



Nesos Group
INTELLIGENT PROPERTY MANAGEMENT

LoRAD PRO – Radon Monitor

Integration Guide

Document overview

Document Type:	Integration Guide
Sponsoring organisation:	Nesos Group
Product Name:	LoRAD PRO Radon Monitor
Author:	Jonathan Anderson
Date:	04 July 2025
Document Status:	Published

Confidentiality Statement.

The information embodied in this document is strictly confidential and is supplied on the understanding that it will be held confidentially and not disclosed to third parties without the prior written consent of Nesos Group.

Contents

Document overview	2
Confidentiality Statement.	2
Overview	4
Understanding the Payload	5
Network Server Integration	5
ChirpStack v4.....	5
The Things Network (TTN v3).....	7
Helium Console	7
Extracting Radon Values.....	7
Implementation Workflow	8
Sensor Data Format.....	8
Unit Conversion Reference	8
Testing and Validation	9
Calculating Long-Term Averages	9
Important Averaging Considerations	10
Placement Guidelines	10
Next Steps	11
Support.....	11



Overview

This JavaScript decoder provides a universal starting point for integrating the LoRaWAN Radon Sensor (lorawanradonsensor.com) across major LoRaWAN network servers including ChirpStack v4, The Things Network (TTN), and Helium. The decoder extracts the raw radon measurement data from base64-encoded payloads and converts it to hexadecimal format, serving as a foundation for network-specific implementations.

Understanding the Payload

The `hexdata` field contains the hexadecimal representation of the radon sensor's measurement. This raw value represents the radon concentration in Bq/m³ (Becquerels per cubic meter), the SI unit for radioactivity concentration:

The sensor reports **radon** levels directly in Bq/m³

- Multi-byte payloads provide high precision readings
- Additional environmental data may be included alongside radon values

Important: The `hexValue` returned represents the encoded radon level in Bq/m³ that must be extracted according to the sensor's data format specification.

Network Server Integration

ChirpStack v4

Deploy directly in the device profile's codec section. The `decodeUplink()` function matches ChirpStack v4's expected signature:

```
/**
 * ChirpStack v4 Decoder - Converts base64 encoded payload to JSON/hex
 *
 * This decoder function is formatted for ChirpStack v4's expected structure
 */
// Main decode function expected by ChirpStack v4
function decodeUplink(input) {
  const bytes = input.bytes;
  const fPort = input.fPort;
  const variables = input.variables || {};

  try {
    // Convert base64 to hex if it's in the data field
    if (typeof input.bytes !== 'undefined') {
      const hexValue = bytesToHex(bytes);

      return {
        data: {
          hexdata: hexValue
        }
      };
    }
  } else {
    return {
```



```
    errors: ["No bytes provided in the input"]
  },
}

} catch (error) {
  return {
    errors: ["Failed to decode: ${error.message}"]
  };
}
}
}

/**
 * Converts bytes array to hex string
 * @param {Array|Uint8Array} bytes - Byte array
 * @returns {String} - Hex string
 */
function bytesToHex(bytes) {
  let hex = "";
  for (let i = 0; i < bytes.length; i++) {
    const byteHex = bytes[i].toString(16).padStart(2, '0');
    hex += byteHex;
  }
  return hex;
}

// Old function kept for reference or backwards compatibility
function Decode(fPort, bytes, variables) {
  return decodeUplink({fPort: fPort, bytes: bytes, variables: variables}).data;
}

// For testing outside ChirpStack environment
function decodeBase64ToHex(base64String) {
  // Simple base64 decoder for testing
  const binary = atob(base64String);
  const bytes = new Uint8Array(binary.length);
  for (let i = 0; i < binary.length; i++) {
    bytes[i] = binary.charCodeAt(i);
  }
  return bytesToHex(bytes);
}

// Export for Node.js environments (if used outside ChirpStack)
if (typeof module !== 'undefined' && module.exports) {
  module.exports = {
    decodeUplink,
    Decode,
  };
}
```

```
decodeBase64ToHex  
};  
}
```

The Things Network (TTN v3)

Adapt for TTN's formatter by modifying the function signature:

```
javascript  
function decodeUplink(input) {  
  // Core logic remains the same  
  // TTN expects: { data: { radon_raw: hexValue } }  
  // Value represents Bq/m3  
}
```

Helium Console

Implement as a custom function in Helium Console's decoder interface:

```
javascript  
function Decoder(bytes, port) {  
  const hexValue = bytesToHex(bytes);  
  return { radon_hex: hexValue }; // In Bq/m3  
}
```

Extracting Radon Values

Once you have the hex data, extract the actual radon reading. The decoded value will be in Bq/m³:

Example for 2-byte radon value (big-endian):

```
javascript  
// If hexdata = "00FA" (250 in decimal)  
// Radon level = 250 Bq/m3  
const radonBytes = hexdata.substr(0, 4);  
const radonRaw = parseInt(radonBytes, 16);  
const radonBqm3 = radonRaw; // Direct reading in Bq/m3  
  
// Convert to pCi/L if needed (1 pCi/L = 37 Bq/m3)  
const radonPciL = radonBqm3 / 37;
```

