



**Implementing a
Flood Management System
using The Things Network (TTN) and Google
Sheets**

Table of Contents

Table of Contents	2
Implementing a Flood Management System using The Things Network (TTN) and Google Sheets	3
1. Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor Overview	4
2. Sensor Configuration using Android App and Near Field Communication (NFC)	6
3. The Things Network (TTN), and adding Sensors.	8
4. Google Sheets to Receive Sensor Data	15
5. Configuring TTN Web Hook, and Plotting Data	21
6. Final Steps: Staging/Test Deployment	24
Appendix 1: Overview of LoRaWAN Spread Factor	27
Appendix 2: Elsys Battery Life Calculator	28
Appendix 3: Adding A LoRaWAN Gateway to The Things Network (TTN)	29
Appendix 4: Adding Decentlab Sensors to The Things Network (TTN)	31
Decentlab DL-MBX Ultrasonic Sensor Configuration	31
Appendix 5: Google Sheets code for Decentlab DL-MBX Ultrasonic Sensor	36
Appendix 6: Things Network - How to add a device to the LoRaWAN network	42
Additional YouTube Videos	42
Appendix 7: GitHub Link	43
References	44

Implementing a Flood Management System using The Things Network (TTN) and Google Sheets

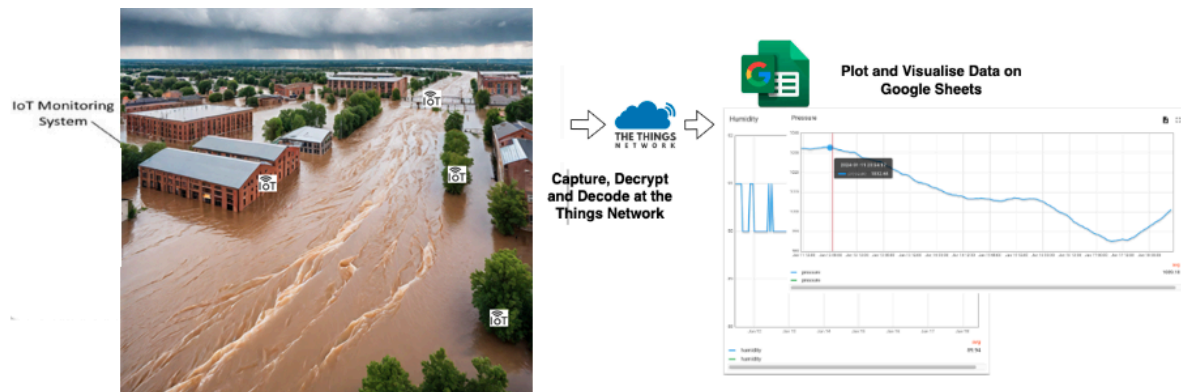


Figure 1: Proposed Low Cost Sustainable Flood Management System

The primary objective of this tutorial is to document the deployment of a low cost practical and sustainable flood management system using LoRaWAN technologies. This system will leverage freeware technologies to gather data from Decentlab Flood Management LoRaWAN IoT sensors, and plot the data collected. By documenting the process for provisioning LoRaWAN sensors using the Things Network, decoding the acquired data using JavaScript, and visualising the extracted results on Google Sheets, our objective is to facilitate the replication of a low cost flood management system in diverse geographical locations particularly by citizen scientists and small community groups. This tutorial has been prepared in cooperation with the Geography Department in Trinity College Dublin.

Please be aware that although we strive to offer user-friendly documentation for the Flood Management System, implementing it may necessitate some technical expertise. Additionally, as systems evolve through upgrades and other changes outside of our control, occasional technical debugging may be necessary.

Scope of Technical Work and Documentation

This system will leverage The Things Network (TTN) Sandbox for capturing data from Decentlab LoRaWAN sensors. As a free version of The Things Network, a decentralised LoRaWAN network, the Things Network Sandbox is designed for non-commercial small-scale experimentation, accommodating up to 10 gateways and 10 sensors per account. This resource is freely accessible to The Things Network community[7], offering an ideal platform for citizen scientists and community groups to initiate testing of sensor devices, applications, and integrations while capturing useful data to support Flood Management activities. To support data storing and visualisation, we will employ Google Sheets. Google Sheets, a web-based spreadsheet application developed by Google, facilitates real-time collaboration, enabling easy sharing of flood datasets across community groups. A high-level overview of the proposed system is shown in Figure 1.

1. Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor

Overview

The *Elsys ERS Maxbotix Ultrasonic Level Sensor*, shown in Figure 2, is a reliable solution for monitoring fill levels. It combines the Elsys ELT-2 HP with the Maxbotix Ultrasonic sensor, allowing it to measure levels up to a distance of 5 metres. Here are some key features of this sensor[1][2][3][4]:

- LoRaWAN® Certified: Ensures compatibility with LoRaWAN networks.
- Adjustable Mounting Options: Can be installed in various environments.
- Measures Ambient Temperature & Humidity: Provides additional environmental data.
- Detects Acceleration: Useful for applications requiring motion sensing.
- IP67 Rated: Robust protection against dust and water.
- Battery-Powered: Operates on a single 3.6V AA lithium battery.
- NFC for Configuration: Convenient setup using Near Field Communication.
- Configuration Over the Air: Remotely configure alerts for changes in fill levels.

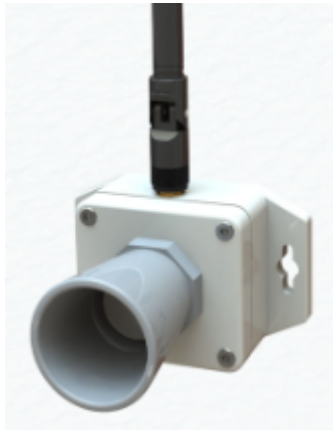


Figure 2: Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor

Applications for the Elsys Maxbotix Ultrasonic Level Sensor include:

- Bin and Tank Levels
- Water Level Measurements (e.g., under bridges or wells)
- Smart Towns & Cities
- Agricultural Environments
- Monitoring Extreme Weather Conditions and Flood Alerting

The sensor transmits recorded data to a connected LoRaWAN gateway, which then relays it to your chosen dashboard (Google Sheets in our case) via a LoRaWAN network server (The Things Network Sandbox). Our chosen dashboard for this project is Google Sheets.

Things to note about this sensor

Ultrasonic Distance Sensor: Distance Range 30 cm – 5 metres. Note, targets closer than 30 cm will measure a distance of 30 cm.

Type Value	Type	Data size	Comment
0x01	Temperature	2	-3276.5 °C – 3276.5 °C (Value of: 100 → 10.0 °C)
0x02	Humidity	1	0 – 100 %
0x07	VDD (Battery voltage)	2	0 – 65535 mV
0x0E	External distance	2	0 – 65535 mm
0x0F	Motion (Acceleration)	1	0 – 255 (Interrupts from accelerometer)
0x14	Pressure	4	Pressure data (hPa)
0x3D	Debug information	4	Data depends on debug information
0x3E	Sensor settings	n	Sensor setting sent to server at startup (first package). Sent on Port+1.

Table 1: Elsys ELT Ultrasonic Level Sensor Data Types[3]

For more technical details, you can refer to the Elsys ELT Ultrasonic Level Sensor Datasheet, website and manual[1][2][3][4].

2. Sensor Configuration using Android App and Near Field Communication (NFC)

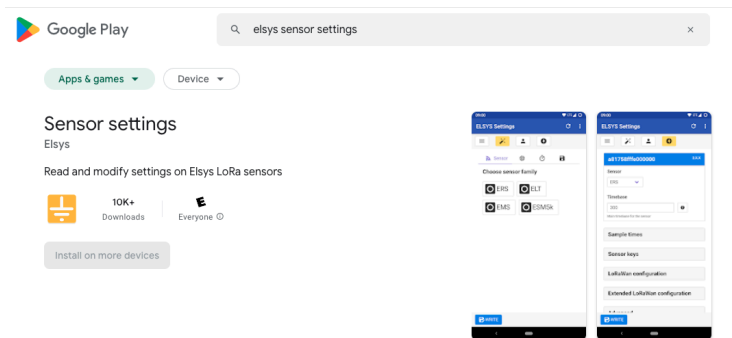


Figure 3: Sensor settings app on Google Android Store.

All sensor settings can be configured via a smartphone application with Near Field Communication (NFC). The sampling rate, spreading factor, encryption keys, port, and modes can all be viewed and changed if required using this app.



Figure 4: Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor

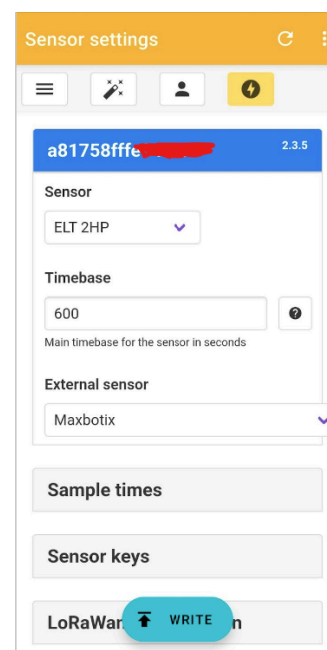


Figure 5: Sensor Settings Android App

Download ELSYS “Sensor Settings”

1. Download ELSYS “Sensor Settings” [5] application from Google Play or App Store (from iOS 13) and install it on a smartphone or tablet, see Figure 3. The device (mobile or tablet) must support NFC communication.
2. Enable NFC on the device and start the “Sensor Settings” application.

3. Place your Android / iOS device on top of the Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor, see Figure 4.
4. Remove the device. Current settings will be displayed in the application, see Figure 5.
5. Use the application to change any settings if required. For example, in our initial Flood site deployments, we have set the Timebase Setting to 1800 seconds (30minutes x 60 seconds) - representing 30 minutes. A sensor measurement will be recorded and transmitted every 30 minutes onto the LoRaWAN network. A definition of Timebase is shown in Figure 6. At this juncture, it's crucial to document the Sensor Keys Parameters shown in Figure 7, namely the AppEUI and AppKey. Both parameters are essential for installing the sensor on the Things Network.
6. Tap the device on top of the NFC antenna to give the new settings to the sensor. For example, if you were to set the Timebase Setting to 1800 seconds (30 minutes). Make sure that the application confirms your new settings.
7. Wait for the sensor to reboot (5 sec), indicated by the LED flashing. Sensor settings have been updated.

