# Ingeniería inversa de sensores Oregon Scientific THN132N

## 1. Introducción

Este trabajo presenta el proceso completo de ingeniería inversa, modelado y reproducción de las tramas de un sensor de temperatura inalámbrico Oregon Scientific THN132N, con el objetivo de generar tramas sintéticas totalmente compatibles tanto con el decodificador software rtl_433 como con la estación meteorológica de consumo Oregon Scientific BAR206.

## 2. Objetivos

Los objetivos principales del proyecto son:

- Comprender el protocolo de radiofrecuencia Oregon Scientific v2.1 utilizado por el sensor THN132N.
- Obtener tramas reales capturadas del sensor original y analizarlas en detalle.
- Deducir la estructura interna del payload EC40, incluyendo temperatura, canal, house code y campo R12.
- Reconstruir las tablas internas P[d] y M[e] que gobiernan el cálculo del campo R12.
- Implementar un generador de tramas sintéticas en Python totalmente compatibles con BAR206.
- Implementar un emisor RF basado en ESP32 + módulo 433, capaz de emular un sensor THN132N real.
- Validar experimentalmente que las tramas generadas son aceptadas por la pantalla BAR206.

## 3. Protocolo Oregon Scientific v2.1 y sensor THN132N

El protocolo Oregon Scientific v2.1 utiliza modulación OOK sobre 433,92 MHz con codificación Manchester y un preámbulo característico compuesto por patrones 0x55. El sensor THN132N transmite periódicamente un telegrama de 168 bits (21 bytes equivalentes), que tras demanchesterizar y reordenar nibbles da lugar a un payload de 8 bytes identificado como tipo EC40 (0xEC40).

El payload EC40 contiene, entre otros campos, el identificador de sensor (house code), el canal, la temperatura en BCD y un campo de 12 bits denominado R12 que no forma parte del checksum oficial pero que la estación BAR206 sí utiliza para aceptar o rechazar tramas.

## 4. Ingeniería inversa del payload EC40

A partir de múltiples capturas de tramas reales usando rtl_433 y un dongle RTL-SDR, se extrajeron las tramas EC40 y se construyó un CSV con campos: fecha, trama RAW 168 bits, payload EC40, temperatura interpretada, canal, house code, nibble alto de R12 y byte bajo de R12. Esto permitió estudiar la dependencia del campo R12 respecto a la temperatura visible.

Tras analizar cientos de muestras se observó que el campo R12 no se comporta como un simple CRC ni como un contador lineal, sino que puede expresarse como la combinación XOR de dos tablas: una tabla P[d] dependiente de la décima de grado, y una tabla M[e] dependiente de la parte entera de la temperatura en grados Celsius.

### 4.1. Tablas P[d] y M[e]

La temperatura se descompone en parte entera e y décima d según:

e = int(T)

d = décima de $|T|$, es decir, $d \in [0,9]$

El campo R12 se modela como:

$$R12(T) = P[d] \text{ XOR } M[e]$$

donde P[d] es una tabla de 10 entradas (d = 0..9) y M[e] es una tabla indexada por la parte entera e en el rango $[-16, 54]$ °C reconstruida a partir de las capturas.

Tabla 1. Tabla P[d] reconstruida (décimas de grado).

| d=0 | d=1 | d=2 | d=3 | d=4 | d=5 | d=6 | d=7 | d=8 | d=9 |
|---|---|---|---|---|---|---|---|---|---|
| 0x000 | 0x075 | 0x0EA | 0x09F | 0x0B5 | 0x0C0 | 0x05F | 0x02A | 0x06B | 0x01E |

Tabla 2. Tabla M[e] reconstruida (parte entera de la temperatura).

