinicio en 500ms

11:07:51.094 > 🚧 TX ocupado - reinicio en 500ms

11:07:51.595 > 🚧 TX ocupado - reinicio en 500ms

11:07:52.095 > 🚧 TX ocupado - reinicio en 500ms

11:07:52.136 > ✅ ¡CONECTADO A dht22ap!

11:07:52.139 > 🛠 Configurando canales US915...

11:07:52.142 > 🛡 Optimizaciones aplicadas: ADR OFF, SF7 fijo

11:07:54.145 > 🌡 DHT2 2: 25.0°C, 42.0% RH (OK)

11:07:54.148 > 📦 Payload: [09 C4 10 68 01 22]

11:07:54.151 > 📖 Evento: 17

11:07:54.173 > 🚢 TX #1 enviado

11:08:00.312 > ✓ TX completado

11:08:00.315 > ⏱ Próxima TX programada en 10000 ms

## 11. SALIDA END LIVE

TIME	TYPE	DATA PREVIEW	Verbose stream
↑ 11:06:50	Forward uplink data message	DevAddr: 26 0C ED 2F Payload: { bytes_received: 6, humidity: 61.8, humidity_status: "OK", humidity_unit: "%", humidity_valid: true }	<input checked="" type="radio"/> X
↑ 11:06:50	Successfully processed data message	DevAddr: 26 0C ED 2F	
↑ 11:06:34	Forward uplink data message	DevAddr: 26 0C ED 2F Payload: { bytes_received: 6, humidity: 99.9, humidity_status: "OK", humidity_unit: "%", humidity_valid: true }	
↑ 11:06:34	Successfully processed data message	DevAddr: 26 0C ED 2F	
↑ 11:06:18	Forward uplink data message	DevAddr: 26 0C ED 2F Payload: { bytes_received: 6, humidity: 41.6, humidity_status: "OK", humidity_unit: "%", humidity_valid: true }	

TTGO LoRa32-OLED V1