Please refer to the Elsys Operating Manual ELT Ultrasonic [2] for additional information if needed.

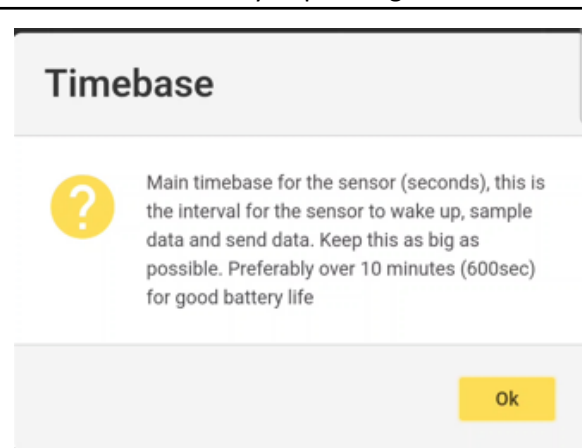


Figure 6: Sensor Settings App - definition of TimeBase setting.

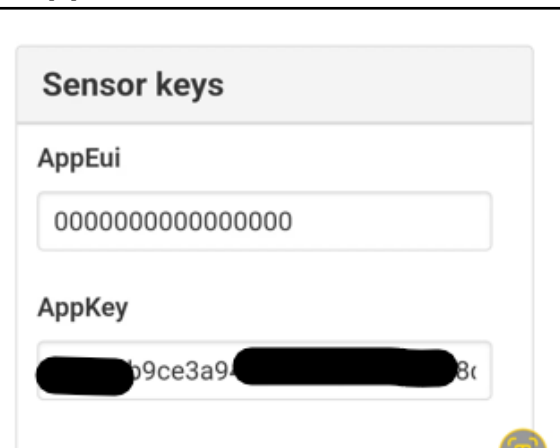


Figure 7: Take note of the AppEUI, and AppKey parameters from the Sensor. These are needed when installing the Sensor on the Things Network.

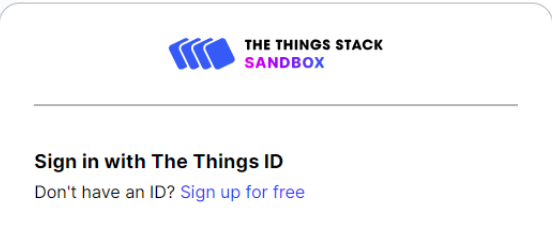
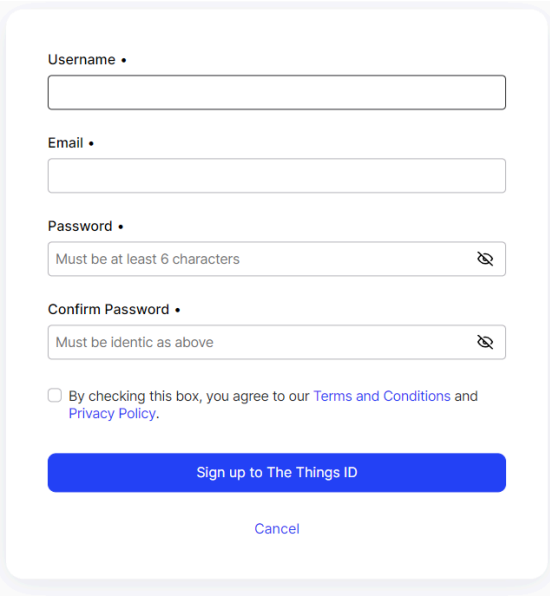
3. The Things Network (TTN), and adding Sensors.

As discussed above, this system will leverage the Things Network Sandbox for capturing data from Decentlab LoRaWAN sensors.

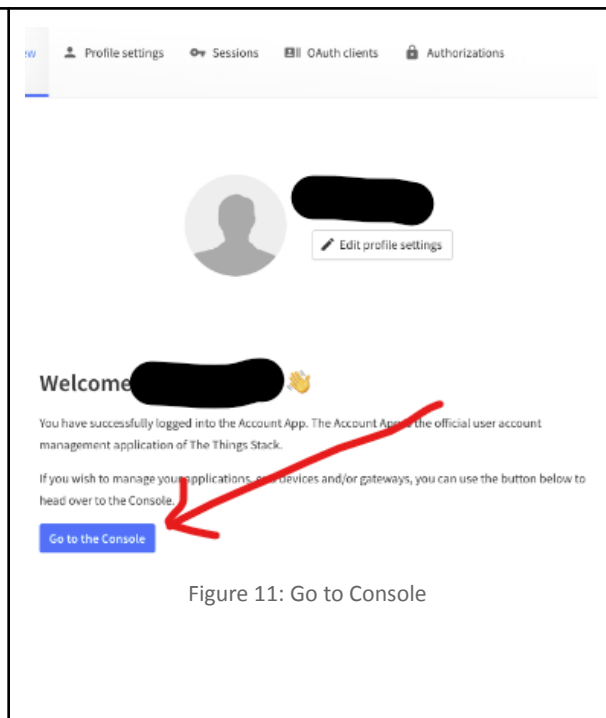
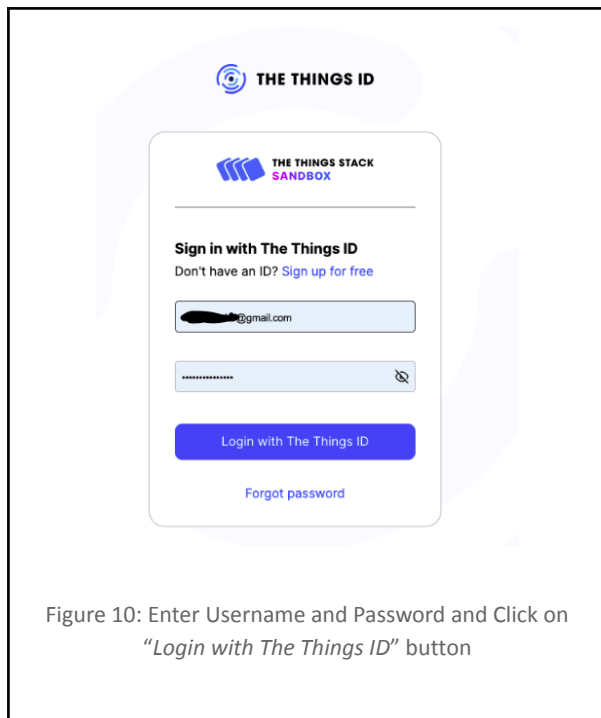
Go to the following Things Network URL and log in [7]:

```
https://eu1.cloud.thethings.network/oauth/login
```

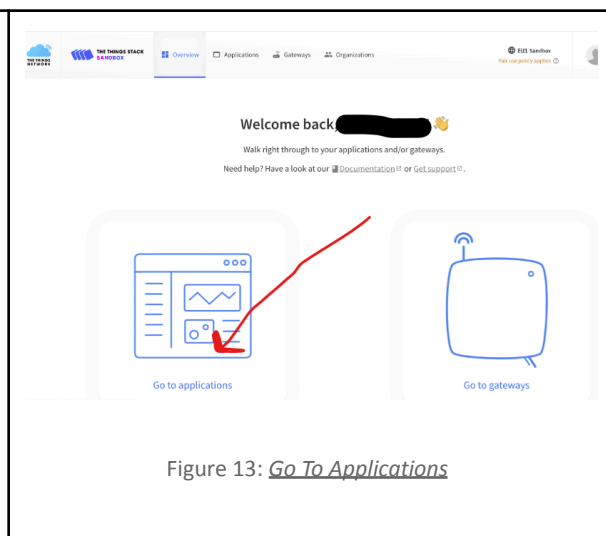
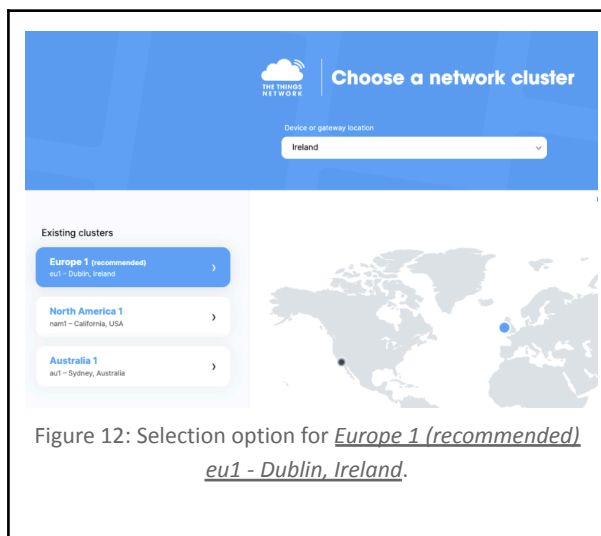
Note, if you do not currently have a Things Network sandbox account, a new account can be created easily by clicking on the “[Sign up for Free](#)” link, and entering a username, email address, and password as shown in Figure 8 and Figure 9.

 <p>Figure 8: Sign up for free for a Things Network account</p>	 <p>Figure 9: Data required to create a Things Network account</p>
---	---

Enter your Things ID (username) and Password. Click on the “[Login with The Things ID](#)” button, see Figure 10. Then Click the “[Go To Console](#)” button, as shown on Figure 11.



In the next menu, "Choose a network Cluster", See Figure 12, select Ireland. Then choose the Existing clusters option for Europe 1 (recommended) eu1 - Dublin, Ireland, See Figure 12.



In the next Screen shown in Figure 13, click Go to applications and then Create Application, see Figure 14.

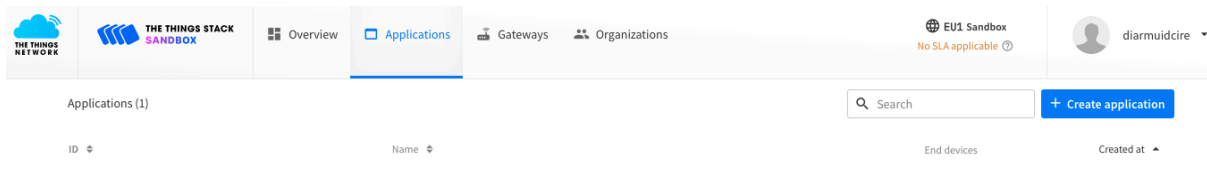


Figure 14: Create application

In the Create Application screen, enter information about the Application ID, Application Name, and Project Description. Then Click Create Application. Example shown in Figure 15. In our case, each new Application is a flood site. We recommended associating the application ID with a particular flood location (e.g., south-wexford-flood-site-1).