| e (°C) | M[e] |
|---|---|
| -16 | 0x2A1 |
| -15 | 0x252 |

| | |
|---|---|
| -14 | 0x203 |
| -13 | 0x2B5 |
| -12 | 0x2E4 |
| -11 | 0x217 |
| -10 | 0x246 |
| -9 | 0x29A |
| -8 | 0x2CB |
| -7 | 0x2F7 |
| -6 | 0x2A6 |
| -5 | 0x255 |
| -4 | 0x204 |
| -3 | 0x2B2 |
| -2 | 0x2E3 |
| -1 | 0x210 |
| 0 | 0x2C2 |
| 1 | 0x148 |
| 2 | 0x1BB |
| 3 | 0x1EA |
| 4 | 0x15C |
| 5 | 0x10D |
| 6 | 0x1FE |
| 7 | 0x1AF |
| 8 | 0x193 |
| 9 | 0x1C2 |

| | |
|---|---|
| 10 | 0x11E |
| 11 | 0x14F |
| 12 | 0x1BC |
| 13 | 0x236 |
| 14 | 0x280 |
| 15 | 0x10A |
| 16 | 0x1F9 |
| 17 | 0x1A8 |
| 18 | 0x194 |
| 19 | 0x866 |
| 20 | 0x2CC |
| 21 | 0x146 |
| 22 | 0x1B5 |
| 23 | 0x1E4 |
| 24 | 0x152 |
| 25 | 0x103 |
| 26 | 0x1F0 |
| 27 | 0x1A1 |
| 28 | 0x246 |
| 29 | 0x1CC |
| 30 | 0x110 |
| 31 | 0x141 |
| 32 | 0x1B2 |
| 33 | 0x1E3 |

| | |
|---|---|
| 34 | 0x8F6 |
| 35 | 0x8A7 |
| 36 | 0x854 |
| 37 | 0x805 |
| 38 | 0x839 |
| 39 | 0x868 |
| 40 | 0x8C6 |
| 41 | 0x897 |
| 42 | 0x864 |
| 43 | 0x835 |
| 44 | 0x883 |
| 45 | 0x8D2 |
| 46 | 0x821 |
| 47 | 0x870 |
| 48 | 0x84C |
| 49 | 0x81D |
| 50 | 0x162 |
| 51 | 0x133 |
| 52 | 0x863 |
| 53 | 0x191 |
| 54 | 0x884 |

## 4.2. Ejemplo de cálculo de R12

Como ejemplo, para T = 14,9 °C se tiene e = 14, d = 9. Por tanto:

R12 = P[9] XOR M[14]

= 0x01E XOR 0x280
= 0x29E

Este valor encaja exactamente con el campo R12 observado en tramas reales y ha sido validado experimentalmente en la pantalla BAR206.

## 5. Implementación del generador de tramas en Python

Se ha desarrollado un script en Python, gen_tramas_thn132n.py, que genera tramas EC40 y sus correspondientes tramas RAW 168 bits para un rango configurable de temperaturas, canal y house code. El script aplica la ecuación R12 = P[d] XOR M[e], calcula el checksum Oregon v2.1 y construye la codificación Manchester, partiendo de un preámbulo base.

Listado 1. Fragmento del generador de tramas THN132N en Python (cálculo de R12 y payload EC40).

```
def calc_R12(temp_c: float) -> int:
    e, d = temp_to_e_d(temp_c)
    if e < M_MIN_E:
        e = M_MIN_E
    if e > M_MAX_E:
        e = M_MAX_E
    # Clamp d to valid range [0-9] to avoid IndexError
    if d < 0:
        d = 0
    if d > 9:
        d = 9
    P = P_TABLE[d]
    M = M_TABLE[e]
    return (P ^ M) & 0x0FFF

def temp_to_bcd_bytes(temp_c: float):
    sign_bit = 0
    if temp_c < 0:
        sign_bit = 1
        temp_c = -temp_c
```

```
    t10 = int(round(temp_c * 10.0))
    d0 = t10 % 10
    ent = t10 // 10
    u = ent % 10
    d1 = (ent // 10) % 10

    msg4 = (d0 << 4) | (u & 0x0F)

    hundreds = 0
    low_nibble = (sign_bit << 3) | (hundreds & 0x07)
    msg5 = ((d1 & 0x0F) << 4) | low_nibble
    return msg4, msg5

def calc_os21_checksum(msg):
    s = 0
    for i in range(6):
        b = msg[i]
        s += (b >> 4) + (b & 0x0F)
    s &= 0xFF
    high = (s & 0xF0) >> 4
    low  = (s & 0x0F)
    return (low << 4) | high

def reflect_nibbles(buf: bytes) -> bytes:
    return bytes(((b & 0x0F) << 4) | (b >> 4) for b in buf)

# ---------- EC40 post-reflect ----------

def build_ec40_post(temp_c: float,
                    channel: int = 1,
                    device_id: int = 247) -> bytes:
    msg = [0] * 8
    msg[0] = 0xEC
    msg[1] = 0x40

    id_low  = device_id & 0x0F
    id_high = device_id & 0xF0
```

```
msg[2] = ((channel & 0x0F) << 4) | id_low
msg[3] = id_high

msg[4], msg[5] = temp_to_bcd_bytes(temp_c)

r12 = calc_R12(temp_c)
msg[3] = (msg[3] & 0xF0) | ((r12 >> 8) & 0x0F)
msg[7] = r12 & 0xFF

msg[6] = calc_os21_checksum(msg)
return bytes(msg)
```

