Executive Summary

This document walks through getting a Browan TBHV110 (Health Home Indoor Air Quality) Sensor connected to the Helium network, connected to Cayenne, and capturing data to a Google Sheet via a Pipedream workflow.

Steps

1. Add the device to the console and get data flowing
2. Build and apply a decoder function so you can see/access the sensor data in the payload
3. Build a Cayenne Integration so you can see the data in Cayenne
4. Build a Google Sheet integration so you can see/log/share the data in a spreadsheet

Table of Contents

Contents

Executive Summary 1
Table of Contents 1
References 1
Add the Device to the Helium Console 2
Build a Decoder and Apply to the Device with a Label 3
Build a Cayenne Integration 11
Build a Google Sheet integration 19
Debugging Tips 20
Appendix A – JSON Data from Console Debugger 23

References

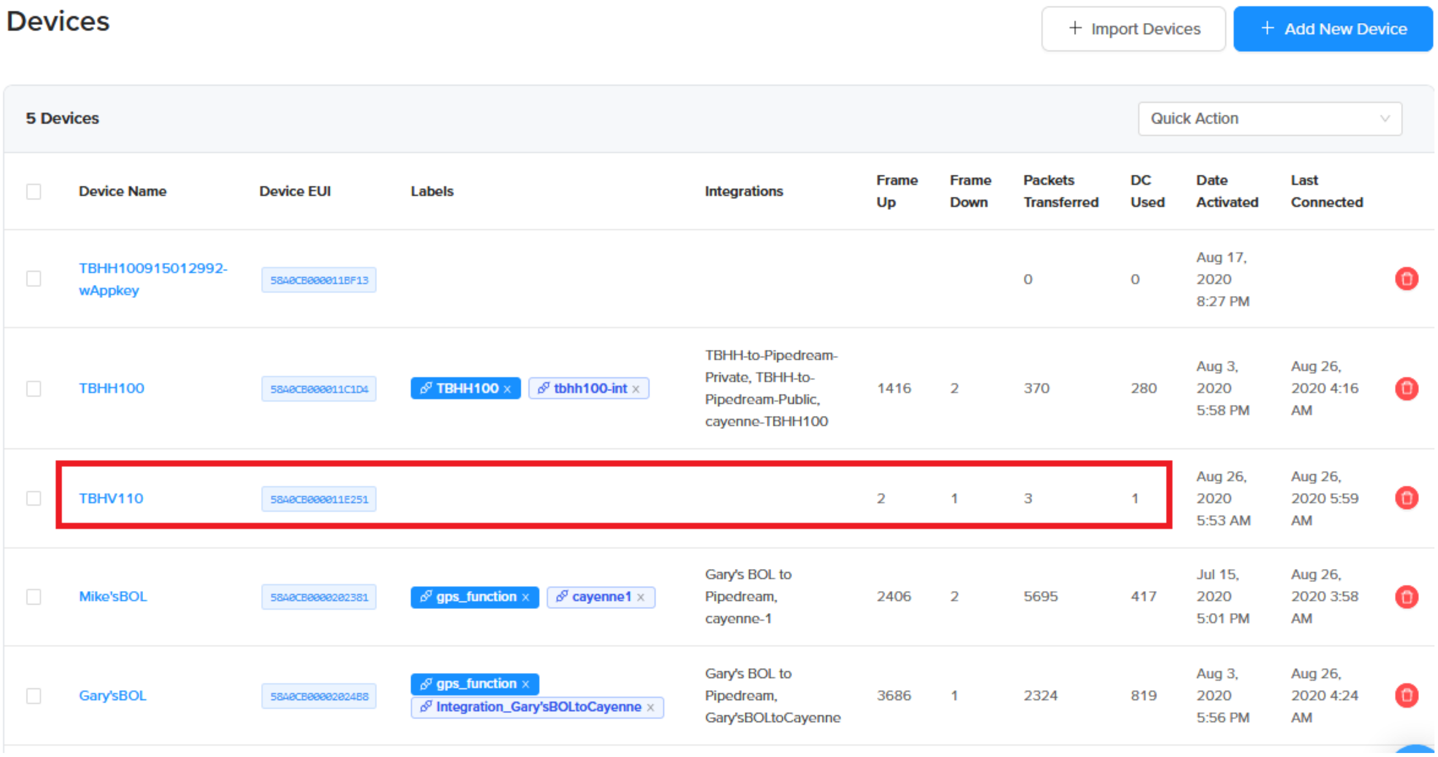
ID	Topic	Reference	Description
1	TBHV100	1. http://docs.microshare.io/assets/pdf/TBHV110.pdf 2. https://github.com/SensationalSystems/smart_building_sensors_decoder/blob/master/healthyhome.js	1. Reference Manual for the Browan TBHV110 2. Source code for a decode function
2	Helium	1. https://developer.helium.com/console/adding-devices 2. https://developer.helium.com/console/functions 3. https://developer.helium.com/console/integrations/mydevices-cayenne-integration 4. https://developer.helium.com/console/labels	1. How to add a device to the console 2. How to create a decoder function 3. How to set up a Cayenne dashboard for your Helium Devices 4. Organizing and Connecting with Labels
3	Pipedream	1. https://github.com/mikedsp/helium/blob/	1. How to get data from a Browan

		master/MyDocuments/HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf 2. https://github.com/mikedsp/helium/blob/master/MyDocuments/20200831_TBHV110%20data.xlsx 3. https://pipedream.com/@dangermikeb/tbhv110-to-google-sheet-private-p_ezCn1j	TBHH100 temperature and humidity sensor to flow in real time to a Google Sheet. 2. Example data sample from the Google Sheet 3. Pipedream workflow to get data from the Helium Console to the Google Sheet

Add the Device to the Helium Console

Using the instructions at the following URL, add the device to your Helium Console:

- https://developer.helium.com/console/adding-devices



Devices

5 Devices

□	Device Name	Device EUI	Labels	Integrations	Frame Up	Frame Down	Packets Transferred	DC Used	Date Activated	Last Connected
□	TBHH100915012992-wAppkey	58A0CB000011BF13					0	0	Aug 17, 2020 8:27 PM	
□	TBHH100	58A0CB000011C1D4	TBHH100 × tbhh100-int ×	TBHH-to-Pipedream-Private, TBHH-to-Pipedream-Public, cayenne-TBHH100	1416	2	370	280	Aug 3, 2020 5:58 PM	Aug 26, 2020 4:16 AM
□	TBHV110	58A0CB000011E251			2	1	3	1	Aug 26, 2020 5:53 AM	Aug 26, 2020 5:59 AM
□	Mike'sBOL	58A0CB0000202381	gps_function × cayenne1 ×	Gary's BOL to Pipedream, cayenne-1	2406	2	5695	417	Jul 15, 2020 5:01 PM	Aug 26, 2020 3:58 AM
□	Gary'sBOL	58A0CB00002024B8	gps_function × Integration_Gary'sBOLtoCayenne ×	Gary's BOL to Pipedream, Gary'sBOLtoCayenne	3686	1	2324	819	Aug 3, 2020 5:56 PM	Aug 26, 2020 4:24 AM

(above) TBHV110 added to the Console and data is flowing

Build a Decoder and Apply to the Device with a Label

Using the instructions at the following URL, create a decoder function and apply it to the device:

- https://developer.helium.com/console/functions

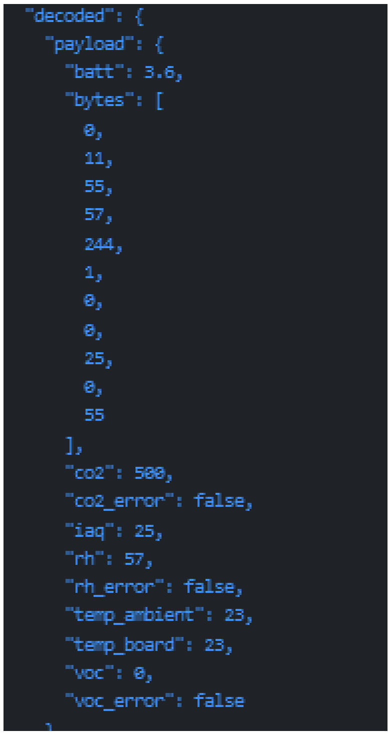
Fortunately, someone has already written a decoder function and shared it in the following GitHub repository:

- https://github.com/SensationalSystems/smart_building_sensors_decoder/blob/master/healthyhome.js

Here are the steps to go through:

1. Create a label called 'TBHV110decoder'
2. Create a new custom function
   a. Name it 'TBHV10decoderFunction' to make it easy to identify in the Console
   b. Copy and paste in the function code
   c. Apply the label called 'TBHV110decoder' to the function
3. Go back to the Devices tab of the Console and apply the 'TBHV110decoder' label from the previous step to the TBHV110 device
4. While looking at the TBHV110 device in the Console, turn on the debugger and wait for data to flow