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 *
flood-site-location-1

Application name
Flood Site Location 1

Description
Sample Flood Site Location 1

Optional application description; can also be used to save notes about the application

[Create application](#)

Figure 15: Create an application.

It is now possible to start registering *Elsys ELT-2 HP Ultrasonic* Sensors with this Application (or Flood Site) defined in Figure 15 - flood-site-location-1. Click on End Devices, and click Register end Device, as shown in Figure 16.

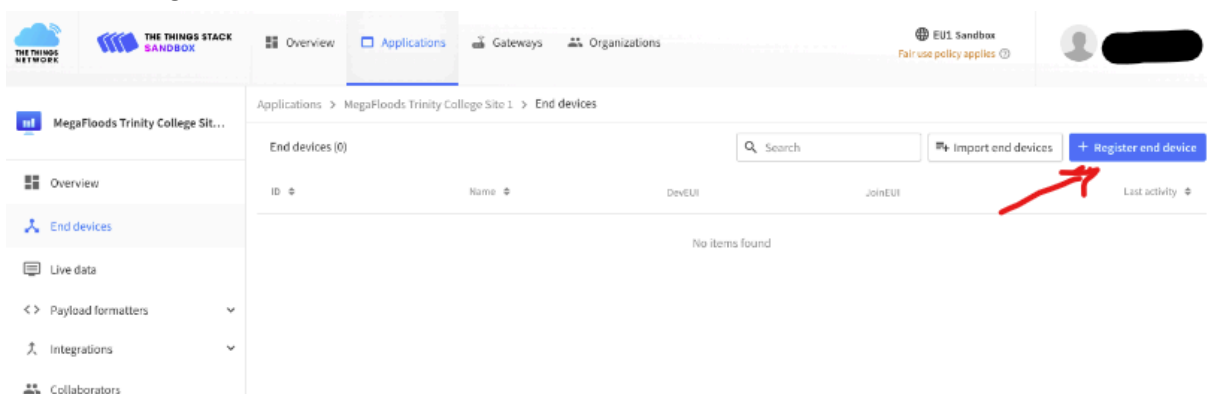


Figure 16: Register End Device

We are adding the *Elsys ELT-2 HP with the Maxbotix Ultrasonic* Sensor to the Things Network. Select the option Select the end device in the LoRaWAN Device Repository as shown in Figure 17.

Applications > MegaFloods Trinity College Site 1 > End devices

Register end device

Does your end device have a LoRaWAN® Device Identification QR Code? Scan it to speed up onboarding.

[Scan end device QR code](#) [Device registration help](#)


End device type

Input method

☒ Select the end device in the LoRaWAN Device Repository

☐ Enter end device specifics manually

End device brand: Model: Hardware Ver.: Firmware Ver.: Profile (Region):



ELT Ultrasonic LoRaWAN Distance Sensor
LoRaWAN Specification 1.0.3, RP001 Regional Parameters 1.0.3 revision A, Over the air activation (OTAA), Class A

The ELT Ultrasonic is designed for measuring the distance. It can be placed in various environments and can work as a bin level sensor, tank level sensor, or water level under bridges, for example.

[Product website](#) | [Data sheet](#)

Frequency plan

Figure 17: Parameters to select when registering the *Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor* on the Things Network.

The following parameters shown in Table 2 need to be specified when adding the *Elsys ELT-2 HP with the Maxbotix Ultrasonic Sensor* to the Things Network.

TTN Label	Sample ELT Sensor Parameter
End Device Brand	Elsys
Model	ELT Ultrasonic LoRaWAN Distance Sensor
Hardware Version	Default (1.0)
Firmware Version	Default (1.0)
Profile (Region)	EU_863_870
Frequency Plan	Europe 863-870 MHz (SF12 for RX2) ** Detailed explanation about this parameter in Appendix 1.
JoinEUI (A JoinEUI, also called the AppEUI, is a 64 bit extended unique identifier used to identify the Join Server During Activation - Retrieved from the Sensor Settings App in Section 2)	0000000000000000

DevEUI (A 64-bit extended unique identifier for your end device - Retrieved from the Sensor Settings App in Section 2)	A81758FFFB3D57ED
AppKey (An end device specific encryption key used during ITAA in LoRaWAN - Retrieved from the Sensor Settings App in Section 2)	DD353652A69850365894E9C5EC4CC9A9
End Device ID (a unique human-readable identifier for your end device)	s-wex-sensor-1

Table 2: Parameters of an Elsys ELT-2 HP with Maxbotix Ultrasonic Sensor for Integration with the Things Network

It is important to highlight the significance of the LoRaWAN spread factor parameter, which, as indicated in Table 2 on the ELT-2 parameters table, aligns with the Frequency Plan Europe 863-870 MHz (SF12 for RX2) in the Things Network. In summary, as elaborated in Appendix 1, the LoRaWAN spread factor (SF) determines whether the sensor operates in close proximity to the gateway or at a distance. Use Spread Factor 7 if the sensor is close to the gateway, as this setting prolongs battery life and supports more sensor transmissions. If the sensor is far away from the gateway and there are anticipated environmental factors such as tree cover, foliage, etc, use Spread Factor 12 to ensure the transmission is successful. SF12 uses more battery power resulting in the battery not lasting as long. Appendix 2 provides information about estimating the Battery Life of an Elsys Sensor.

Applications > Flood Site Location 1 > End devices

Register end device

Does your end device have a LoRaWAN® Device Identification QR Code? Scan it to speed up onboarding.


[Scan end device QR code](#) [Device registration help](#)

End device type

Input method

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

End device brand Model Hardware Ver. Firmware Ver. Profile (Region)



ELT Ultrasonic LoRaWAN Distance Sensor
LoRaWAN Specification 1.0.3, RP001 Regional Parameters 1.0.3 revision A, Over the air activation (OTAA), Class A

The ELT Ultrasonic is designed for measuring the distance. It can be placed in various environments and can work as a bin level sensor, tank level sensor, or water level under bridges, for example.

[Product website](#) [Data sheet](#)

Frequency plan

[Register end device](#)

Figure 18: Configure the ELT Ultrasonic Sensor According to Table 2.

Frequency plan

Provisioning information

JoinEUI [Reset](#)

This end device can be registered on the network

DevEUI [Generate](#) 0/50 used

AppKey [Generate](#)

End device ID

This value is automatically prefilled using the DevEUI

After registration

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

[Register end device](#)

Figure 19: Configure the ELT Ultrasonic Sensor According to Table 2 (continued).

When all the parameters have been entered, click on [Register end Device](#), shown in Figure 18 and Figure 19. Assuming the device is registered correctly, you should see a similar screen as shown in Figure 20.

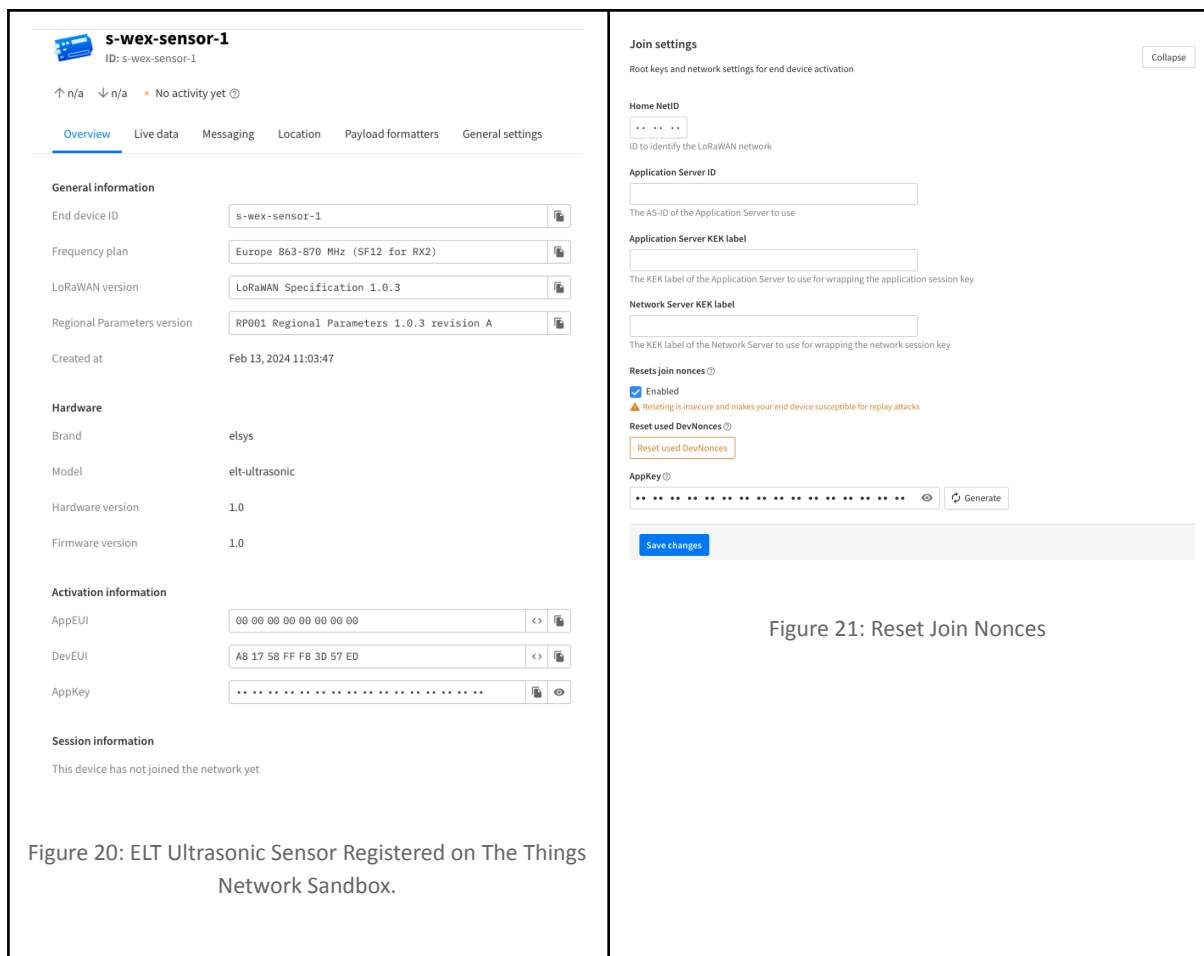


Figure 20: ELT Ultrasonic Sensor Registered on The Things Network Sandbox.

