Decoder LoRaWAN – temperature&humidity sensor (TTN/Chirpstack)

Decoders · https://www.lansitec.com/pt/docs/decoders/decoder-lorawan-temperaturehumidity-sensor-ttn-chirpstack/

decode-temperature-humidity-ttn_chirpstackv4.jsDownload

Below the full decoder code:

```
// Decode decodes an array of bytes into an object.
//  - fPort contains the LoRaWAN fPort number
//  - bytes is an array of bytes, e.g. [225, 230, 255, 0]
//  - variables contains the device variables e.g. {"calibration": "3.5"} (both the key / value are of type string)
// The function must return an object, e.g. {"temperature": 22.5}

// Temperature Humidity
function decodeUplink(input) {
  // bytes
  var bytes = input.bytes;
  // type
  var uplinkType = (bytes[0] >> 4) & 0x0f;

  switch (uplinkType) {
    case 0x01:
      return { data: decodeRegistration(bytes) };

    case 0x02:
      return { data: decodeHeartbeat(bytes) };

    case 0x0f:
      return { data: decodeAcknowledgment(bytes) };

    default:
      return null;
  }
}

// type: 0x1 Registration
function decodeRegistration(bytes) {
  var data = {};
```

```
data.type = "Registration";
// adr
data.adr = ((bytes[0] >> 3) & 0x1) == 0 ? "OFF" : "ON";
// mode
var modeValue = bytes[0] & 0x07;
if (modeValue == 0x01) {
  data.mode = "AU920";
} else if (modeValue == 0x02) {
  data.mode = "CLAA";
} else if (modeValue == 0x03) {
  data.mode = "CN470";
} else if (modeValue == 0x04) {
  data.mode = "AS923";
} else if (modeValue == 0x05) {
  data.mode = "EU433";
} else if (modeValue == 0x06) {
  data.mode = "EU868";
} else if (modeValue == 0x07) {
  data.mode = "US915";
}
// smode
var smodeValue = bytes[1];
if (smodeValue == 0x01) {
  data.smode = "AU920";
} else if (smodeValue == 0x02) {
  data.smode = "CLAA";
} else if (smodeValue == 0x04) {
  data.smode = "CN470";
} else if (smodeValue == 0x08) {
  data.smode = "AS923";
} else if (smodeValue == 0x10) {
  data.smode = "EU433";
} else if (smodeValue == 0x20) {
  data.smode = "EU868";
} else if (smodeValue == 0x40) {
  data.smode = "US915";
}
// power
data.power = ((bytes[2] >> 3) & 0x1f) + "dBm";
// reserved
var reservedValue = bytes[2] & 0x07;
if (reservedValue == 1) {
  data.frequencySweepMode = "A mode";
} else if (reservedValue == 2) {
```

```
    data.frequencySweepMode = "B mode";
  } else if (reservedValue == 3) {
    data.frequencySweepMode = "C mode";
  } else if (reservedValue == 4) {
    data.frequencySweepMode = "D mode";
  } else if (reservedValue == 5) {
    data.frequencySweepMode = "E mode";
  } else if (reservedValue == 6) {
    data.frequencySweepMode = "All frequency sweep";
  }
  // dr
  data.dr = (bytes[3] >> 4) & 0x0f;
  // repting
  data.repting = ((bytes[3] >> 3) & 0x01) == 0 ? "false" : "true";
  // temperatureReportPeriod
  data.temperatureReportPeriod =
    (((bytes[4] << 8) & 0xff00) | (bytes[5] & 0xff)) * 10 + "s";
  // crc
  data.crc = ((bytes[6] << 8) & 0xff00) | (bytes[7] & 0xff);
  return data;
}

// type: 0x2 Heartbeat
function decodeHeartbeat(bytes) {
  var data = {};
  // type
  data.type = "Heartbeat";
  // version
  data.version = bytes[0] & 0x0f;
  // battery
  data.battery = bytes[1] * 0.1 + "V";
  // rssi
  data.rssi = bytes[2] * -1 + "dBm";
  // snr
  data.snr = (((bytes[3] << 8) & 0xff00) | (bytes[4] & 0xff)) / 100;
  // temperature
  if (0 == ((bytes[5] >> 7) & 0x01)) {
    var tempInt = bytes[5] & 0x7f;
    var tempFra = bytes[6];
    data.temperature = tempInt + "." + tempFra + "□";
  } else {
    var tempInt = (bytes[5] & 0x7f) * -1;
    var tempFra = bytes[6];
    data.temperature = tempInt + "." + tempFra + "□";
```

```
  }
  // humidity
  data.humidity = bytes[7] + "%";
  // crc
  data.crc = ((bytes[8] << 8) & 0xff00) | (bytes[9] & 0xff);

  return data;
}

// type: 0xf Acknowledgment
function decodeAcknowledgment(bytes) {
  var data = {};
  data.type = "Acknowledgment";
  data.result = (bytes[0] & 0x0f) == 0 ? "Success" : "Failure";
  data.msgId = (bytes[1] & 0xff).toString(16).toUpperCase();

  return data;
}
```