---

## 6. Emisor ESP32 + módulo 433 MHz

Para emular el sensor THN132N se ha implementado un emisor basado en ESP32 y un módulo OOK de 433 MHz. El ESP32 utiliza el periférico RMT (Remote Control) para generar trenes de pulsos con el temporizado correcto (~488 µs por semibit) y transmitir tramas Manchester ya codificadas en bruto.

El firmware en C++/Arduino define una función build_raw_ook_frame() que convierte una trama en hexadecimal a un vector de rmt_item32_t, donde cada bit se traduce a un pulso ON/OFF con la duración adecuada. En el loop principal se selecciona una trama RAW, se emite dos veces con una separación de 4 ms (como hace el sensor original) y se repite el proceso cada cierto intervalo.

El código completo del emisor se encuentra en el fichero:
/mnt/data/oregon_transmitter.ino

## 7. Resultados experimentales

Se han realizado pruebas emitiendo tramas sintéticas generadas con el script Python y transmitidas por el ESP32. El decodificador rtl_433 reconoce correctamente todas las tramas como sensores Oregon-THN132N, mostrando la temperatura, canal y house code esperados.

La validación clave se ha realizado con la pantalla BAR206: se han generado y emitido tramas para temperaturas completamente sintéticas, por ejemplo 16,9 °C

y 14,9 °C, y la estación ha mostrado la temperatura correspondiente como si proviniera de un sensor THN132N real. Esto confirma que el modelo de R12 y las tablas P[d], M[e] son coherentes con la implementación original del fabricante para esos rangos de temperatura.

## 8. Visualización de tablas y tramas

Figura 1. Evolución de M[e] en función de la parte entera de la temperatura.