Figure 21: Reset Join Nonces

Note, based on experience, it can be useful to edit the device, and Enable the Resets join Nonces parameter as shown in Figure 21. The Nonce, included within the message payload, fulfils the essential roles of upholding message integrity and preventing replay attacks in LoRaWAN transmissions. Should a LoRaWAN sensor transmission, particularly from sensors distant from gateways, fail to reach the gateway, it can potentially disrupt subsequent sensor communications with the LoRaWAN network. This occurs because the gateway anticipates the next nonce, which was lost in the previous communication, thus compromising the verification of message integrity and prompting the gateway to ignore all future messages from the sensor. This is why we disable the integrity functionality.

To Enable the Resets join Nonces parameter, click the General Settings tab, Shown in Figure 20, and expand Join Settings, shown in Figure 21. Tick the Enabled box under Resets join Nonces.

At this point, allow the sensor to remain inactive for a few hours. Initially, you may observe no activity from the sensor as it requires time for the Things Network to recognize the sensor is provisioning on the network. This factor is beyond our control, as it's related to the Things Network. If you have a Gateway provisioned against your account, you may observe activity similar to that shown in Figure 22, where JoinEUI requests are issued by the sensor. If you lack access to a gateway, you will need to be patient and hope your sensor is within range of a community gateway. You can refer to the list of community gateways on the Things Network Map [8].

north-wexford-thnp-multitech > Gateways > Live data			
Receive uplink message		Data preview	
time	type	Verbose stream <input type="checkbox"/> Export as JSON <input type="checkbox"/> Pause <input type="checkbox"/> Clear <input type="button" value="Clear"/>	
↑ 15:58:08	Receive uplink message	JoinEUI: 32 A8 CB 90 00 00 00 11	DevEUI: A8 17 58 FF FE 04 CE 96 Data rate: SF11BW125 SNR: 11
↑ 15:57:52	Receive uplink message	JoinEUI: 32 A8 CB 90 00 00 00 11	DevEUI: A8 17 58 FF FE 04 CE 96 Data rate: SF7BW125 SNR: 8

Figure 22: Sensor Join Requests received by the Gateway.

When the Sensor Successfully joins the network (*this might take a few hours*), you will see something similar to Figure 23. At this point sensor data will start to be automatically decoded, as shown in Figure 24. Note sensor measurements for Distance, Humidity, Pressure, Temperature, etc.

Overview

Applications

Gateways

Organizations

EUI Sandbox

No support plan

diarmuidcire

MegaFloods Trinity College Sit...

Applications > MegaFloods Trinity College Site 1 > Application data

Time

Entity ID

Type

Data preview

Verbose stream

Export as JSON

Pause

Clear

Overview

End devices

Live data

Payload formatters

Integrations

Collaborators

API keys

General settings

16:50:36

eui-a81758fffe04c...

Update end device

["activated_at"]

16:50:36

eui-a81758fffe04c...

Forward uplink data message

DevAddr: 26 08 74 6A

Payload: { }

16:50:32

eui-a81758fffe04c...

Forward join-accept message

DevAddr: 26 08 74 6A

16:50:31

eui-a81758fffe04c...

Successfully processed joi...

16:50:30

eui-a81758fffe04c...

Accept join-request

DevAddr: 26 08 74 6A

16:50:30

Console: Events cleared

The events list has been cleared

Figure 23: Sensor successfully Joins the Things Network LoRaWAN Server.

↑ 17:12:23	eui-a81758fffe04ce96	Forward uplink data message	DevAddr: 26 08 33 67	Payload: { distance: 1215, humidity: 67, pressure: 1008.92, temperature: 20
↑ 17:10:23	eui-a81758fffe04ce96	Forward uplink data message	DevAddr: 26 08 33 67	Payload: { distance: 1682, humidity: 66, pressure: 1008.84, temperature: 20
17:09:33		Console: Events cleared	The events list has been cleared	

Figure 24: Sensor Payload information decoded by the Things Network Server

4. Google Sheets to Receive Sensor Data

We will employ Google Sheets for visualising data collected from the Things Network. Google Sheets, a web-based spreadsheet application developed by Google, facilitates real-time collaboration, enabling easy sharing and graphing of flood datasets across community groups. An overview of Google Sheets is outside the scope of this tutorial. There are, however, plenty of tutorials available online [9].

To prepare Google Sheets to receive Things Network Sensor readings, we need to create a Google Sheets Apps Script. Here are the steps to get started:

1. Open your Google Sheets document.
2. Click on Extensions > Apps Script.
3. Delete any existing code in the script editor.
4. Replace it with the following sample code. Note, if you are collecting different sensor parameters then these: 'dev_eui', 'distance', 'humidity', 'pressure', 'temperature', 'vdd', 'date_time', 'current_time', then you will need to update this code. Code available on the following GitHub link[15]: <https://github.com/dcollin5/ttn-google-sheet-integration>

```
//  
// Create or open an existing Sheet and click Tools > Script editor and  
enter the code below  
// 1. Run > setup  
// 2. Publish > Deploy as web app  
//    - enter Project Version name and click 'Save New Version'  
//    - set security level and enable service (most likely execute as  
'me' and access 'anyone, even anonymously')  
// 3. Copy the 'Current web app URL' and post this in your form/script  
action in the Things Network  
  
var SCRIPT_PROP = PropertiesService.getScriptProperties(); // new  
property service  
  
// If you don't want to expose either GET or POST methods you can comment  
out the appropriate function  
//function doGet(e){  
//  return handleResponseV2(e);  
//}
```

```
//ttn uses doPost function
function doPost(e){
    return handleResponseV2(e);
}

function currentTime(){
    // Create a new Date object
    var currentTime = new Date();

    // Get the current hour, minute, and second
    var currentHour = currentTime.getHours();
    var currentMinute = currentTime.getMinutes();
    var currentSecond = currentTime.getSeconds();

    // Format the time nicely
    var formattedTime = currentHour + ":" + currentMinute + ":" +
currentSecond;

    // Display the current time
    console.log("The current time is: " + formattedTime);
    return formattedTime;
}

function currentDateUSA() {
    // Create a new Date object
    var currentTime = new Date();

    // Get the current month, day, and year
    var currentMonth = currentTime.getMonth() + 1; // Add 1 because
getMonth() returns zero-based values
    var currentDay = currentTime.getDate();
    var currentYear = currentTime.getFullYear();
}
```



```

// Format the date
var formattedDate = currentMonth + '/' + currentDay + '/' + currentYear;
return formattedDate;
}

function dateTime() {
var currentdate = new Date();
var datetime = "Last Sync: " + currentdate.getDate() + "/"
    + (currentdate.getMonth()+1) + "/"
    + currentdate.getFullYear() + " @ "
    + currentdate.getHours() + ":"
    + currentdate.getMinutes() + ":"
    + currentdate.getSeconds();
return currentdate;
}

function handleResponseV2(e) {
var lock = LockService.getPublicLock();
lock.waitLock(30000); // wait 30 seconds before conceding defeat.
try {
    //enable the following line for testing.
    //var jsonData = flatten(jsonData2);
var jsonData = flatten(JSON.parse(e.postData.contents));
var devEui = jsonData.dev_eui;

// Create or open sheet based on dev_eui
var doc = SpreadsheetApp.getActiveSpreadsheet();
var sheet = doc.getSheetByName(devEui);
if (!sheet) {
    sheet = doc.insertSheet(devEui);
    // Set headers if the sheet is newly created
    sheet.appendRow(['dev_eui', 'distance', 'humidity', 'pressure',
'temperature', 'vdd', 'date_time', 'current_time', 'usa_date']);

```

```

    }

    var nextRow = sheet.getLastRow() + 1; // get next row

    // Extract data from JSON object
    var rowData = [
        jsonData.dev_eui,
        jsonData.distance,
        jsonData.humidity,
        jsonData.pressure,
        jsonData.temperature,
        jsonData.vdd,
        dateTime(),
        currentTime(),
        currentDateUSA(),
        // Add more parameters as needed
    ];

    // Append data to the next empty row in the spreadsheet
    sheet.appendRow(rowData);

    // return json success results
    return ContentService
        .createTextOutput(JSON.stringify({"result":"success", "row":
nextRow}))
        .setMimeType(ContentService.MimeType.JSON);
    } catch(e) {
        // if error return this
        console.error(e);
        return ContentService
            .createTextOutput(JSON.stringify({"result":"error", "error":
e}))
            .setMimeType(ContentService.MimeType.JSON);
    } finally { //release lock

```

```

    lock.releaseLock();
  }
}

function flatten(obj) {
  var flattened = {};
  Object.keys(obj).forEach(function(key) {
    if (typeof obj[key] === 'object' && obj[key] !== null) {
      Object.assign(flattened, flatten(obj[key]));
    } else {
      flattened[key] = obj[key];
    }
  });
  return flattened;
}

```

5. Save your script by clicking the Save button.
6. Rename your project by clicking on "Untitled project" and typing a new name (e.g., "FloodMgt-v1"). Then Click the "Deploy" button, and "New Deployment", see Figure 25.

The image shows a 'New deployment' dialog in Google Cloud Platform. On the left, the 'Web app' tab is selected. The 'Description' field is filled with 'FloodMgt-v1'. Under 'Web app', the 'Execute as' dropdown is set to 'Me (@gmail.com)'. Below that, the 'Who has access' dropdown is set to 'Anyone'. At the bottom of the dialog are 'Cancel' and 'Deploy' buttons. To the right of the dialog, there is a warning box titled 'Make sure you trust Untitled project' with the text: 'You may be sharing sensitive info with this site or app. You can always see or remove access in your [Google Account](#). Learn how Google helps you [share data safely](#). See Untitled project's Privacy Policy and Terms of Service.' Below this text are 'Cancel' and 'Allow' buttons.

Figure 25: New Deployment

Figure 26: Authorization Required, review permissions and Allow.

7. Click Deploy.
8. Select Type: "Web App", See Figure 25. Enter Description, and set the "Who has access" to Anyone. This will enable the Things Network to forward data to the worksheet for plotting.
Note, every time you change the code, you will need to redeploy the application.
9. The first time you Deploy an App you'll be prompted to authorise access. Click Review permissions, choose an account, and click Allow, see Figure 26.