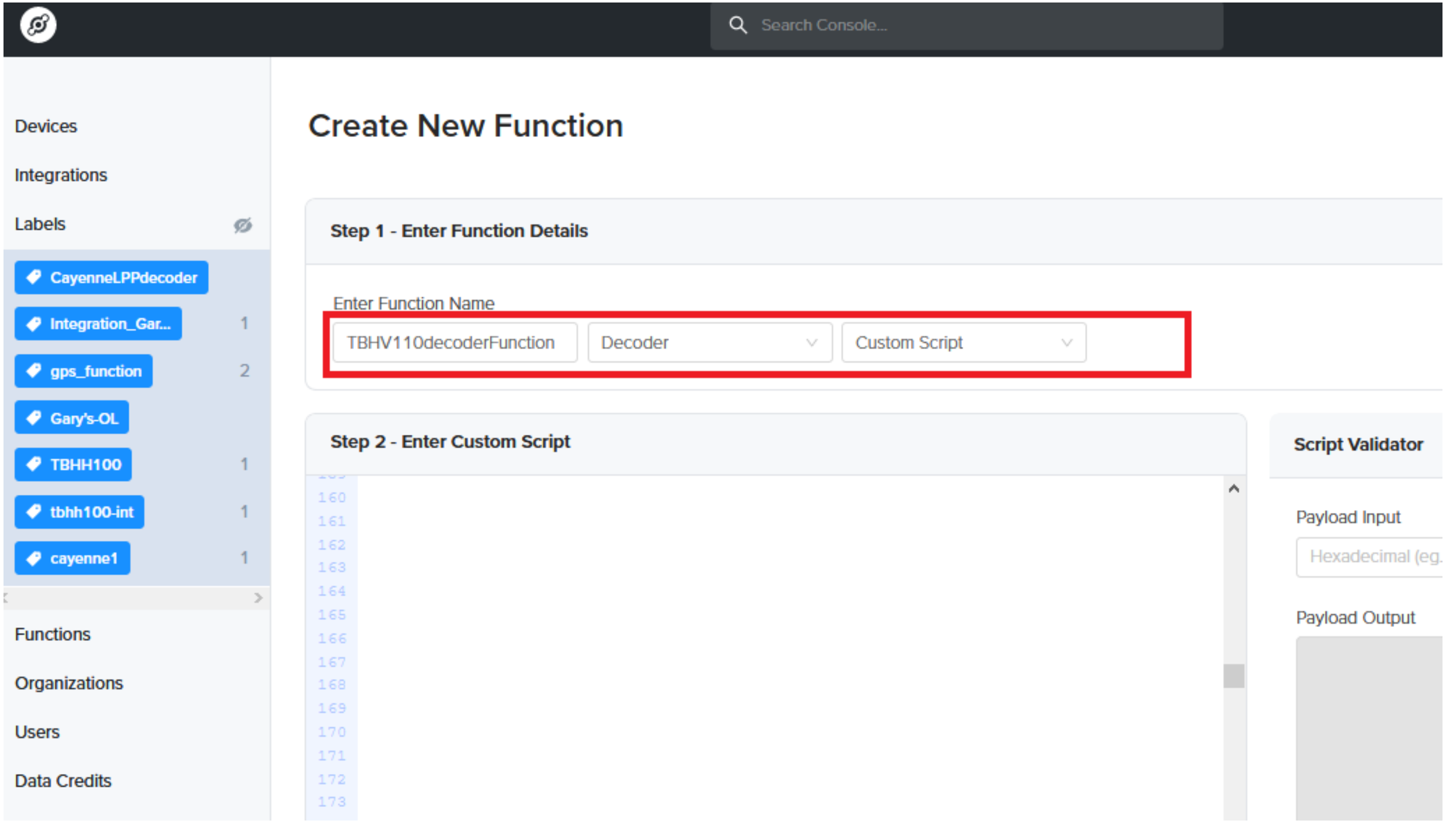
   Hint: Move the device into or out of your freezer to trigger a temperature event. Even doing that, it took about 10-15 minutes before I started seeing data in the debugger
5. Once the Console/debugger receives a data packet from the device, verify you can see the decoded data, which should look similar to screenshot below



```
  "decoded": {
    "payload": {
      "batt": 3.6,
      "bytes": [
        0,
        11,
        55,
        57,
        244,
        1,
        0,
        0,
        25,
        0,
        55
      ],
      "co2": 500,
      "co2_error": false,
      "iaq": 25,
      "rh": 57,
      "rh_error": false,
      "temp_ambient": 23,
      "temp_board": 23,
      "voc": 0,
      "voc_error": false
```

(above) Decoded data from the TBHV110 in the Console debugger

The screenshots below illustrate the steps above.



Create New Function

Step 1 - Enter Function Details

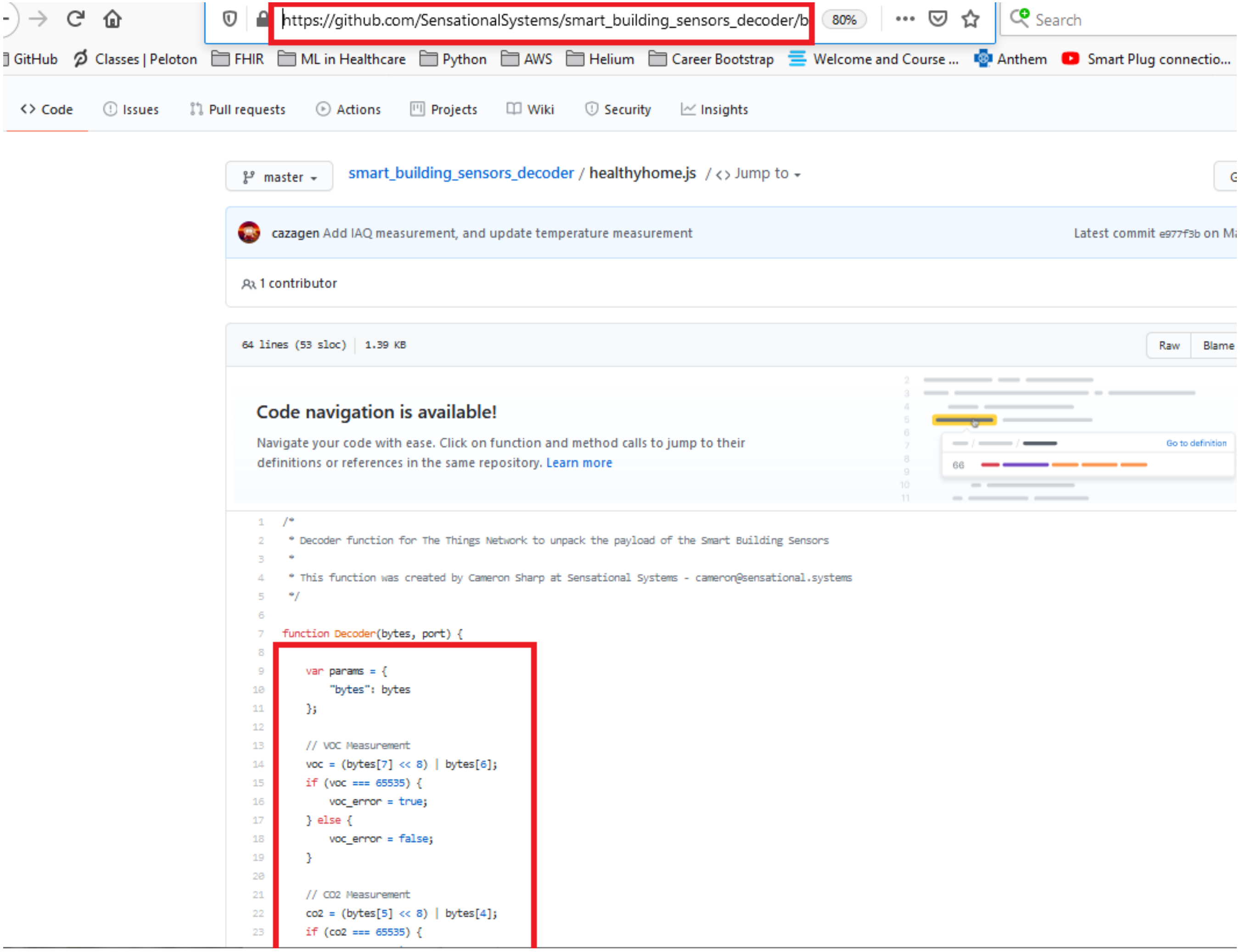
Enter Function Name: TBHV110decoderFunction | Decoder | Custom Script

Step 2 - Enter Custom Script

(above) Starting the creation of a new function.

- Name it ‘TBHV10decoderFunction’ so that is clear what it is
- Type = Custom Script

