

# Capteur Température & Humidité - Dragino LHT65-E1 LoraWAN

<https://www.dragino.com/products/temperature-humidity-sensor/item/151-lht65.html>

## Connexion au Serveur LoRaWAN ChirpStack

Un serveur est maintenu par l'Unistra pour le compte du réseau LoRa de l'Eurométropole de Strasbourg. Le serveur est une instance du serveur de réseau Open Source [Chirpstack](#)

Chirpstack fournit une interface Web pour l'administration des Portails (Gateway), des appareils. L'Interface Web permet également l'interfaçage des données du serveur vers des fournisseurs de Cloud, des bases de données et autres services communément utilisés pour le traitement des données émises par les appareils. Cette interface Web est accessible à l'URL suivante :

<https://inetlab-lorawan.icube.unistra.fr/>

## Créer un nouveau profil pour le capteur

- Sélectionner la rubrique **Devices Profiles**.
- Sélectionner ensuite l'onglet **Général**.
- Donner un nom au profile et configurer le profile LoRa du pour le capteur Dragino LHT65 avec les paramètres suivants :

\* Region

EU868

Region configuration ?

\* MAC version ?

LoRaWAN 1.0.2

\* Regional parameters revision ?

A

\* ADR algorithm ?

Default ADR algorithm (LoRa only)

Flush queue on activate ?

☒

\* Expected uplink interval (secs) ?

3600

Device-status request frequency (req/day) ?

1

## Ajouter le Codec du capteur Dragino LH65

- Télécharger le décodeur Dragino TH65 pour ChirpStack :  
<https://github.com/dragino/dragino-end-node-decoder/tree/main/LHT65N>
- Sélectionner l'onglet **Codec** puis l'option **Java-Script** à partir du menu déroulant et déposé le code du décodeur du capteur :

```
function decodeUplink(input) {
    return {
        data: Decode(input.fPort, input.bytes, input.variables)
    };
}

function Str1(str2){
    var str3 = "";
    for ( var i=0; i<str2.length; i++){
        if ( str2[ i] <= 0x0f){
            str2[ i] = "0" + str2[ i].toString(16) + "";
        }
        str3+= str2[ i].toString(16) + "";
    }
    return str3;
}

function str_pad(byte){
    var zero = '00';
    var hex= byte.toString(16);
    var tmp  = 2-hex.length;
    return zero.substr(0, tmp) + hex + " ";
}
```

```

function datalog(i, bytes){
    var Ext= bytes[ 6] &0x0F;
    var bb;
    if((Ext==' 1') || (Ext==' 9'))
    {
        bb=parseFloat(((bytes[ 0+i]<<24>>16 | bytes[ 1+i])/100).toFixed(2));
    }
    else if(Ext==' 2')
    {
        bb=parseFloat(((bytes[ 0+i]<<24>>16 | bytes[ 1+i])/100).toFixed(2));
    }
    else if(Ext==' 4')
    {
        var Exti_pin_level=bytes[ 0+i] ? "High": "Low";
        var Exti_status=bytes[ 1+i] ? "True": "False";
        bb=Exti_pin_level+Exti_status;
    }
    else if(Ext==' 5')
    {
        bb=bytes[ 0+i]<<8 | bytes[ 1+i];
    }
    else if(Ext==' 6')
    {
        bb=(bytes[ 0+i]<<8 | bytes[ 1+i])/1000;
    }
    else if(Ext==' 7')
    {
        bb=bytes[ 0+i]<<8 | bytes[ 1+i];
    }
    else if((Ext==' 8') || (Ext==' 14'))
    {
        bb=bytes[ 0+i]<<8 | bytes[ 1+i];
    }
    else if(Ext==' 11')
    {
        bb=parseFloat(((bytes[ 0+i]<<24>>16 | bytes[ 1+i])/100).toFixed(2));
    }
    var cc= parseFloat(((bytes[ 2+i]<<24>>16 | bytes[ 3+i])/100).toFixed(2));
    var dd= parseFloat((((bytes[ 4+i]<<8 | bytes[ 5+i])&0xFFF)/10).toFixed(1));
}