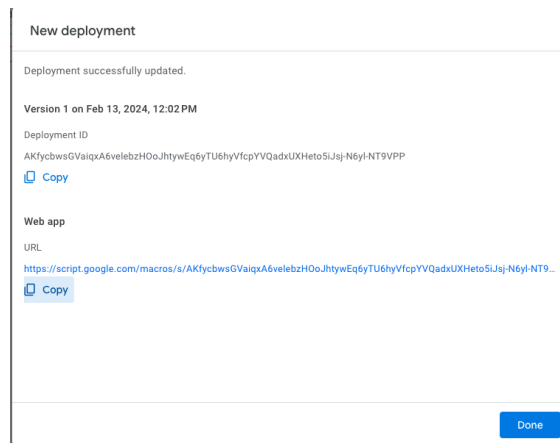


Figure 27: New Deployment. Note the Web App URL, which is needed on the Things Network.

10. Finally, Copy the Web App URL shown in Figure 27. It is now possible to start forwarding Sensor data decoded on the Things Network to this Google Sheet.

```
https://script.google.com/macros/s/AKfycbwsGVaiqxA6velebzHOoJhtywEq6yTU6hyVfcpYVQadxUXHeto5iJsj-N6yl-NT9VRR/exec
```

That's it! You've created a basic Google Sheets Apps Script that logs sensor information from the Things Network.

Note, Appendix 5 shows the code for processing Decentlab DL-MBX Ultrasonic Sensors in Google Sheets.

5. Configuring TTN Web Hook, and Plotting Data

The Things Network (TTN) allows for the integration of webhooks, which enable the forwarding of sensor data to external services such as Google Sheets. Here's a brief overview of how to configure a webhook for flood-site-location-1 application created in Section 3 to forward data to the Google Sheet Apps Script Created in Section 4.

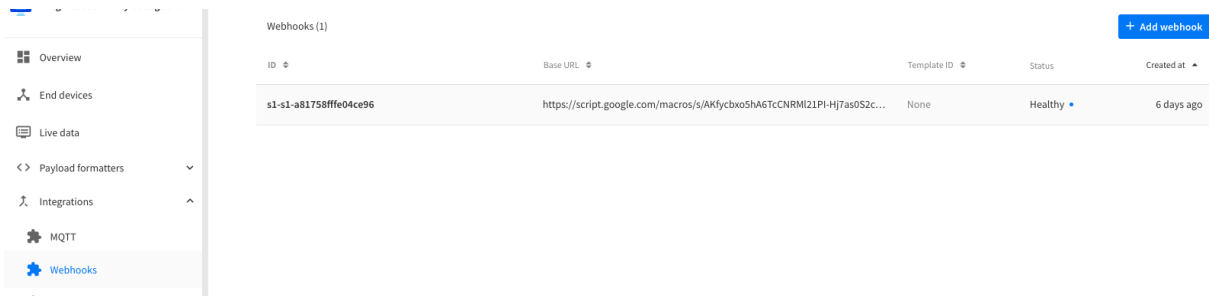


Figure 28: Adding the Google Sheets webhook created in Section 4 to Things Network *flood-site-location-1* Application

1. Log into Things Network Sandbox, and Click the flood-site-location-1 Application Created in Section 3
2. Click Integrations and Webhook in the left hand menu. Then click Add Webhook. See Figure 28.
3. Select Custom Webhook on the next Screen.
4. Enter a Web Hook ID, Select Web Hook Format: JSON, and copy the Apps Script URL created in Section 4 into Base URL. See Figure 29.
5. Then scroll down to the button of the page, and Enable Uplink Message (see Figure 30) and click Save Webhook

Web Hook ID: my-google-sheet-flood-site-1

Web Hook Format: JSON

Base URL:

https://script.google.com/macros/s/AKfycbwsGVAiqxA6velebzHOoJhtywEq6yTU6hyVfcpYVQadxUXHeto5iJsJ-N6yl-NT9VRR/exec

Add webhook

The Webhooks feature allows The Things Stack to send application related messages to specific HTTP(S) endpoints. You can also use webhooks to schedule downlinks to an end device. Learn more in our [Webhooks guide](#).

General settings

Webhook ID *

my-google-sheet-flood-site-1

Webhook format *

JSON

Base URL *

lebzhOoJhtywEq6yTU6hyVfcPVQadxUXHeto5iJsj-N6yl-NT9VRR/exeq

Figure 29: Copy the URL created by the Google Sheets Apps Script in Section 4 into the Base URL field. Then scroll down to the button of the page, and Enable Uplink Message (see Figure 30) and click Save Webhook.

Enabled event types

For each enabled event type an optional path can be defined which will be appended to the base URL

☒ Uplink message

An uplink message is received by the application

Figure 30: Enable the Uplink message Type.

The next sensor reading received by the Things Network for the ELT Ultrasonic LoRaWAN Distance Sensor will be sent to the Google Sheet.

Here is a sample of one sensor reading from the Elsys ELT2 Ultrasonic sensor, which has been recorded on Google Sheets, see Table 3.

Dev_eui	Distance (mm)	Humidity	Pressure	Temperature	Vdd (Battery)	dateTime	Time
A81758FFFE04CBB8	1412	67	1010.251	19.3	3645	07/02/2024 13:42:06	13:42:06

Table 3: Sample Sensor Reading in Google Sheets

Note, every additional sensor added to the Things Network flood-site-location-1 application will have a new tab with the DevEUI automatically created on the Google Sheet defined in Section 4. No further editing of the code is required, see Figure 31.

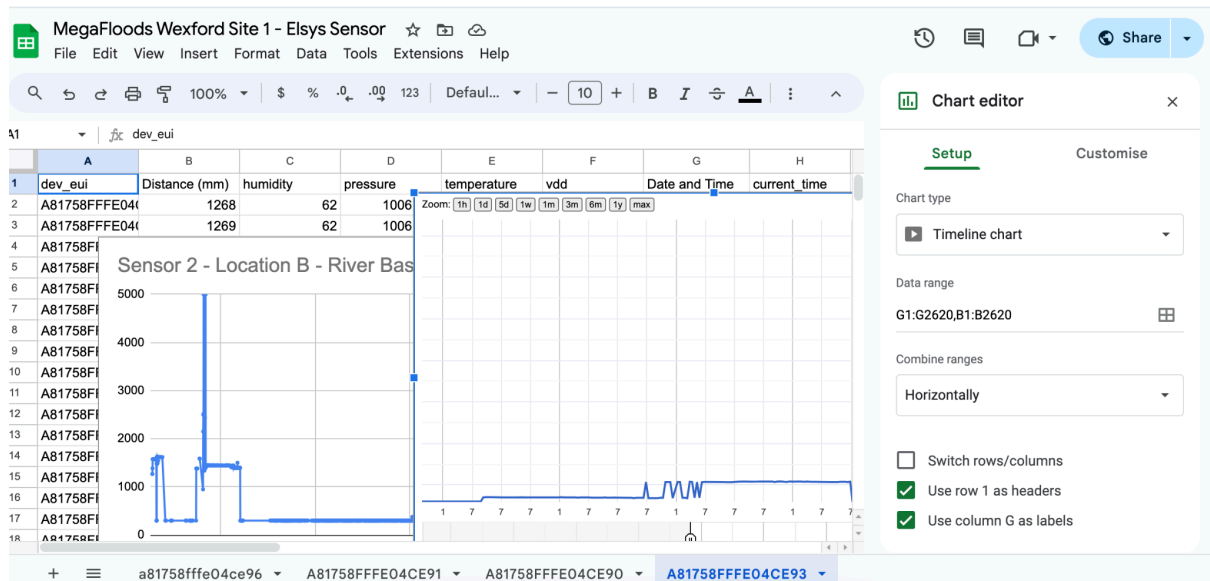


Figure 35: Timeline Chart

6. Final Steps: Staging/Test Deployment

Before final site deployment, it is worth testing if the sensor is recording distance to the water correctly. Position the sensor above a body of water, and use a measuring tape to ensure the distances between the sensor and the water body are getting measured accurately (allowing for a margin of error within a few millimetres).

Test Deployment

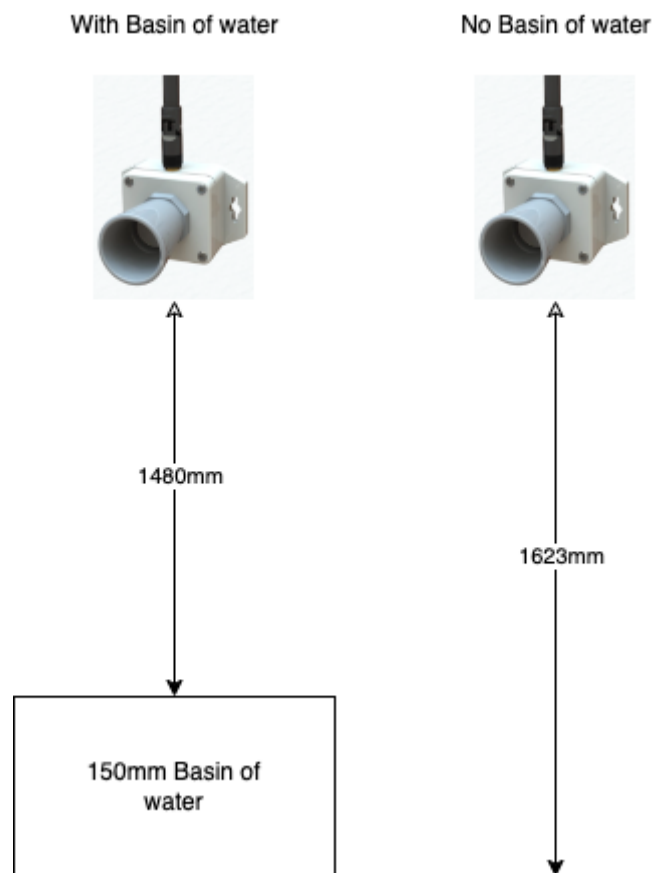


Figure 35: Testing Sensor Before Final Site Deployment

Sample Staging/Test Scenario:

1. Put the sensor above a basin of water which has about 150mm of water in it.
2. The distance from the sensor to the top of the water is 1480mm, See Figure 35. The following sensor reading was received in Google Sheets, see Table 4:

Dev_eui	Distance (mm)	Humidity	Pressure	Temperature	Vdd (Battery)	dateTime	Time
A81758FFFE04CE88	1480	62	986.208	18.3	3649	08/02/2024 13:13:05	13:13:04

Table 4: Sample Sensor Reading - The distance from the sensor to the top of the water is 1480mm in Google Sheets

3. Remove the basin of water. Do not touch or move the sensor.
4. The sensor is now reading a distance of 1623 mm, See Figure 35.