Note that the function body is empty



https://github.com/SensationalSystems/smart_building_sensors_decoder/b

smart_building_sensors_decoder / healthyhome.js

cazagen Add IAQ measurement, and update temperature measurement

Latest commit e977f3b on M

1 contributor

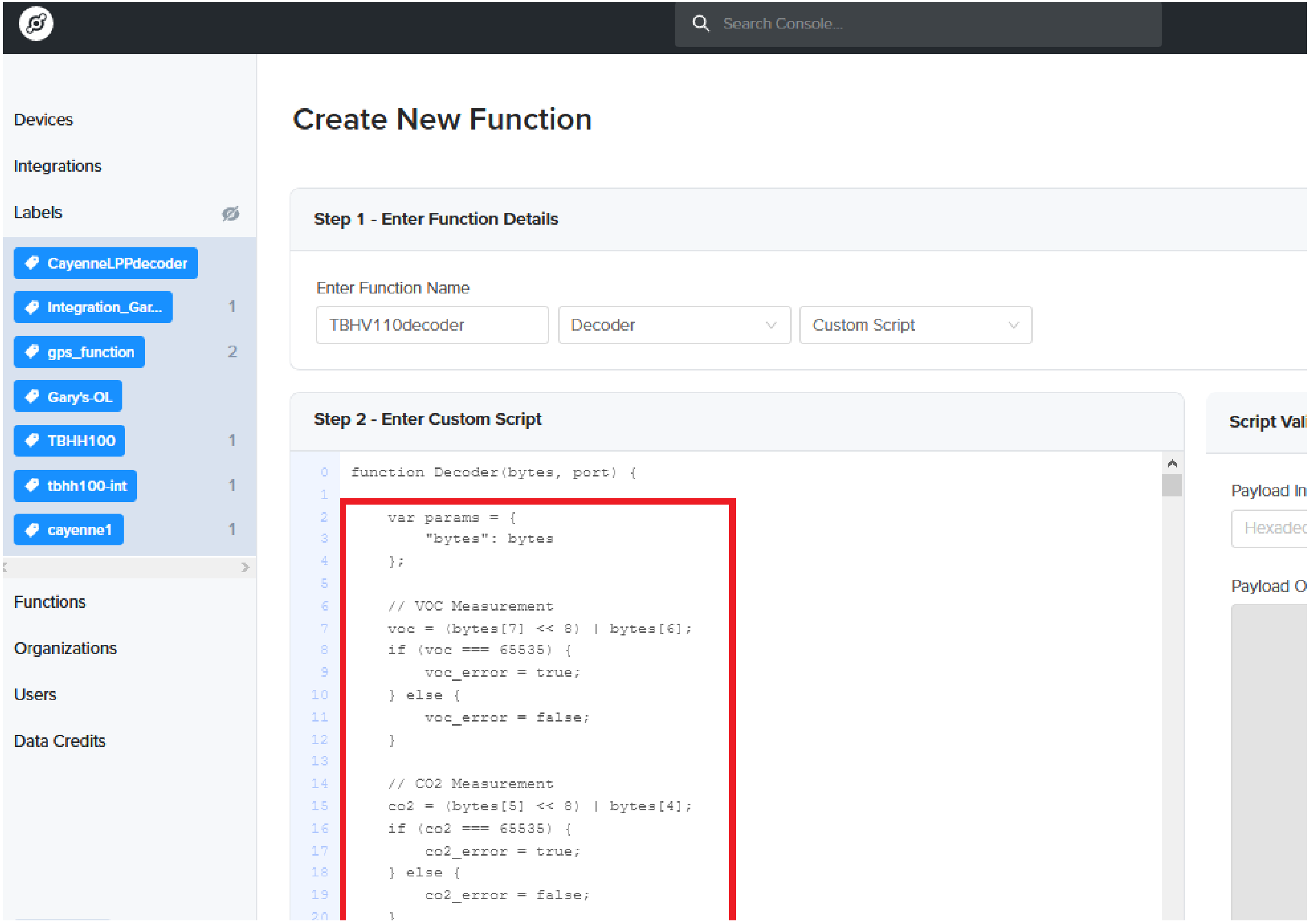
64 lines (53 sloc) | 1.39 KB

Code navigation is available!

Navigate your code with ease. Click on function and method calls to jump to their definitions or references in the same repository. Learn more

```
/*
 * Decoder function for The Things Network to unpack the payload of the Smart Building Sensors
 *
 * This function was created by Cameron Sharp at Sensational Systems - cameron@sensational.systems
 */

function Decoder(bytes, port) {

    var params = {
        "bytes": bytes
    };

    // VOC Measurement
    voc = (bytes[7] << 8) | bytes[6];
    if (voc === 65535) {
        voc_error = true;
    } else {
        voc_error = false;
    }

    // CO2 Measurement
    co2 = (bytes[5] << 8) | bytes[4];
    if (co2 === 65535) {
```

(above) Copying the decoder source code from the decoder function in the Sensational Systems GitHub repository