```

```

    var ee= getMyDate((bytes[ 7+i]<<24 | bytes[ 8+i]<<16 | bytes[ 9+i]<<8 |
bytes[10+i])).toString(10));
    var string=['+bb+', '+cc+', '+dd+', '+ee+'],'+', ' ';

    return string;
}

function getzf(c_num){
    if(parseInt(c_num) < 10)
        c_num = '0' + c_num;

    return c_num;
}

function getMyDate(str){
    var c_Date;
    if(str > 9999999999)
        c_Date = new Date(parseInt(str));
    else
        c_Date = new Date(parseInt(str) * 1000);

    var c_Year = c_Date.getFullYear(),
    c_Month = c_Date.getMonth()+1,
    c_Day = c_Date.getDate(),
    c_Hour = c_Date.getHours(),
    c_Min = c_Date.getMinutes(),
    c_Sen = c_Date.getSeconds();
    var c_Time = c_Year + '-' + getzf(c_Month) + '-' + getzf(c_Day) + ' ' + getzf(c_Hour) + ':' +
getzf(c_Min) + ':' +getzf(c_Sen);

    return c_Time;
}

function Decode(fPort, bytes, variables) {
var Ext= bytes[ 6]&0x0F;
var poll_message_status=(( bytes[ 6]>>6) &0x01);
var retransmission_Status=(( bytes[ 6]>>7) &0x01);
var Connect=( bytes[ 6] &0x80)>>7;
var decode = {};

```

```

var data = {};
if(( fPort==3) &&(( bytes[ 2] ==0x01)|| ( bytes[ 2] ==0x02)|| ( bytes[ 2] ==0x03)|| ( bytes[ 2] ==0x04) ) ) {
    var array1=[]
    var bytes1="0x"
    var str1=Str1(bytes)
    var str2=str1.substring( 0, 6)
    var str3=str1.substring( 6, )
    var reg=/. {4}/g;
    var rs=str3.match(reg);
    rs.push(str3.substring(rs.join(' ').length));
    rs.pop()
    var new_arr = [...rs]
    var data1=new_arr
    decode.bat=parseInt(bytes1+str2.substring( 0, 4) & 0x3FFF)
    if ( parseInt(bytes1+str2.substring( 4, ))==1){
        decode.sensor="ds18b20"
    }
    else if(parseInt(bytes1+str2.substring( 4, ))==2){
        decode.sensor="tmp117"
    }
    else if(parseInt(bytes1+str2.substring( 4, ))==3){
        decode.sensor="gxht30"
    }
    else if(parseInt(bytes1+str2.substring( 4, ))==4){
        decode.sensor="sht31"
    }
    for ( var i=0; i<data1.length; i++){
        var temp=( parseInt(bytes1+data1[ i].substring( 0, 4) ) ) /100
        array1[ i]=temp
    }
    decode.Temp=array1
    {
        return decode;
    }
}
else if( fPort==5)
{
    var sub_band;
    var freq_band;

```

```
var sensor;

if( bytes[ 0]==0x0B)
    sensor= "LHT65N";
else if( bytes[ 0]==0x1A)
    sensor= "LHT65N-PIR";

if( bytes[ 4]==0xff)
    sub_band="NULL";
else
    sub_band=bytes[ 4];

if( bytes[ 3]==0x01)
    freq_band="EU868";
else if( bytes[ 3]==0x02)
    freq_band="US915";
else if( bytes[ 3]==0x03)
    freq_band="IN865";
else if( bytes[ 3]==0x04)
    freq_band="AU915";
else if( bytes[ 3]==0x05)
    freq_band="KZ865";
else if( bytes[ 3]==0x06)
    freq_band="RU864";
else if( bytes[ 3]==0x07)
    freq_band="AS923";
else if( bytes[ 3]==0x08)
    freq_band="AS923_1";
else if( bytes[ 3]==0x09)
    freq_band="AS923_2";
else if( bytes[ 3]==0x0A)
    freq_band="AS923_3";
else if( bytes[ 3]==0x0B)
    freq_band="CN470";
else if( bytes[ 3]==0x0C)
    freq_band="EU433";
else if( bytes[ 3]==0x0D)
    freq_band="KR920";
else if( bytes[ 3]==0x0E)
```

```

    freq_band="MA869";

var firm_ver= (bytes[1] &0x0f) +'.' +( bytes[2] >>4&0x0f) +'.' +( bytes[2] &0x0f);
var bat= (bytes[5] <<8 | bytes[6])/1000;

return {
    SENSOR_MODEL: sensor,
    FIRMWARE_VERSION: firm_ver,
    FREQUENCY_BAND: freq_band,
    SUB_BAND: sub_band,
    BAT: bat,
};
}
if (retransmission_Status==0)
{
switch (poll_message_status) {[]
case 0: []
{
if(Ext==0x09)
{
decode.TempC_DS=parseFloat(((bytes[0] <<24>>16 | bytes[1])/100).toFixed(2));
decode.Bat_status=bytes[4] >>6;
}
else
{
decode.BatV= ((bytes[0] <<8 | bytes[1]) & 0x3FFF)/1000;
decode.Bat_status=bytes[0] >>6;
}

if(Ext! =0x0f)
{
decode.TempC_SHT=parseFloat(((bytes[2] <<24>>16 | bytes[3])/100).toFixed(2));
decode.Hum_SHT=parseFloat((((bytes[4] <<8 | bytes[5]) &0xFFF)/10).toFixed(1));
}
if(Connect=='1')
{
decode.No_connect="Sensor no connection";
}
}