Dev_eui	Distance (mm)	Humidity	Pressure	Temperature	Vdd (Battery)	dateTime	Time
---------	---------------	----------	----------	-------------	---------------	----------	------

A81758FFFE0 4CE88	1623	62	986.106	18.4	3649	08/02/2024 13:18:05	13:18:05
----------------------	------	----	---------	------	------	------------------------	----------

Table 5: Sample Sensor Reading - The distance from the sensor to the ground is now 1623mm after removing the basin of water

The conclusion here is that the sensor will read the distance to the top of the water.

To get the actual height of the water, you need to do a calculation in Google Sheets. So you would need to know the distance from the river bed to the sensor. In a separate Google Sheets column you can determine the depth of the river by subtracting the difference. For example:

Parameter	Measurement
Distance from sensor to river bed	2000mm
Sensor reading	1623mm
Result: River Depth	$2000 - 1623 = 377\text{mm}$

Table 5: River Depth Calculation in Google Sheets

Appendix 1: Overview of LoRaWAN Spread Factor

LoRaWAN spread factor (SF) is a parameter used in LoRaWAN communication to define the spreading of data over a certain frequency band. LoRaWAN, which stands for Long Range Wide Area Network, is a low-power, wide-area networking protocol designed for long-range communication between IoT (Internet of Things) devices and gateways.

The spread factor determines the rate at which data is transmitted, as well as the robustness of the communication link against noise and interference. A higher spread factor (e.g., 7) means a longer transmission time and greater resistance to interference, but it also reduces the data rate.

In LoRaWAN, spreading factors typically range from SF7 to SF12, with SF7 being the fastest but least robust and SF12 being the slowest but most robust. Here's a brief overview:

- SF7: Fastest data rate, shortest range, and least robust in terms of handling interference.
- SF8: Slightly slower than SF7, with a bit longer range and improved robustness.
- SF9: Slower than SF8, with further improved range and robustness.
- SF10: Slower than SF9, with an even longer range and increased robustness.
- SF11: Slower than SF10, offering extended range and enhanced robustness.
- SF12: Slowest data rate, longest range, and highest robustness against interference.

Note, the Spread Factor is configured in The Things Network. Here are some key points about LoRaWAN spread factor:

Spreading Factor Range: LoRaWAN supports spreading factors ranging typically from 7 to 12. The spreading factor essentially controls the bandwidth-time product. A lower spreading factor means wider bandwidth but shorter transmission time, while a higher spreading factor means narrower bandwidth but longer transmission time.

Data Rate vs. Range: Lower spreading factors (e.g., SF7) result in higher data rates but shorter ranges, while higher spreading factors (e.g., SF12) provide longer ranges at the expense of lower data rates.

Impact on Battery Life: The choice of spreading factor can have implications for battery life in IoT devices. Higher spreading factors (e.g., SF7) typically require less energy per bit transmitted, making them more suitable for battery-powered devices that need to operate for extended periods.

In summary, LoRaWAN spread factor is a critical parameter in LoRaWAN communication, influencing data rate, range, signal robustness, and energy consumption. The appropriate choice of spreading factor depends on factors such as desired range, data rate requirements, and the operating environment's interference levels making them more suitable for battery-powered devices that need to operate for extended periods.

Appendix 2: Elsys Battery Life Calculator

The following link shows the Elsys battery life calculator:

<https://www.elsys.se/en/battery-life-calculator/>

For our Flood Management project, we concluded that utilising a Spread Factor (SF) of 12 is suitable due to the anticipated dense foliage surrounding the waterways during summer months. We determined that setting the Sensor Sample Time to 1800 (every 30 minutes) would strike an optimal balance between receiving sensor measurements and preserving battery life, see Figure 36.

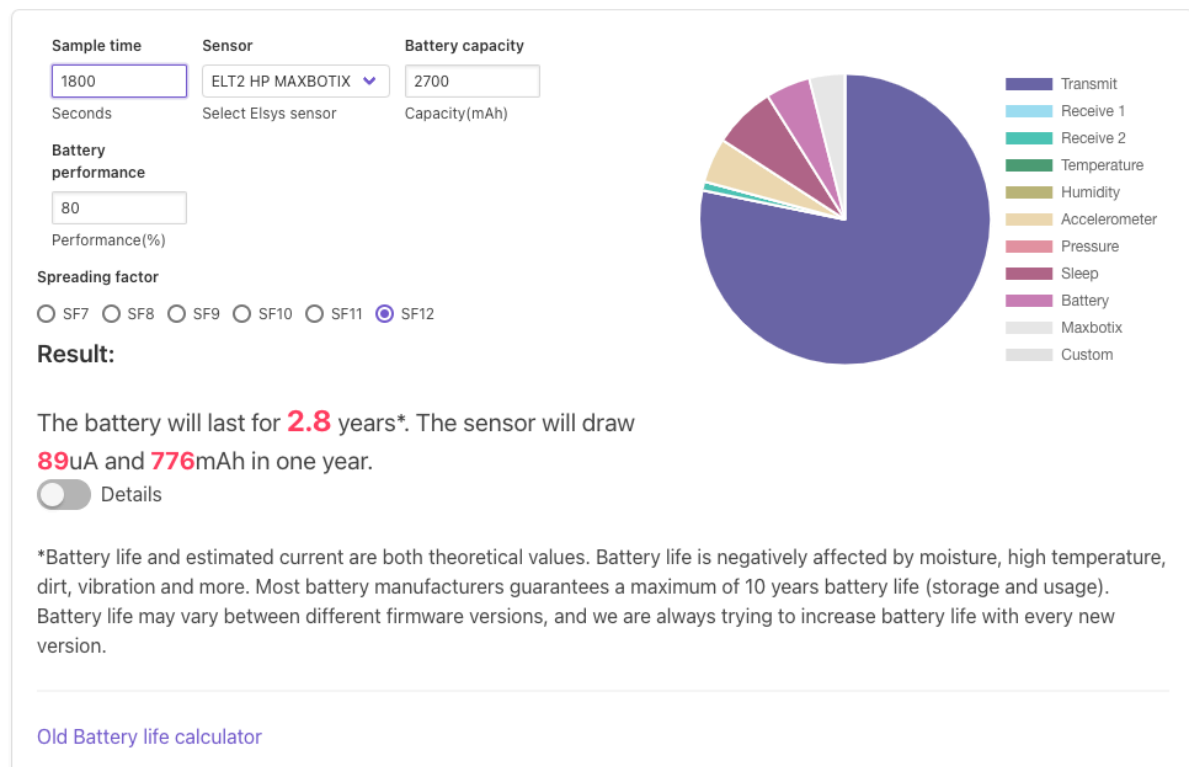


Figure 36: LoRaWAN Battery Life Calculator

Appendix 3: Adding A LoRaWAN Gateway to The Things Network (TTN)

Adding a LoRaWAN gateway to The Things Network (TTN) is a crucial step in extending the network coverage and enabling communication between devices and the TTN backend, See Figure 35. Gateways serve as bridges between devices and The Things Network. All gateways within reach of an IoT device receive the device's messages and forward them to the network. There are many different types of LoRaWAN gateways, and each one needs to be configured slightly differently. The Things Network provides comprehensive documentation for adding different types of gateways to the network. These instructions can be viewed on their website[11] [12] [13].

<https://www.thethingsindustries.com/docs/gateways/>

Using these instructions, it's possible to add LoRaWAN Gateways to the Things Network Sandbox, clicking the Go To Gateways option in Figure 35, then clicking Register Gateway, and Adding the Gateway EUI show in Figure 37. The Gateway EUI can be retrieved from the Gateway documentation, while configuring parameters associated with the Things Network can be done by logging into the Gateway, See Figure 38.

By following these instructions, you can integrate a multitude of different types of LoRaWAN Gateways into the Things Network Sandbox. Initially, navigate to the Gateways section on your Things Network account [14] and select "Go To Gateways" as depicted in Figure 35. Then proceed to register the gateway by clicking on "Register Gateway" and inputting the Gateway EUI displayed in Figure 36. The Gateway EUI can typically be found in the Gateway documentation. Further configuration of parameters related to the Things Network can be accomplished by logging into the Gateway interface, as illustrated in Figure 39.