Create New Function

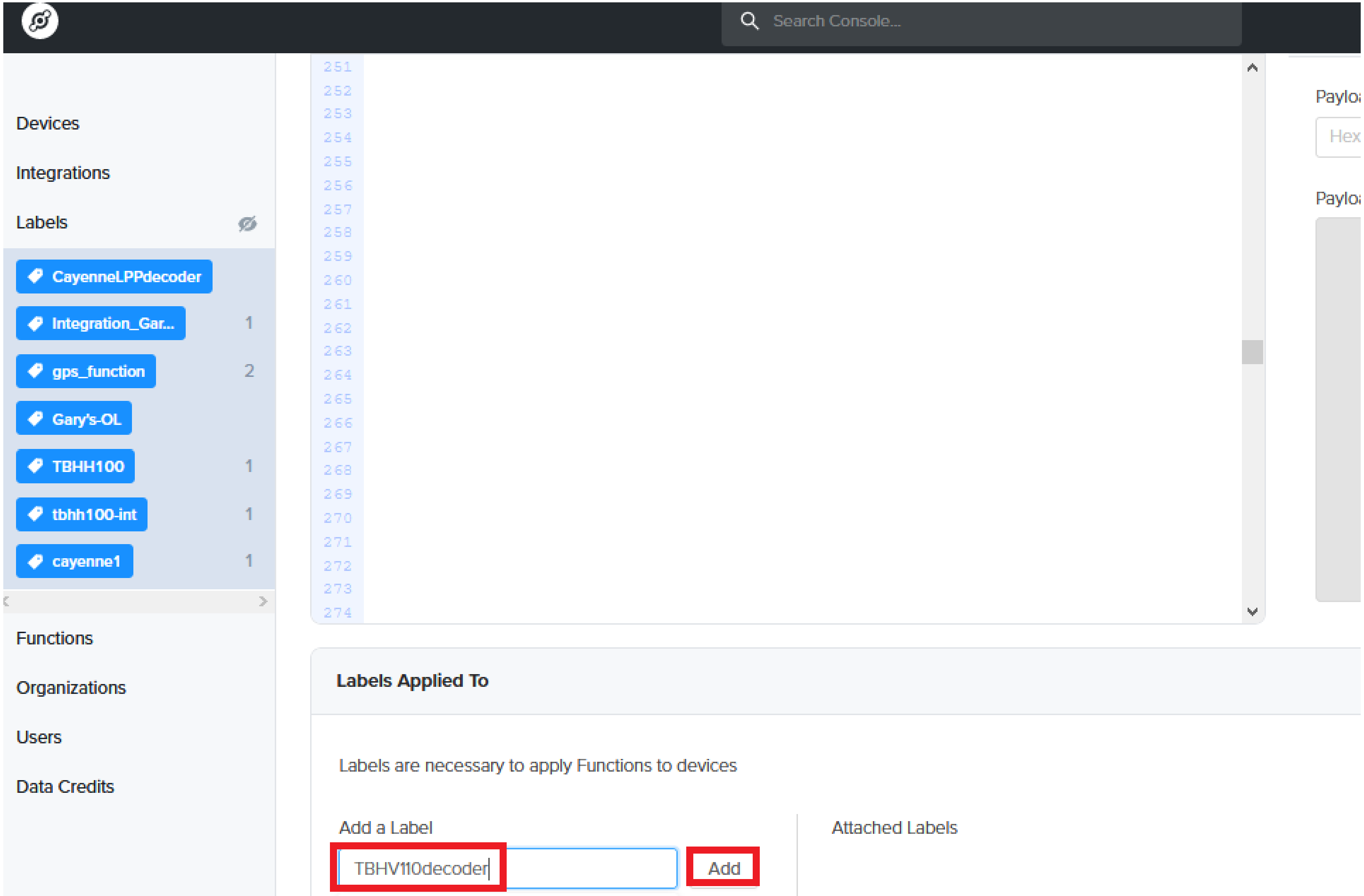
Step 1 - Enter Function Details

Enter Function Name: TBHV110decoder | Decoder | Custom Script

Step 2 - Enter Custom Script

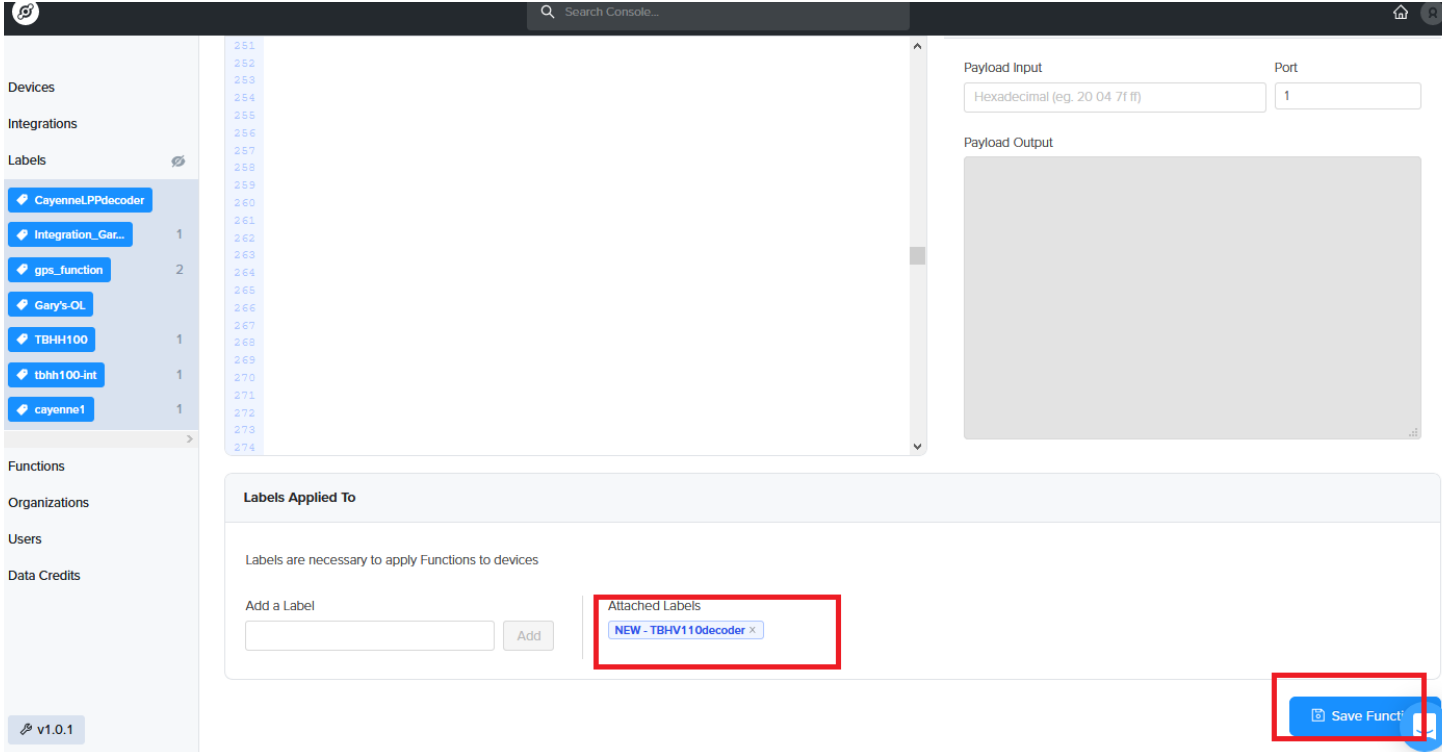
```
function Decoder(bytes, port) {

    var params = {
        "bytes": bytes
    };

    // VOC Measurement
    voc = (bytes[7] << 8) | bytes[6];
    if (voc === 65535) {
        voc_error = true;
    } else {
        voc_error = false;
    }

    // CO2 Measurement
    co2 = (bytes[5] << 8) | bytes[4];
    if (co2 === 65535) {
        co2_error = true;
    } else {
        co2_error = false;
    }
```

(above) Pasting the code into the new decoder function



(above) Creating/adding a label called 'TBHV110decoder'

Note – in the screenshot above I created 'TBHV110decoder' right here. Don't do that as it leads to trouble later. Create this label before applying it to the decoder function.



Payload Input: Hexadecimal (eg. 20 04 7f ff)

Port: 1

Payload Output

Labels Applied To

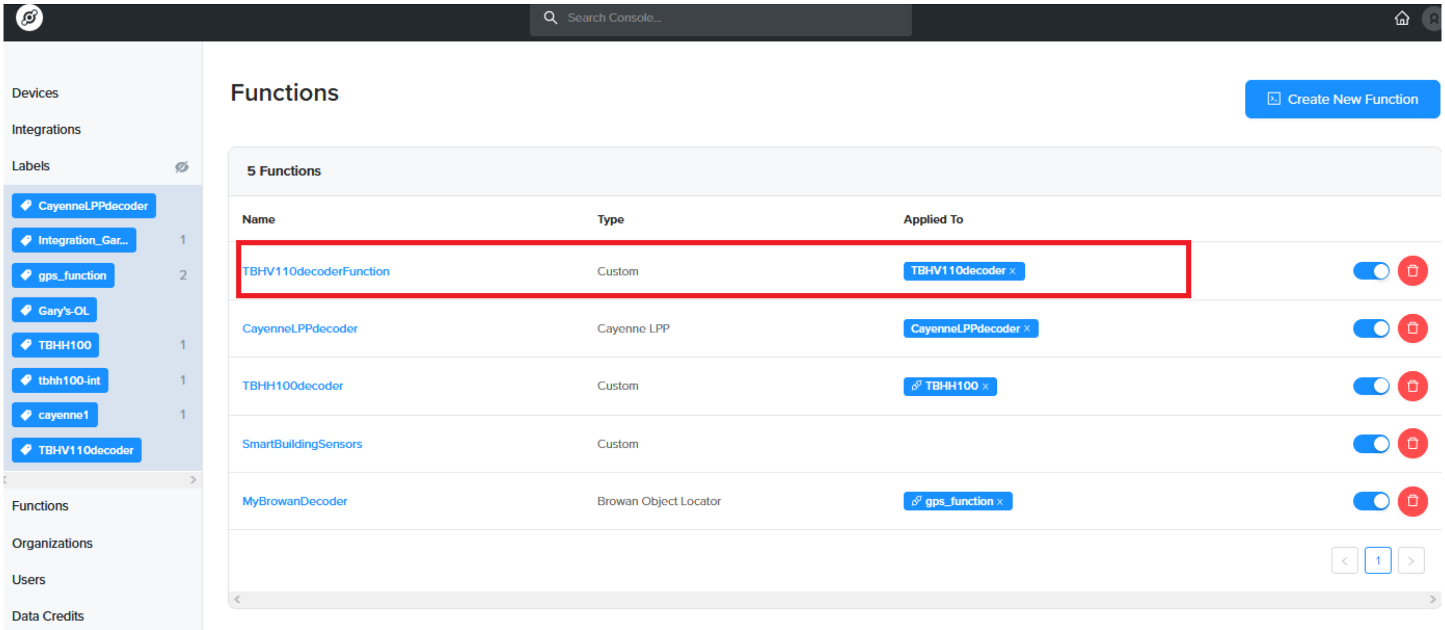
Labels are necessary to apply Functions to devices

Add a Label: Add

Attached Labels: NEW - TBHV110decoder ×

(above) New label added, then hit the 'Save the function' button

Note – in the screenshot above I created 'TBHV110decoder' right here. Don't do that as it leads to trouble later. Create this label before applying it to the decoder function.

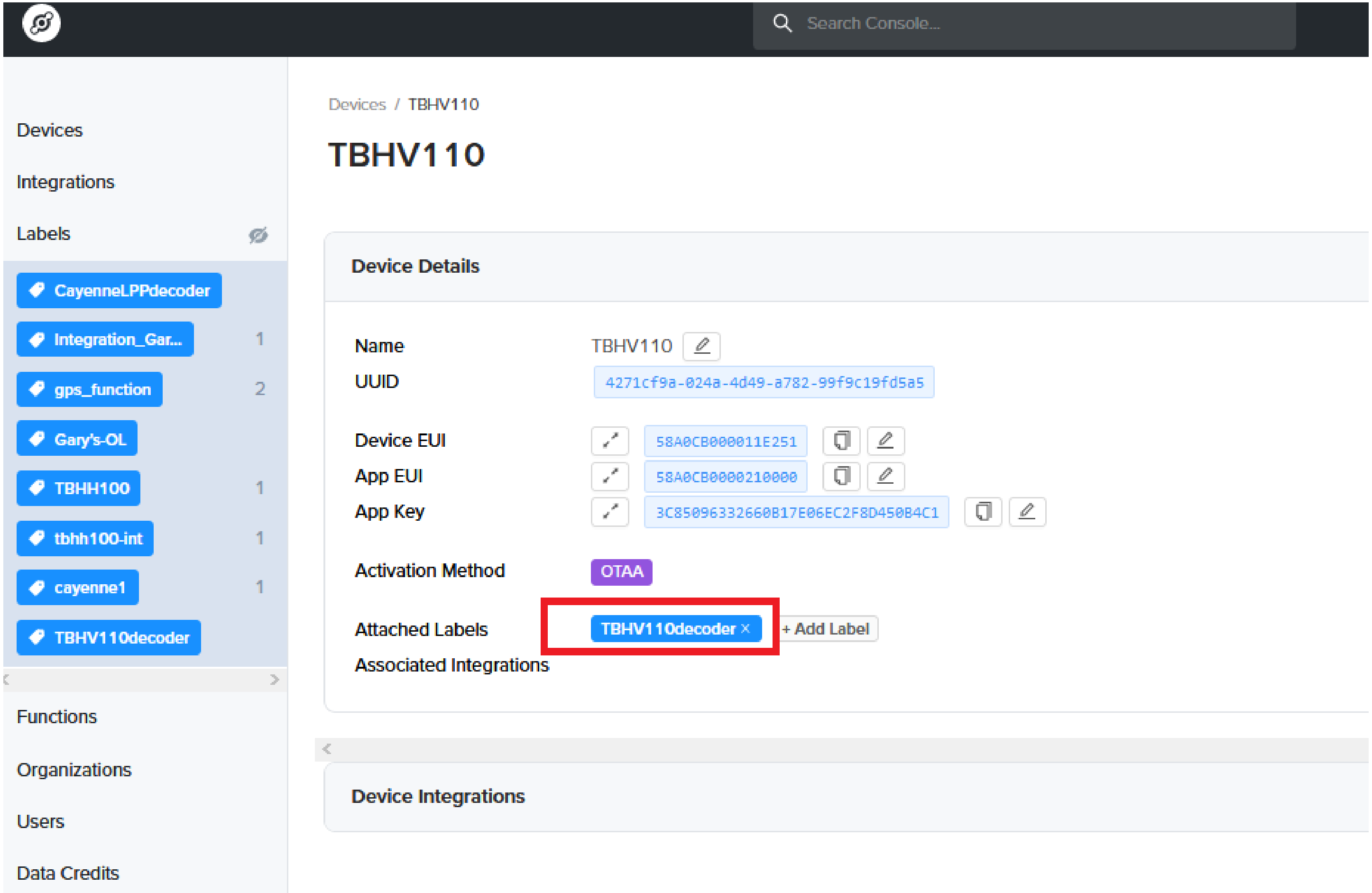


Functions

5 Functions

Name	Type	Applied To
TBHV110decoderFunction	Custom	TBHV110decoder ×
CayenneLPPdecoder	Cayenne LPP	CayenneLPPdecoder ×
TBHH100decoder	Custom	TBHH100 ×
SmartBuildingSensors	Custom	
MyBrowanDecoder	Browan Object Locator	gps_function ×