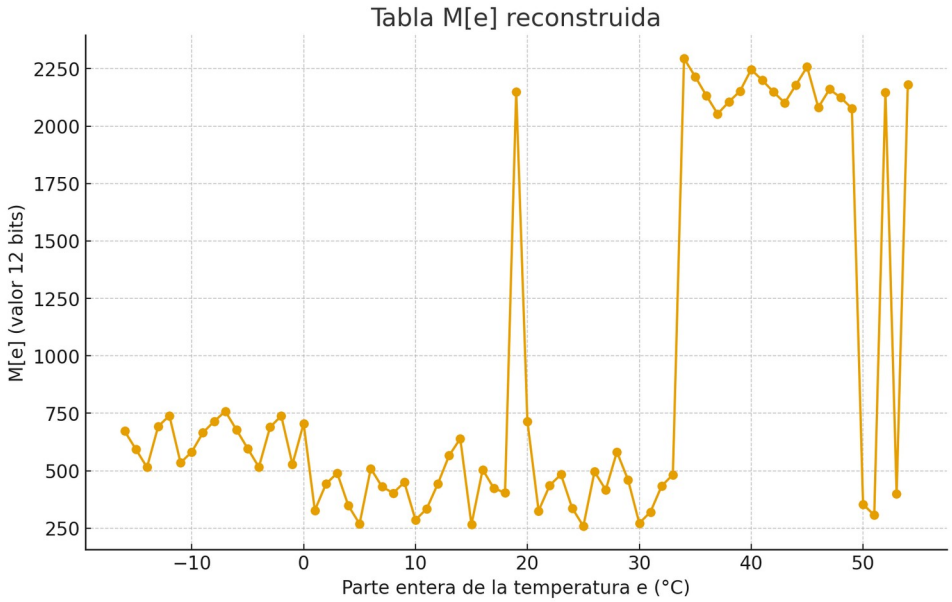
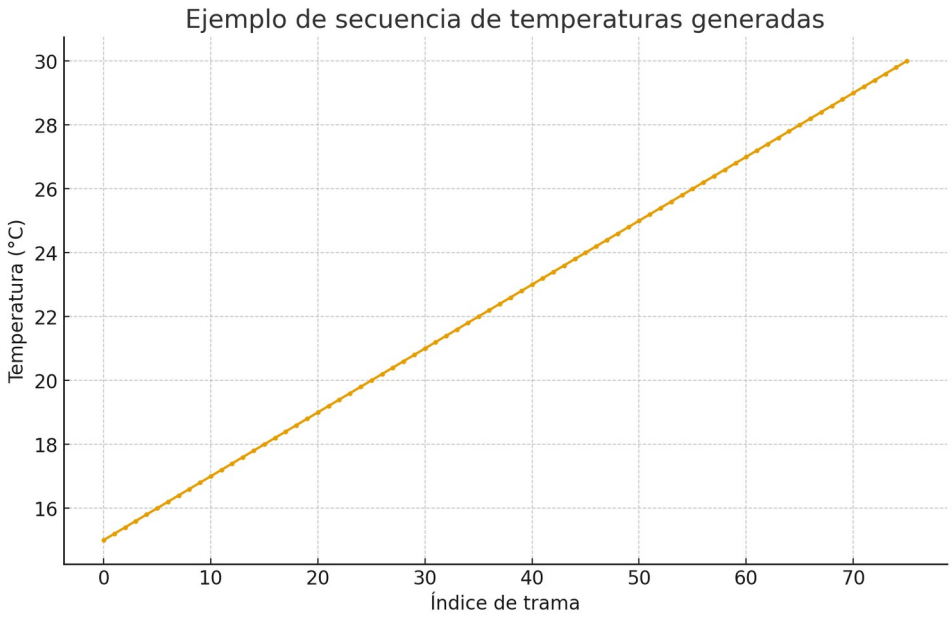



Figura 2. Ejemplo de temperaturas para las que se han generado tramas sintéticas THN132N.




## 9. Conclusiones

Se ha demostrado que es posible reproducir de forma completamente sintética las tramas de un sensor Oregon Scientific THN132N, incluyendo el campo interno R12 que la estación BAR206 utiliza para validar la autenticidad del sensor. El modelo R12 = P[d] XOR M[e], junto con las tablas reconstruidas P[d] y M[e], permite generar tramas compatibles para un amplio rango de temperaturas.

La implementación práctica, basada en un ESP32 y un módulo OOK de 433 MHz, permite emular al sensor original hasta el punto de que la estación BAR206 muestra las temperaturas generadas de forma indistinguible de un sensor real. Esto abre la puerta a usos como pruebas de estaciones, simuladores de sensores o integración de datos procedentes de otras fuentes en estaciones comerciales cerradas.

## 10. Trabajo futuro

Como líneas futuras de trabajo se propone:

- Completar la validación experimental de toda la tabla M[e] en el rango −16 a 54 °C.
- Estudiar el comportamiento del campo R12 ante cambios de canal y house code adicionales.
- Generalizar la metodología a otros sensores Oregon (humedad, lluvia, viento) con estructuras de payload distintas.

• Diseñar una interfaz gráfica que permita seleccionar una temperatura y enviar la trama correspondiente al vuelo desde un ordenador o desde el propio ESP32.