<https://eu1.cloud.thethings.network/console/gateways/add>

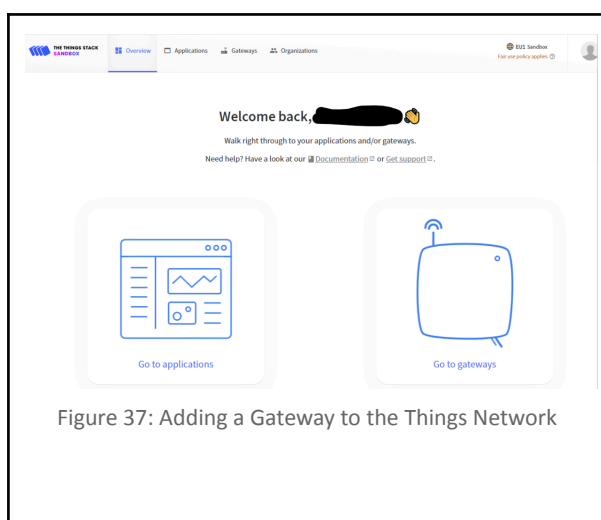


Figure 37: Adding a Gateway to the Things Network

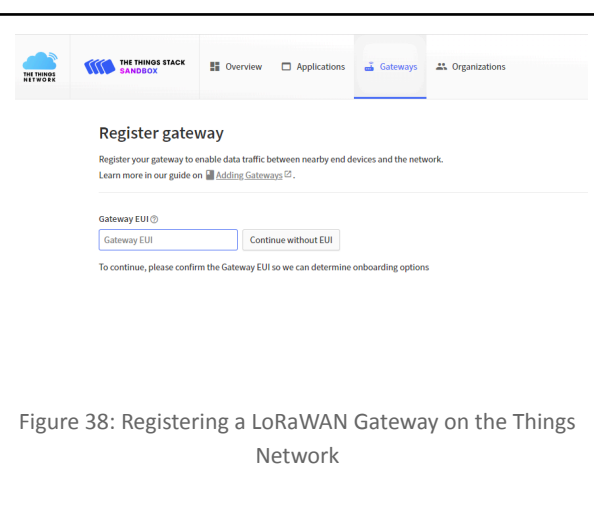


Figure 38: Registering a LoRaWAN Gateway on the Things Network

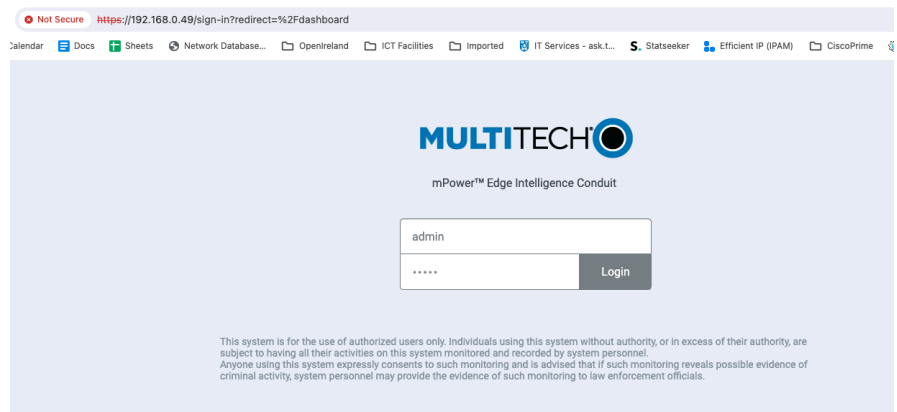


Figure 39: Multitech Gateway Login

Appendix 4: Adding Decentlab Sensors to The Things Network (TTN)

Decentlab is a company specialising in wireless sensor systems for environmental monitoring and other applications. The Model DL-MBX is one of their products, designed for precise measurement of various environmental parameters, specifically distance. Key features of the DL-MBX include:

- **Multi-Parameter Monitoring:** The DL-MBX can measure a range of environmental parameters, including temperature, humidity, distance, making it versatile for various applications.
- **Wireless Communication:** It utilises LoRaWAN technology for wireless data transmission, allowing it to operate over long distances with low power consumption, which is ideal for remote or difficult-to-access locations.
- **High Accuracy and Reliability:** The sensors used in the DL-MBX are designed to provide accurate and reliable data, essential for scientific and industrial applications where precision is critical.
- **Battery-Powered:** The device is battery-operated, ensuring it can function independently without the need for a constant power supply. This feature, combined with its low power consumption, ensures long-term operation in the field.
- **Data Integration:** Data collected by the DL-MBX can be integrated into various data management systems, enabling users to analyse and utilise the information for decision-making processes effectively.

Overall, the Decentlab DL-MBX is a robust and versatile tool for environmental monitoring, offering reliable performance and ease of integration into existing data infrastructures.

Decentlab DL-MBX Ultrasonic Sensor Configuration

Use the following instructions to add Decentlab Ultrasonic sensors to the things network.

Click “Applications” and “Register end Device”

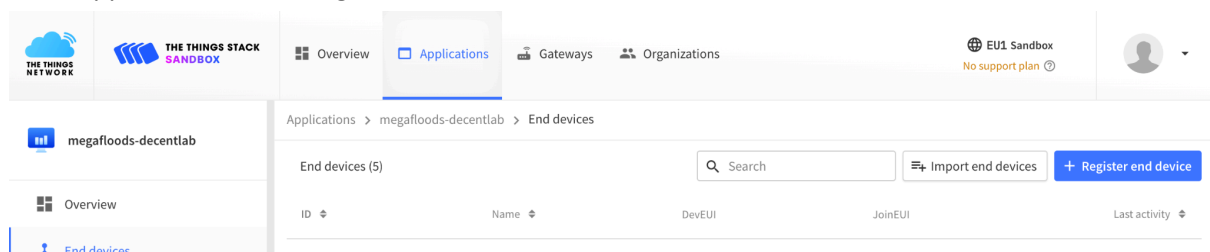





Figure 40: Click “Applications” and “Register end Device”

Register end device


Does your end device have a LoRaWAN® Device Identification QR Code? Scan it to speed up onboarding.


 Scan end device QR code
  [Device registration help](#)


End device type


Input method 


☒ Select the end device in the LoRaWAN Device Repository
☐ Enter end device specifics manually

End device brand  *

Model  *

Hardware Ver.  *

Firmware Ver.  *

Profile (Region)  *


Decentlab GmbH

DL-MBX - Ultraso...

1

1.5.7

EU_863_870



DL-MBX - Ultrasonic Distance/Level Sensor

LoRaWAN Specification 1.0.2, RP001 Regional Parameters 1.0.2 revision B, Over the air activation (OTAA), Class A

The Decentlab DL-MBX is equipped with an ultrasonic distance/level sensor for measuring distance. Suitable for monitoring water level, flood, snow level, silo level, etc.

[Product website](#) | [Data sheet](#)

Figure 41: Select the Decentlab GmbH Sensor

End device brand	Decentlab GmbH
Model	DL-MBX-Ultrasonic
Hardware Ver.	1
Firmware Version	1.5.7
Profile Region	EU_863_870
Frequency Plan	Europe 863-870 MHz (SF9 for RX2 - Recommended)
Join EUI	0000000000000000
DevEUI	A81758FFFB3D57ED
AppKey	DD353652A69850365894E9C5EC4CC9A9
End Device ID	s-wex-sensor-1
View Registered end device	Check box

Table 6: Configuring parameters for the DL-MBX Ultrasonic Distance /Level Sensor

In the “Register end Device” screen select the following parameters(Listed in Table 6) for the DL-MBX Ultrasonic Distance /Level Sensor.

Next, when all the parameters have been entered, click on Register end Device, shown in Figure 18 and Figure 19. Assuming the device is registered correctly, you should see a similar screen as shown in Figure 20.

Frequency plan ⓘ *

Europe 863-870 MHz (SF9 for RX2 - recommended) | ▾

Provisioning information

JoinEUI ⓘ *

00 00 00 00 00 00 00 00

Reset

This end device can be registered on the network

DevEUI ⓘ *

EE 33 44 33 44 33 32 54

Generate

0/50 used

AppKey ⓘ *

A2 2B 91 AA 26 F0 39 14 D0 26 1A A4 73 3A 1B BD

Generate

End device ID ⓘ *

eui-ee33443344333254

This value is automatically prefilled using the DevEUI

After registration

☒ View registered end device

☐ Register another end device of this type

Figure 42: Device Registration Parameters

After registration


☒ View registered end device

☐ Register another end device of this type

Register end device

Figure 43: Register end device

You should see the end device added to the applications screen, as shown in Figure 44.


s-wex-sensor-1
ID: s-wex-sensor-1

↑ n/a ↓ n/a • No activity yet ⓘ

[Overview](#) [Live data](#) [Messaging](#) [Location](#) [Payload formatters](#) [General settings](#)

General information

End device ID	s-wex-sensor-1	
Frequency plan	Europe 863-870 MHz (SF12 for RX2)	
LoRaWAN version	LoRaWAN Specification 1.0.3	
Regional Parameters version	RP001 Regional Parameters 1.0.3 revision A	
Created at	Feb 13, 2024 11:03:47	

Hardware

Brand	elsys
Model	elt-ultrasonic
Hardware version	1.0
Firmware version	1.0

Activation information

AppEUI	00 00 00 00 00 00 00 00	<>
DevEUI	A8 17 58 FF FB 3D 57 ED	<>
AppKey	

Session information

This device has not joined the network yet

Figure 44: Registered end device.

Turn on the sensor and wait for it to join the network, see Figure 45

north-wexford-thnp-multitech > Gateways > Live data			
Receive uplink message		Data preview	
time	type	Verbose stream <input type="checkbox"/> Export as JSON <input type="checkbox"/> Pause <input type="checkbox"/> Clear <input type="checkbox"/>	
↑ 15:58:08	Receive uplink message	JoinEUI: 32 A8 CB 90 00 00 00 11	DevEUI: A8 17 58 FF FE 04 CE 96
↑ 15:57:52	Receive uplink message	JoinEUI: 32 A8 CB 90 00 00 00 11	DevEUI: A8 17 58 FF FE 04 CE 96

Figure 45: Sensor Join Requests received by the Gateway.

When the Sensor Successfully joins the network (*this might take a few hours*), you will see something similar to Figure 46. At this point sensor data will start to be automatically decoded. Note sensor measurements for Distance, battery, etc.

Overview

Applications

Gateways

Organizations

EU1 Sandbox

No support plan

diarmuidc

MegaFloods Trinity College Sit...

Overview

End devices

Live data

Payload formatters

Integrations

Collaborators

API keys

General settings

Applications > MegaFloods Trinity College Site 1 > Application data

Time

Entity ID

Type

Data preview

Verbose stream

Export as JSON

Pause

Clear

16:50:36

eui-a81758fffe04c...

Update end device

["activated_at"]

16:50:36

eui-a81758fffe04c...

Forward uplink data message

DevAddr: 26 08 74 6A <> Payload: { } 3E 61 07 01 08 05 09 01 0A 00 08 05 00 00 0C 05 10 0

16:50:32

eui-a81758fffe04c...

Forward join-accept message

DevAddr: 26 08 74 6A <>

16:50:31

eui-a81758fffe04c...

Successfully processed joi...

16:50:30

eui-a81758fffe04c...

Accept join-request

DevAddr: 26 08 74 6A <>

16:50:30

Console: Events cleared

The events list has been cleared

Figure 46: Sensor successfully Joins the Things Network LoRaWAN Server.