(above) The TBHV110 function in the Function list



TBHV110

Device Details

Name TBHV110

UUID 4271cf9a-024a-4d49-a782-99f9c19fd5a5

Device EUI 58A0CB000011E251

App EUI 58A0CB0000210000

App Key 3C85096332660B17E06EC2F8D450B4C1

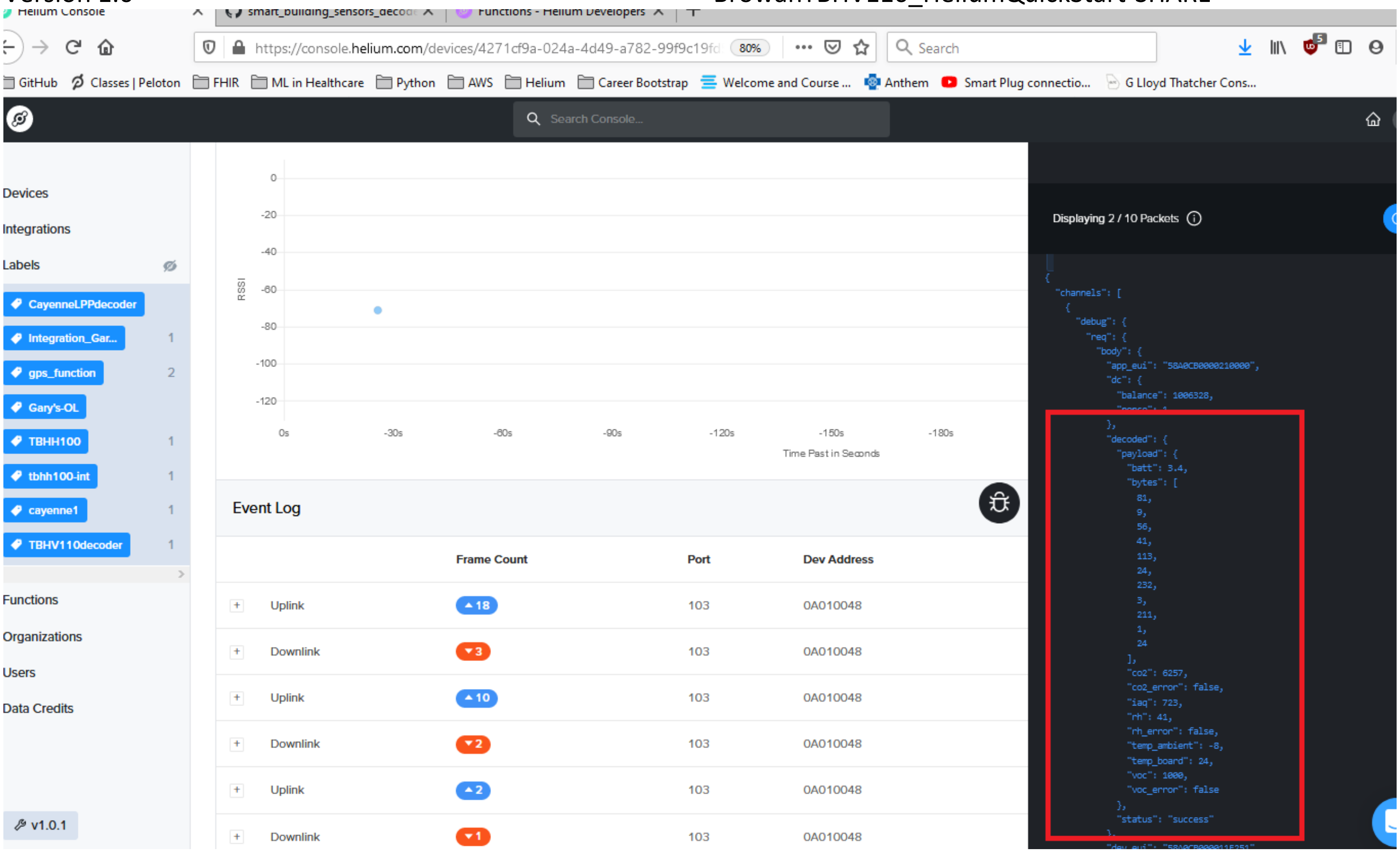
Activation Method OTAA

Attached Labels TBHV110decoder ×

Associated Integrations

Device Integrations

(above) Result of applying the 'TBHV110Decoder' label to the TBHV110 device



Event Log

	Frame Count	Port	Dev Address
+ Uplink	▲18	103	0A010048
+ Downlink	▼3	103	0A010048
+ Uplink	▲10	103	0A010048
+ Downlink	▼2	103	0A010048
+ Uplink	▲2	103	0A010048
+ Downlink	▼1	103	0A010048

```
{
  "channels": [
    {
      "debug": {
        "req": {
          "body": {
            "app_eui": "58A0CB0000210000",
            "dc": {
              "balance": 1006328,
              "nonce": 1
            },
            "decoded": {
              "payload": {
                "batt": 3.4,
                "bytes": [
                  81,
                  9,
                  56,
                  41,
                  113,
                  24,
                  232,
                  3,
                  211,
                  1,
                  24
                ],
                "co2": 6257,
                "co2_error": false,
                "iaq": 723,
                "rh": 41,
                "rh_error": false,
                "temp_ambient": -8,
                "temp_board": 24,
                "voc": 1000,
                "voc_error": false
              },
              "status": "success"
            },
            "dev_eui": "58A0CB000011E251"
```

(above) Looking at the data flowing to the Device with the debugger – and seeing the decoded data

Build a Cayenne Integration

Using the instructions at the following URL, build a Cayenne integration for the device

- https://developer.helium.com/console/integrations/mydevices-cayenne-integration

Here are the steps to go through:

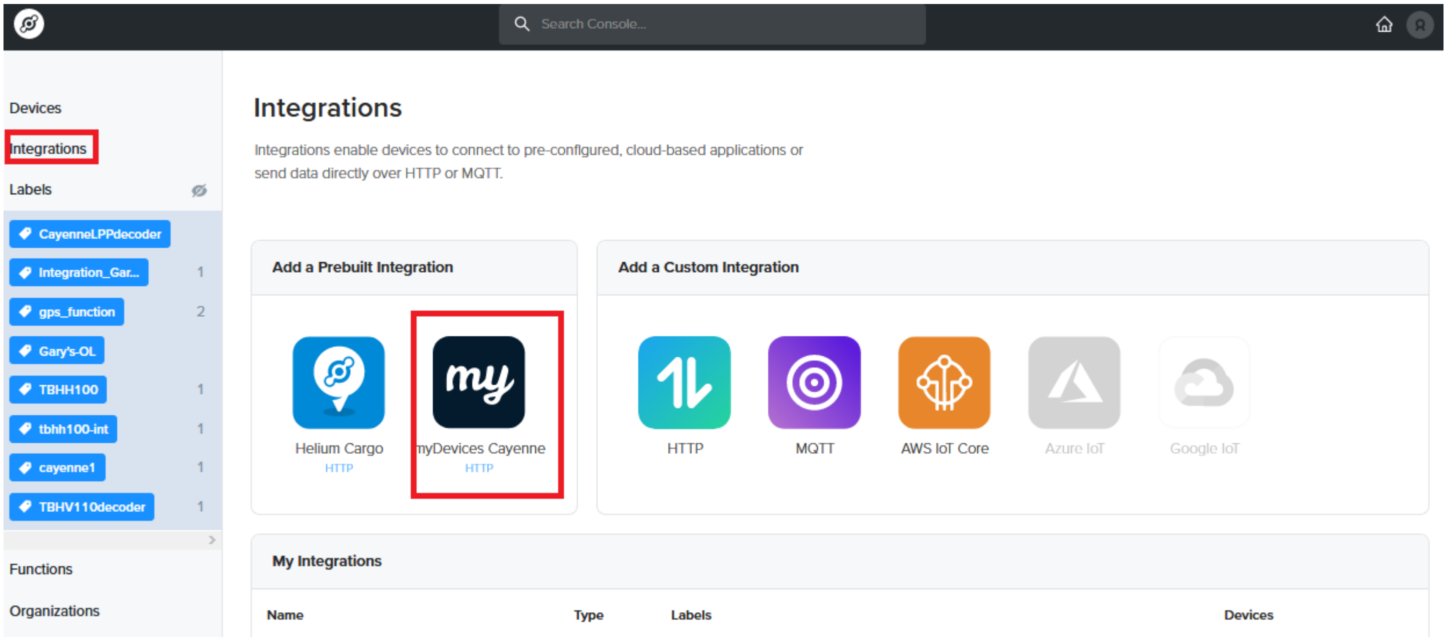
1. In the Helium Console, create a new Cayenne Integration
   a. Name it 'Cayenne-TBHV110'
   b. Create and apply the label called 'Tbhv110-cayenne-int'
2. In the Helium Console, apply the integration label, 'tbhv110-cayenne-int' to the TBHV110 device