```

```
if(Ext==' 0' )
{
    decode.Ext_sensor ="No external sensor";
}
else if(Ext==' 1' )
{
    decode.Ext_sensor ="Temperature Sensor";
    decode.TempC_DS=parseFloat(((bytes[ 7]<<24>>16 | bytes[ 8])/100).toFixed(2));
}
else if(Ext==' 2' )
{
    decode.Ext_sensor ="Temperature Sensor";
    decode.TempC_TMP117=parseFloat(((bytes[ 7]<<24>>16 | bytes[ 8])/100).toFixed(2));
}
else if(Ext==' 4' )
{
    decode.Work_mode="Interrupt Sensor send";
    decode.Exti_pin_level=bytes[ 7] ? "High": "Low";
    decode.Exti_status=bytes[ 8] ? "True": "False";
    decode.Exit_count= bytes[ 9]<<16 | bytes[ 10]<<8 | bytes[ 11];
    decode.Exit_duration= bytes[ 12]<<16 | bytes[ 13]<<8 | bytes[ 14];
}
else if(Ext==' 5' )
{
    decode.Work_mode="Illumination Sensor";
    decode.ILL_lx=bytes[ 7]<<8 | bytes[ 8];
}
else if(Ext==' 6' )
{
    decode.Work_mode="ADC Sensor";
    decode.ADC_V=( bytes[ 7]<<8 | bytes[ 8])/1000;
}
else if(Ext==' 7' )
{
    decode.Work_mode="Interrupt Sensor count";
    decode.Exit_count=bytes[ 7]<<8 | bytes[ 8];
}
else if(Ext==' 8' )
{

```



```

    decode.Work_mode="Interrupt Sensor count";
    decode.Exit_count=bytes[ 7]<<24 | bytes[ 8]<<16 | bytes[ 9]<<8 | bytes[ 10];
}
else if(Ext==' 9' )
{
    decode.Work_mode="DS18B20 & timestamp";
    decode.Systimestamp=(bytes[ 7]<<24 | bytes[ 8]<<16 | bytes[ 9]<<8 | bytes[ 10] );
}
else if(Ext==' 11' )
{
    decode.Work_mode="SHT31 Sensor";
    decode.Ext_TempC_SHT=parseFloat(((bytes[ 7]<<24>>16 | bytes[ 8])/100).toFixed(2));
    decode.Ext_Hum_SHT=parseFloat((((bytes[ 9]<<8 | bytes[ 10])&0xFF)/10).toFixed(1));
}
else if(Ext==' 14' )
{
    decode.Work_mode="PIR Sensor";
    decode.Exti_pin_level=bytes[ 7] & 0x01 ? "Activity": "No activity";
    decode.Move_count=bytes[ 8]<<16 | bytes[ 9]<<8 | bytes[ 10];
}
else if(Ext==' 15' )
{
    decode.Work_mode="DS18B20ID";

    decode.ID=str_pad( bytes[ 2] )+str_pad( bytes[ 3] )+str_pad( bytes[ 4] )+str_pad( bytes[ 5] )+str_pad( bytes[ 7] )+str_pad( bytes[ 8] )+str_pad( bytes[ 9] )+str_pad( bytes[ 10] );
}
}

if(( bytes.length==11)|| ( bytes.length==15))
{
    return decode;
}
break;

case 1:
{
    for( var i=0; i<bytes.length; i=i+11)
    {
        var da= datalog(i,bytes);
    }
}

```

```

        if(i==' 0' )
            decode.DATALOG=da;
        else
            decode.DATALOG+=da;
    }
}
{
return decode;
}
break;
default:
    return {
        errors: ["unknown"]
    }
}
}
else
{
switch (retransmission_Status) {
    case 1:
    {
        for( var i=0; i<bytes.length; i=i+11)
        {
            var da= datalog(i,bytes);
            if(i==' 0' )
                data.retransmission_message=da;
            else
                data.retransmission_message+=da;
        }
    }
}
{
return data;
}
break;
default:
    return {
        errors: ["unknown"]
    }
}
}

```

```
}
```

```
}
```

## Créer l'application (si nécessaire)

- Sélectionner la rubrique **Applications**.
- Cliquer sur le bouton ADD Application.
- Saisir le nom de l'application et sa description.

## Créer un appareil dans une application

- Sélectionner l'application
- Cliquer sur le bouton **ADD device** pour ajouter un appareil.
- Saisir le nom de l'appareil et sa description.
- Saisir le numéro **EUI de l'appareil** (capteur commercial) ou le générer de manière aléatoire (capteur fait maison).

---

Revision #3

Created 29 January 2024 15:48:32 by admin\_idf

Updated 29 January 2024 17:55:44 by admin\_geii