Appendix 5: Google Sheets code for Decentlab DL-MBX Ultrasonic Sensor

```
// Create or open an existing Sheet and click Tools > Script editor and
enter the code below
// 1. Run > setup
// 2. Publish > Deploy as web app
//    - enter Project Version name and click 'Save New Version'
//    - set security level and enable service (most likely execute as
'me' and access 'anyone, even anonymously')
// 3. Copy the 'Current web app URL' and post this in your form/script
action in the Things Network
```

```
var SCRIPT_PROP = PropertiesService.getScriptProperties(); // new
property service
```

```
// If you don't want to expose either GET or POST methods you can comment
out the appropriate function
```

```
//function doGet(e){
//  return handleResponseV2(e);
//}
```

```
//ttn uses doPost function
```

```
function doPost(e){
  return handleResponseV2(e);
}
```

```
function currentTime(){
  // Create a new Date object
  var currentTime = new Date();
```

```
// Get the current hour, minute, and second
```

```
var currentHour = currentTime.getHours();
var currentMinute = currentTime.getMinutes();
var currentSecond = currentTime.getSeconds();

// Format the time nicely
var formattedTime = currentHour + ":" + currentMinute + ":" +
currentSecond;

// Display the current time
console.log("The current time is: " + formattedTime);
return formattedTime;
}

function currentDateUSA() {
    // Create a new Date object
    var currentTime = new Date();

    // Get the current month, day, and year
    var currentMonth = currentTime.getMonth() + 1; // Add 1 because
getMonth() returns zero-based values
    var currentDay = currentTime.getDate();
    var currentYear = currentTime.getFullYear();

    // Format the date
    var formattedDate = currentMonth + '/' + currentDay + '/' + currentYear;
    return formattedDate;
}

function dateTime() {
    var currentdate = new Date();
    var datetime = "Last Sync: " + currentdate.getDate() + "/"
        + (currentdate.getMonth()+1) + "/"
```

```

        + currentdate.getFullYear() + " @ "
        + currentdate.getHours() + ":"
        + currentdate.getMinutes() + ":"
        + currentdate.getSeconds();
    return currentdate;
}

function handleResponseV2(e) {
    var lock = LockService.getPublicLock();
    lock.waitLock(30000); // wait 30 seconds before conceding defeat.
    try {
        //enable the following line for testing.
        //var jsonData = flatten(json_data);
        var jsonData = JSON.parse(e.postData.contents);

        //var jsonData = json_data;
        var devEui = jsonData.end_device_ids.dev_eui;

        // Create or open sheet based on dev_eui
        var doc = SpreadsheetApp.getActiveSpreadsheet();
        var sheet = doc.getSheetByName(devEui);
        if (!sheet) {
            sheet = doc.insertSheet(devEui);
            // Set headers if the sheet is newly created
            sheet.appendRow(['dev_eui', 'battery_voltage', 'device_id',
                'distance(mm)', 'number_of_valid_samples', 'date_time', 'current_time',
                'usa_date']);
        }

        var nextRow = sheet.getLastRow() + 1; // get next row

        // Extract data from JSON object
        var rowData = [
            jsonData.end_device_ids.dev_eui,

```

```

        jsonData.uplink_message.decoded_payload.battery_voltage.value,
        jsonData.uplink_message.decoded_payload.device_id,
        jsonData.uplink_message.decoded_payload.distance.value,

jsonData.uplink_message.decoded_payload.number_of_valid_samples.value,
        dateTime(),
        currentTime(),
        currentDateUSA(),
        // Add more parameters as needed
    ];

    // Append data to the next empty row in the spreadsheet
    sheet.appendRow(rowData);

    // return json success results
    return ContentService
        .createTextOutput(JSON.stringify({"result":"success", "row":
nextRow}))
        .setMimeType(ContentService.MimeType.JSON);
    } catch(e) {
        // if error return this
        console.error(e);
        return ContentService
            .createTextOutput(JSON.stringify({"result":"error", "error":
e}))
            .setMimeType(ContentService.MimeType.JSON);
    } finally { //release lock
        lock.releaseLock();
    }
}

function handleResponseV3(e) {
    var lock = LockService.getPublicLock();

```

```
lock.waitLock(30000); // wait 30 seconds before conceding defeat.
try {
    var jsonData = JSON.parse(e.postData.contents);
    var devEui = jsonData.dev_eui; // Assuming devEui is static for this
example

    // Create or open sheet based on devEui
    var doc = SpreadsheetApp.getActiveSpreadsheet();
    var sheet = doc.getSheetByName(devEui);
    if (!sheet) {
        sheet = doc.insertSheet(devEui);
    }

    // Flatten JSON object
    var flattenedData = flatten(jsonData);

    // If headers are not set, set them
    if (sheet.getLastRow() == 0) {
        var headers = Object.keys(flattenedData);
        sheet.appendRow(headers);
    }

    var nextRow = sheet.getLastRow() + 1; // get next row

    // Extract data from JSON object
    var rowData = Object.values(flattenedData);

    // Append data to the next empty row in the spreadsheet
    sheet.appendRow(rowData);

    // return json success results
    return ContentService
        .createTextOutput(JSON.stringify({"result": "success", "row":
nextRow}))
```




```

        .setMimeType(ContentService.MimeType.JSON);
    } catch(e) {
        // if error return this
        console.error(e);
        return ContentService
            .createTextOutput(JSON.stringify({"result":"error", "error":
e}))
        .setMimeType(ContentService.MimeType.JSON);
    } finally { //release lock
        lock.releaseLock();
    }
}

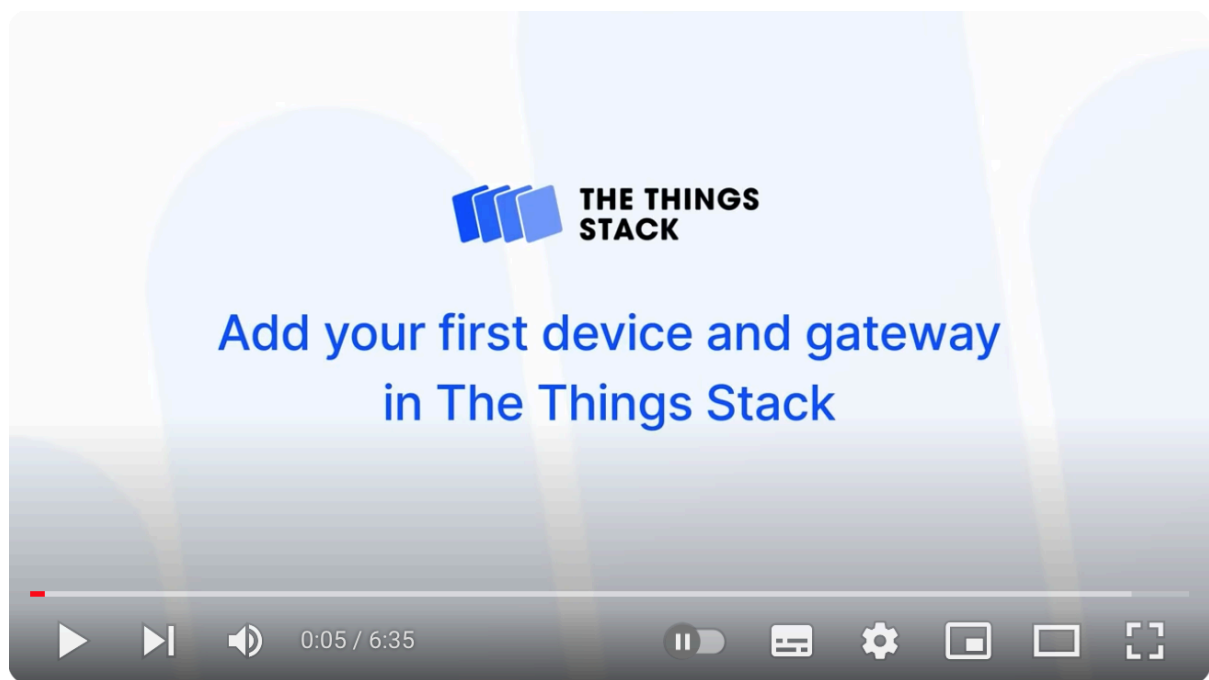
function flatten(obj) {
    var flattened = {};
    Object.keys(obj).forEach(function(key) {
        if (typeof obj[key] === 'object' && obj[key] !== null) {
            Object.assign(flattened, flatten(obj[key]));
        } else {
            flattened[key] = obj[key];
        }
    });
    return flattened;
}

```

Appendix 6: Things Network - How to add a device to the LoRaWAN network

 Adding Your First Device and Gateway in The Things Stack

<https://www.youtube.com/watch?v=Par4-Gio8po>



Additional YouTube Videos

The following set of YouTube videos show how to add:

1. [Log into the Things Network and Add an Elsys Ultrasonic Sensor](https://www.youtube.com/watch?v=e1_4EG8niEI)
2. [Setting up a Things Network Webhook and the AppScript in Google Sheets for Elsys Ultrasonic Sensor](<https://www.youtube.com/watch?v=7KsJ0Zh9yGU>)
3. [Sending REST API to Google Sheets App Script for processing data sent from the Things Network](<https://www.youtube.com/watch?v=6FiKC5Ka12g>)

4. [How to reset Things Network Device Nonce - in case the device loses connectivity to the Things Network](https://youtu.be/JatonWsU-rw)

Appendix 7: GitHub Link

The following is the link to the publicly accessible GitHub link for the project. It has all the code, and links to videos etc.

<https://github.com/dcollin5/ttn-google-sheet-integration/tree/main>

References

- [1] Industrial Distance Sensor, <https://www.elsys.se/en/elt-ultrasonic/>
- [2] Operating Manual ELT Ultrasonic, https://elsys.se/public/manuals/Operating_Manual_ELT_ultrasonic.pdf
- [3] ELT Ultrasonic Datasheet, https://elsys.se/public/datasheets/ELT_Ultrasonic_datasheet.pdf
- [4] Series LoRaWAN indoor wireless sensor, https://elsys.se/public/folders/Folder_EL2020.pdf
- [5] Sensor Settings on Android Store, <https://play.google.com/store/apps/details?id=se.elsys.nfc.sensorsettings&gl=SE>
- [6] Battery Life Calculator, <https://www.elsys.se/en/elt-ultrasonic/battery-life-calculator/#>
- [7] The Things ID - Login Screen, <https://eu1.cloud.thethings.network/oauth/login>
- [8] The Things Network Map, <https://www.thethingsnetwork.org/map>
- [9] Google Sheets training and help, <https://support.google.com/a/users/answer/9282959>
- [10] Google Sheets Graph tutorial: <https://www.youtube.com/watch?v=IFX9e9SqGlw>
- [11] Gateways, <https://www.thethingsindustries.com/docs/gateways/>
- [12] Kerlink Wirnet Station, <https://www.thethingsindustries.com/docs/gateways/models/kerlinkwirnetstation/>
- [13] MultiTech Conduit AEP, <https://www.thethingsindustries.com/docs/gateways/models/multitechconduit/>
- [14] Add LoRaWAN Gateway to the Things Network, <https://eu1.cloud.thethings.network/console/gateways/add>
- [15] ttn-google-sheet-integration on GitHub, <https://github.com/dcollin5/ttn-google-sheet-integration>