TBHV110 to Cayenne Setup Console Settings		
Console Object	**Name**	**Applied Label(s)**
Integration	Cayenne-TBHV110	Tbhv110-cayenne-int
Device	TBHV110	TBHV110decoder Tbhv110-cayenne-int
Function (Decoder)	TBHV110decoderFunction	TBHV110decoder

(above) Helium Console settings for the TBHV110-to-Cayenne Integration

3. In the Cayenne Dashboard, add/create a new Device/Widget
   a. Add new… > Devices & Widgets > Lora > Helium > Tracknet THS-Temperature and Humidity Sensor
   b. Change the default name to 'TBHV110'
   c. Add the DevEUI from the Helium Console
4. Wait for the data to flow into Cayenne

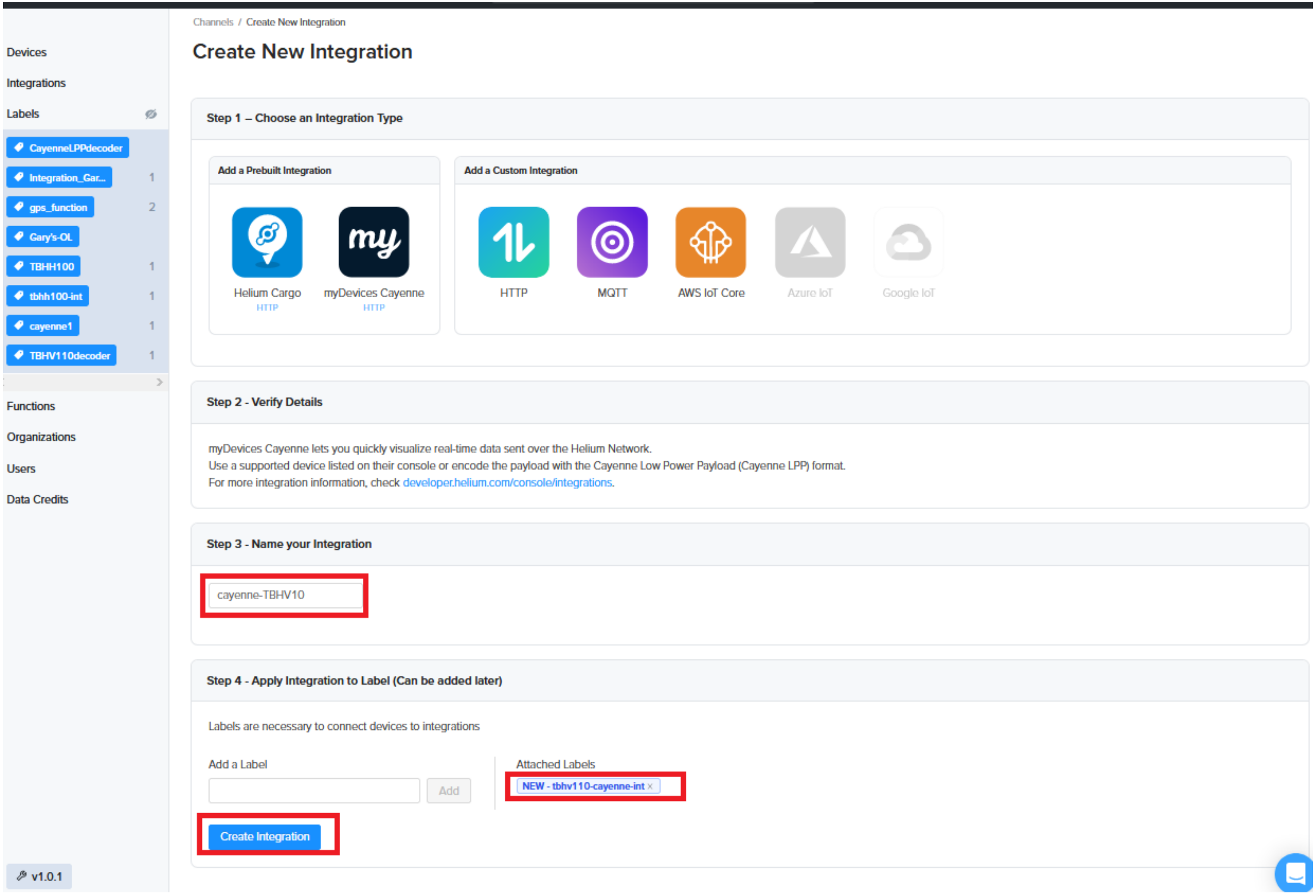
The screenshots below illustrate the steps above.



Integrations

Integrations enable devices to connect to pre-configured, cloud-based applications or send data directly over HTTP or MQTT.

(above) Integrations tab in the Console, then select 'myDevices Cayenne' button



Create New Integration

Step 1 – Choose an Integration Type

Step 2 - Verify Details

myDevices Cayenne lets you quickly visualize real-time data sent over the Helium Network.
Use a supported device listed on their console or encode the payload with the Cayenne Low Power Payload (Cayenne LPP) format.
For more integration information, check developer.helium.com/console/integrations.

Step 3 - Name your Integration

cayenne-TBHV10

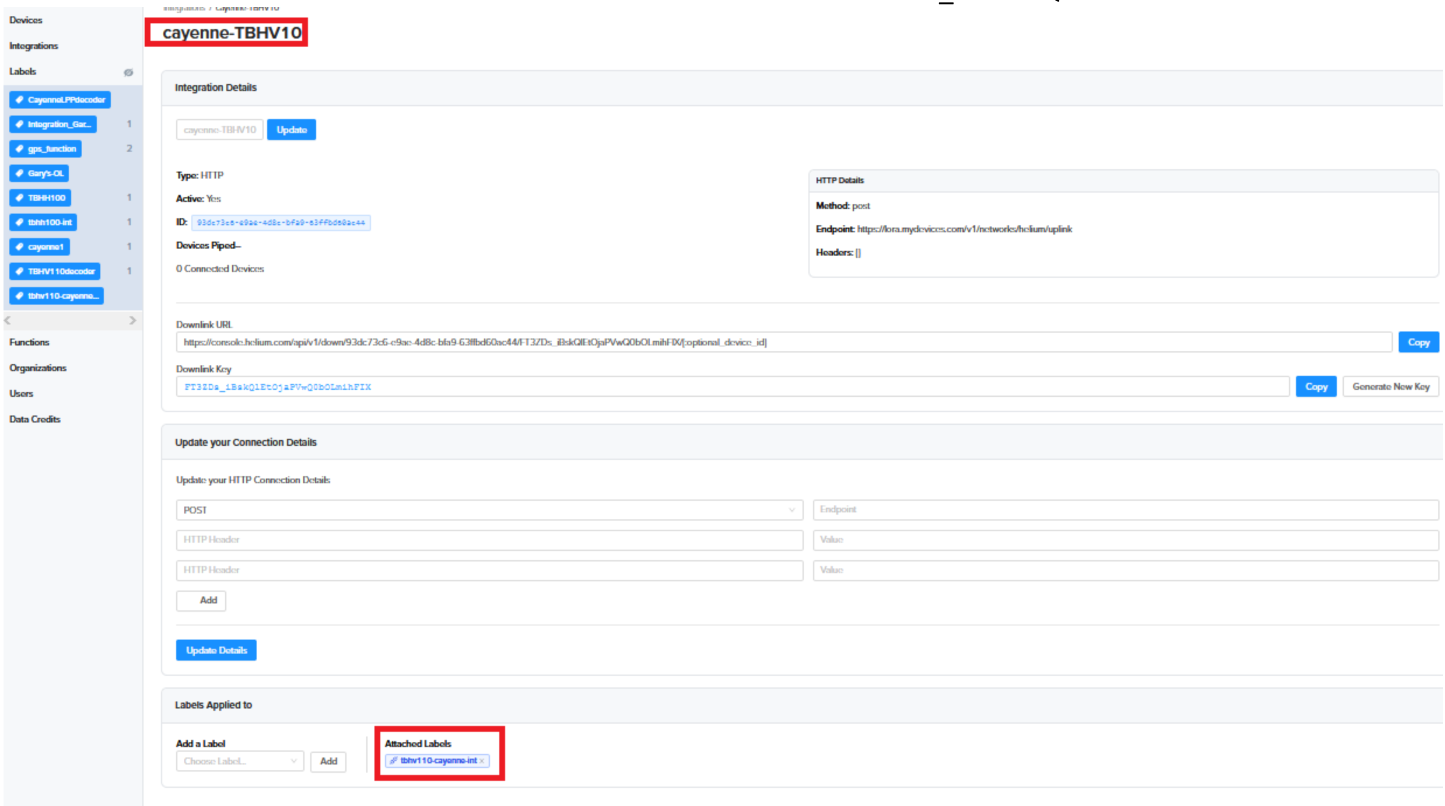
Step 4 - Apply Integration to Label (Can be added later)

Labels are necessary to connect devices to integrations

Attached Labels: NEW - tbhv110-cayenne-int ×

(above) Fill in the Integration Name, assign a new label, then hit the 'Create Integration' button

- Integration Name = 'cayenne-TBHV110'
- Assign a new label called 'tbhv110-cayenne-int'



Integrations / cayenne-TBHV10

cayenne-TBHV10

Integration Details

cayenne-TBHV10

Type: HTTP

Active: Yes

ID: 93dc73c6-c9ac-4d8c-bfa9-63ffbd60ac44

Devices Piped–

0 Connected Devices

HTTP Details

Method: post

Endpoint: https://lora.mydevices.com/v1/networks/helium/uplink

Headers: []