Implementation Workflow

1. **Deploy Base Decoder:** Install this decoder in your network server
2. **Identify Data Format:** Review the sensor documentation for:
 - Byte order (little-endian vs big-endian)
 - Scaling factors (if raw value needs multiplication/division)
 - Payload structure and byte positions
 - Confirmation that output is in Bq/m^3
3. **Extend Decoder:** Add parsing logic to extract actual radon values in Bq/m^3
4. **Configure Integrations:** Route decoded data to your monitoring platform

Sensor Data Format

The LoRaWAN Radon Sensor transmits data with:

- Radon concentration values in **Bq/m^3**
- Configurable transmission intervals
- Battery status information
- Optional environmental parameters

Unit Conversion Reference

The decoder outputs values in Bq/m^3 , but you may need to convert for different regional standards:

- **1 Bq/m^3** = 0.027 pCi/L (used in USA)
- **37 Bq/m^3** = 1 pCi/L
- **Safe levels:** <100 Bq/m^3 (WHO recommendation)
- **Action level:** >300 Bq/m^3 (EU reference level) or >148 Bq/m^3 (4 pCi/L USA)



Testing and Validation

The included `decodeBase64ToHex()` function enables testing outside of production environments:

```
javascript
// Test with sample payload
const testPayload = "AKQ="; // Base64
const result = decodeBase64ToHex(testPayload);
// Returns: "00A4" = 164 in decimal = 164 Bq/m3
```

Calculating Long-Term Averages

For efficiency and performance considerations, the sensor payload contains only instantaneous radon measurements, not pre-calculated averages. Long-term average calculations must be implemented in your application layer. This design choice:

- Minimizes payload size for optimal LoRaWAN transmission
- Reduces sensor power consumption
- Provides flexibility in averaging periods and methods

Recommended Implementation Approach

Store each decoded reading in your application database with timestamps, then calculate averages as needed:

```
javascript
// Example application-side averaging logic
function calculateRadonAverages(readings) {
  // 24-hour average
  const dayAvg = calculateAverage(readings, 24);

  // 7-day average (standard for radon assessment)
  const weekAvg = calculateAverage(readings, 24 * 7);

  // 30-day average (monthly trend)
  const monthAvg = calculateAverage(readings, 24 * 30);

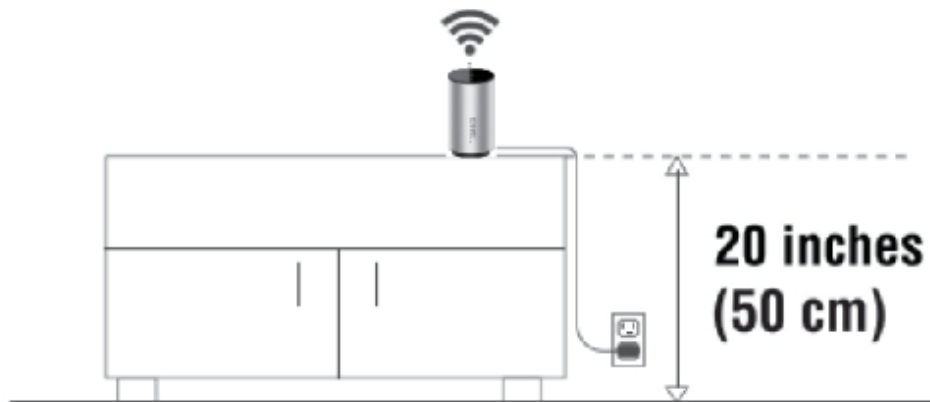
  return {
    current: readings[0].value,
    day_avg: dayAvg,
    week_avg: weekAvg,
    month_avg: monthAvg,
  };
}
```

```
unit: "Bq/m³"  
};  
}
```

Important Averaging Considerations

- 7-Day Average: Industry standard for radon exposure assessment
- Seasonal Variations: Implement year-round monitoring as radon levels fluctuate seasonally
- Data Gaps: Handle missing readings appropriately (interpolation or exclusion)
- Weighted Averages: Consider time-weighted averaging if transmission intervals vary
- Storage Requirements: Plan database capacity for historical data retention

Placement Guidelines



1. Height Requirement: Position the radon monitor at least 20 inches (50 cm) above the ground.
2. Environmental Avoidance: Keep the device away from open windows, vents, and areas with high temperature or humidity.
3. Place in a location shielded from direct sunlight.
4. Vibration Impact: Don't move it during test as Vibrations may cause wrong data,

5. Keep away from high-energy radiation (e.g., X-rays), as exposure may contaminate the sensor cavity

Next Steps

This decoder provides the foundation for radon monitoring integration with values in **Bq/m³**. To complete your implementation:

1. Add the specific parsing logic for extracting radon values from the hexadecimal payload
2. Implement unit conversion if your application requires pCi/L
3. Configure threshold alerts for unsafe radon levels (>300 Bq/m³ EU or >148 Bq/m³ USA)
4. Set up data persistence and visualization in your IoT platform with proper Bq/m³ units display

Support

For detailed payload specifications and advanced integration support for the LoRaWAN Radon Sensor, visit lorawanradonsensor.com



14 rue Edouard Petit - 42000 Saint Etienne
Tél: 04 77 92 03 56 - Email: contact@rg2i.fr
Plus d'infos sur notre site: www.rg2i.fr