Downlink URL

https://console.helium.com/api/v1/down/93dc73c6-c9ac-4d8c-bfa9-63ffbd60ac44/FT3ZDs_iBskQlEtOjaPVwQ0bOLmihFIX/[:optional_device_id]

Downlink Key

FT3ZDs_iBskQlEtOjaPVwQ0bOLmihFIX

Update your Connection Details

Update your HTTP Connection Details

POST | Endpoint

HTTP Header | Value

HTTP Header | Value

Labels Applied to

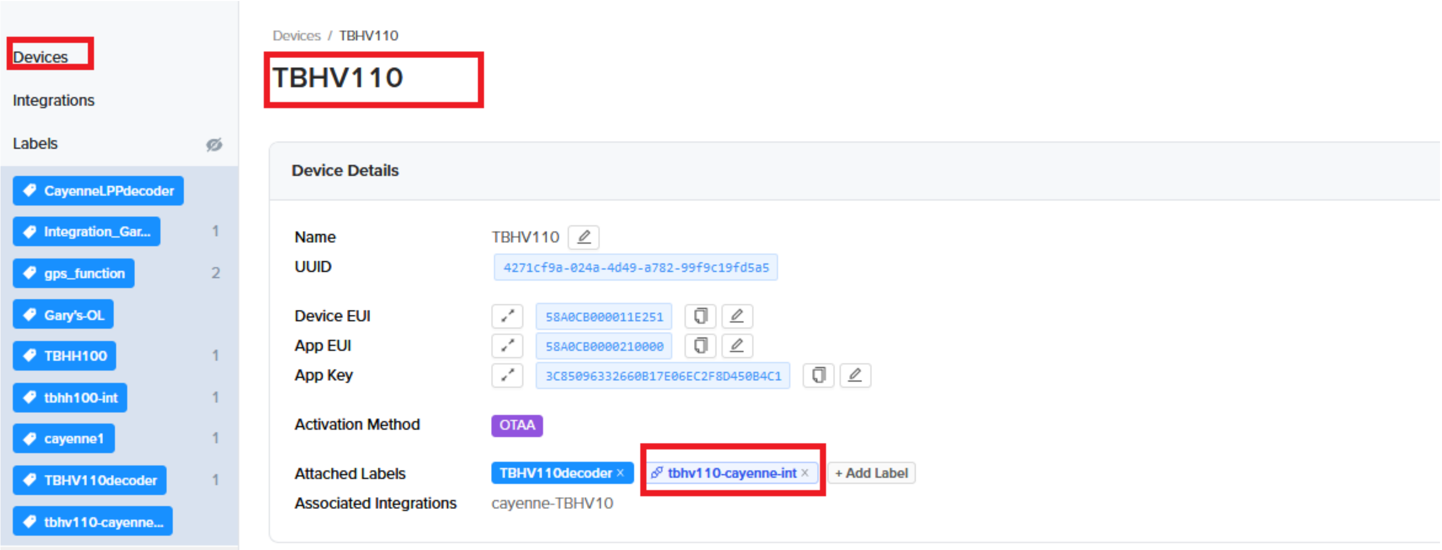
Add a Label

Choose Label...

Attached Labels

tbhv110-cayenne-int ×

(above) the cayenne-TBHV110 integration has been created!



Devices / TBHV110

TBHV110

Device Details

Name TBHV110

UUID 4271cf9a-024a-4d49-a782-99f9c19fd5a5

Device EUI 58A0CB000011E251

App EUI 58A0CB0000210000

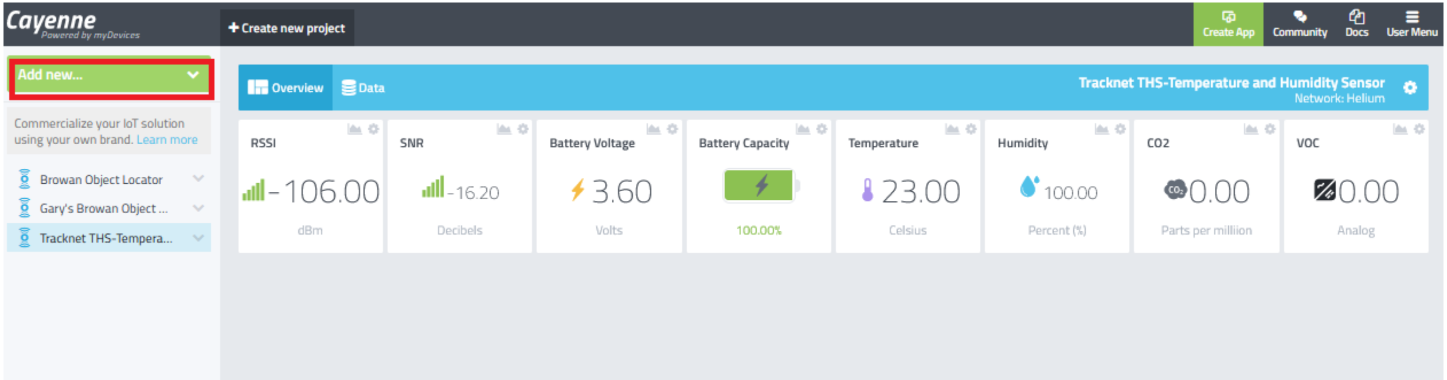
App Key 3C85096332660B17E06EC2F8D450B4C1

Activation Method OTAA

Attached Labels TBHV110decoder × tbhv110-cayenne-int ×

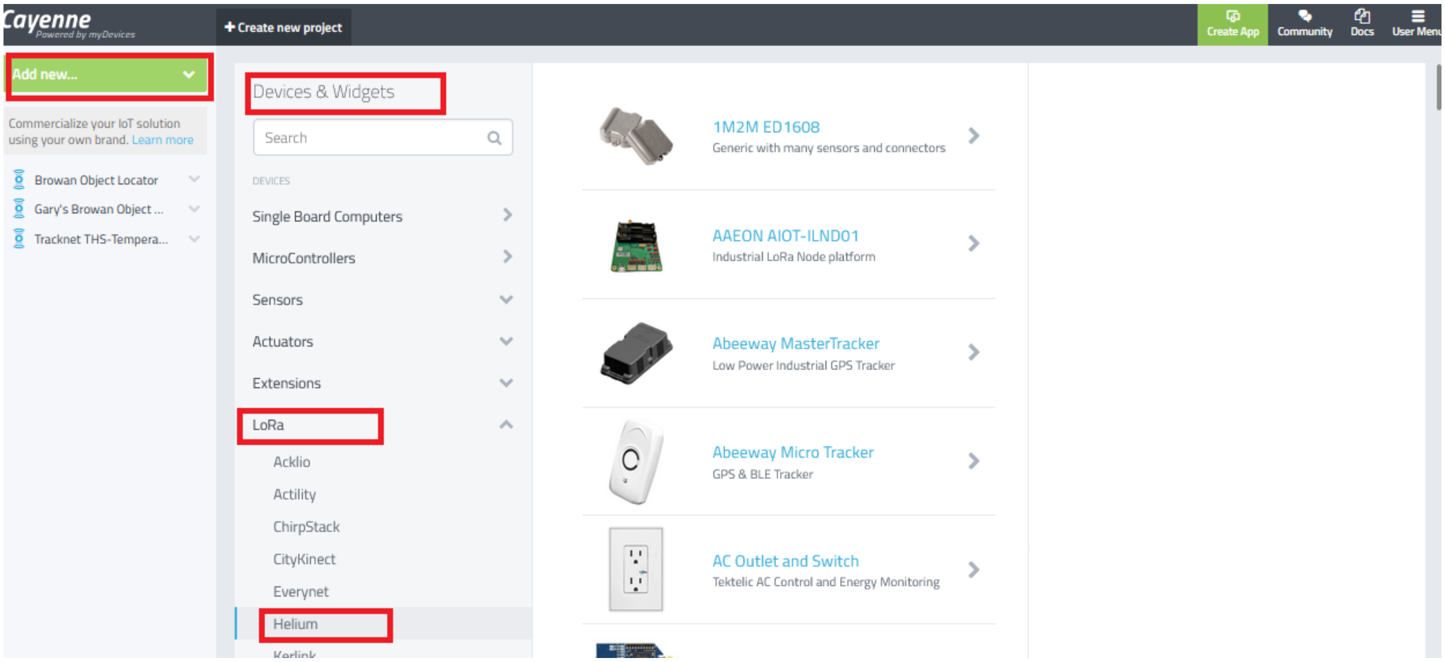
Associated Integrations cayenne-TBHV10

(above) Apply the integration label, 'tbhv110-cayenne-int' to the device

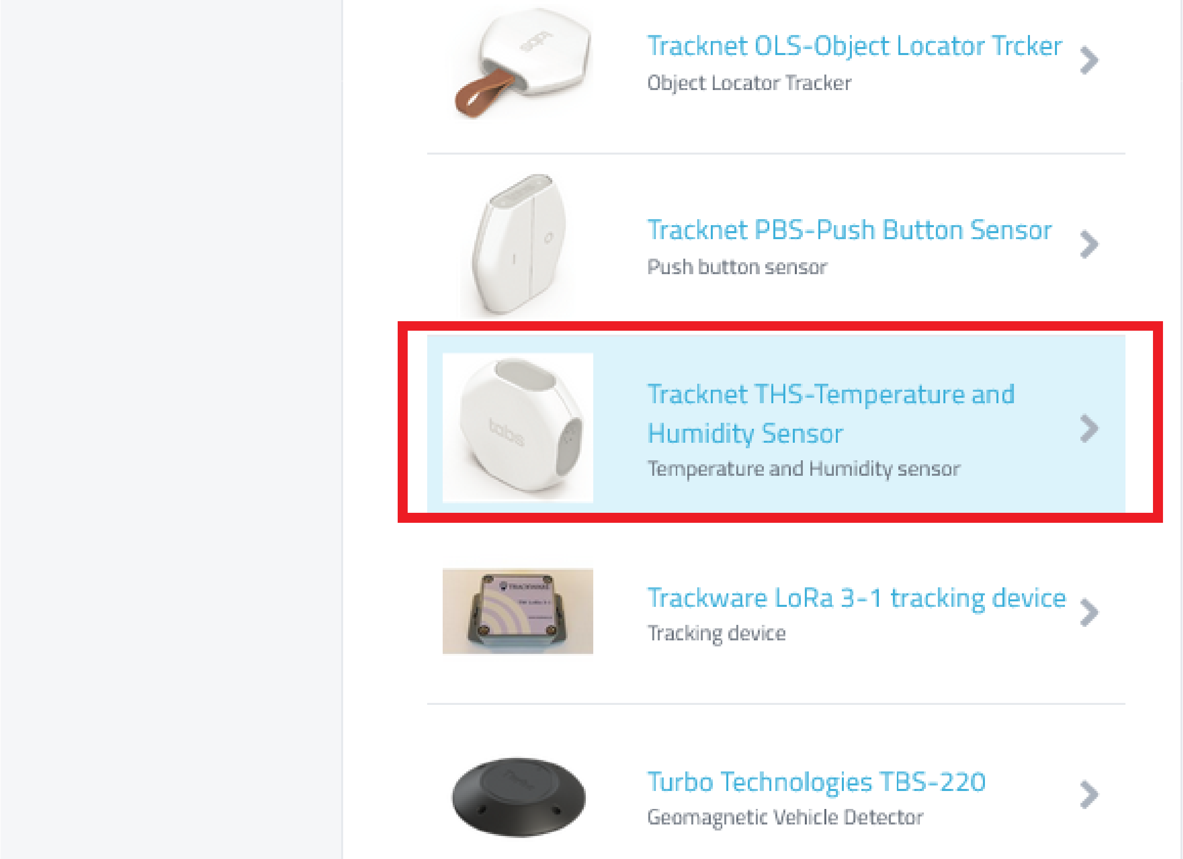


(above) Log into your Cayenne Dashboard and select the 'Add new…' button

Note – The screenshot above happens to show a Tracknet THS-Temperature device. That is from a Helium-Cayenne integration that I set up for my TBHH100 device. We will actually be using that same device type when we set up the TBHV110 in Cayenne. The TBH100 is a TBV110 without the CO2 and VOC sensors. That's why in the screenshot above, there is a 0.00 in both the CO2 and VOC fields.



(above) Add new… > Devices & Widgets > Lora > Helium



Tracknet OLS-Object Locator Trcker

Object Locator Tracker

Tracknet PBS-Push Button Sensor

Push button sensor

Tracknet THS-Temperature and Humidity Sensor

Temperature and Humidity sensor

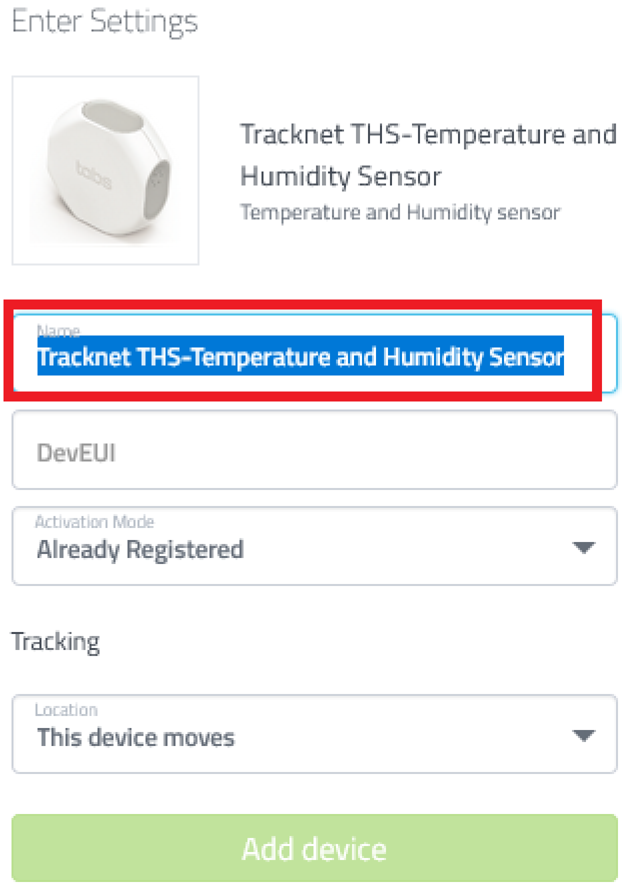
Trackware LoRa 3-1 tracking device

Tracking device

Turbo Technologies TBS-220

Geomagnetic Vehicle Detector

(above) Select the ‘Tracknet THS-Temperature and Humidity Sensor’



Enter Settings

Tracknet THS-Temperature and Humidity Sensor
Temperature and Humidity sensor

Name: Tracknet THS-Temperature and Humidity Sensor

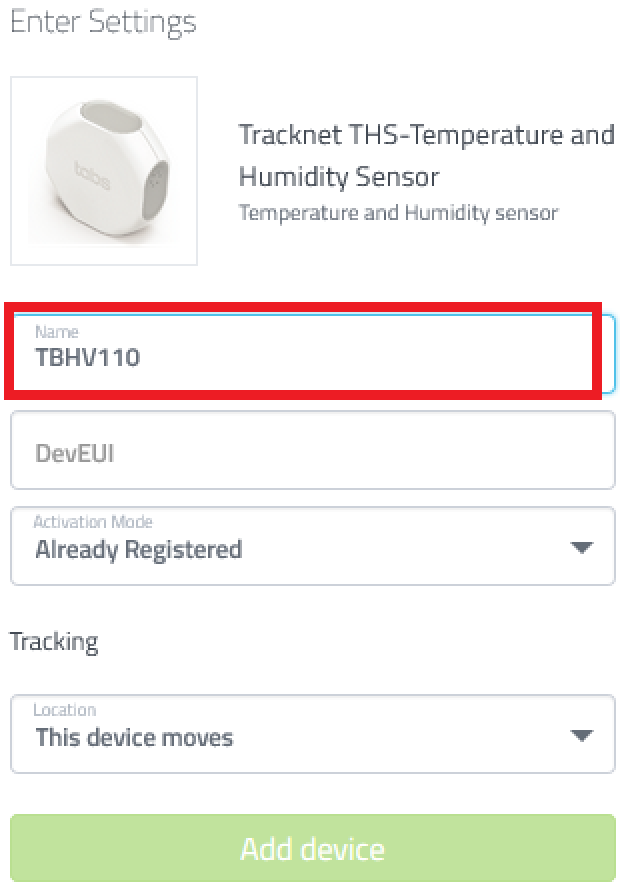
DevEUI

Activation Mode: Already Registered

Tracking

Location: This device moves

(above) Change the default name to ‘TBHV110’



Enter Settings

Tracknet THS-Temperature and Humidity Sensor
Temperature and Humidity sensor

Name: TBHV110

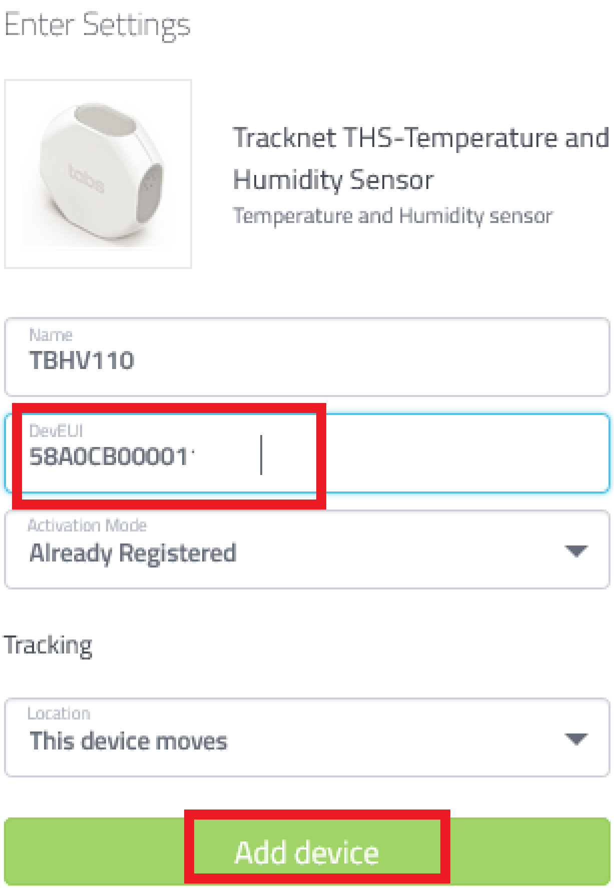
DevEUI

Activation Mode: Already Registered

Tracking

Location: This device moves

(above) Name changed to ‘TBHV110’



Enter Settings

Tracknet THS-Temperature and Humidity Sensor

Temperature and Humidity sensor

Name: TBHV110

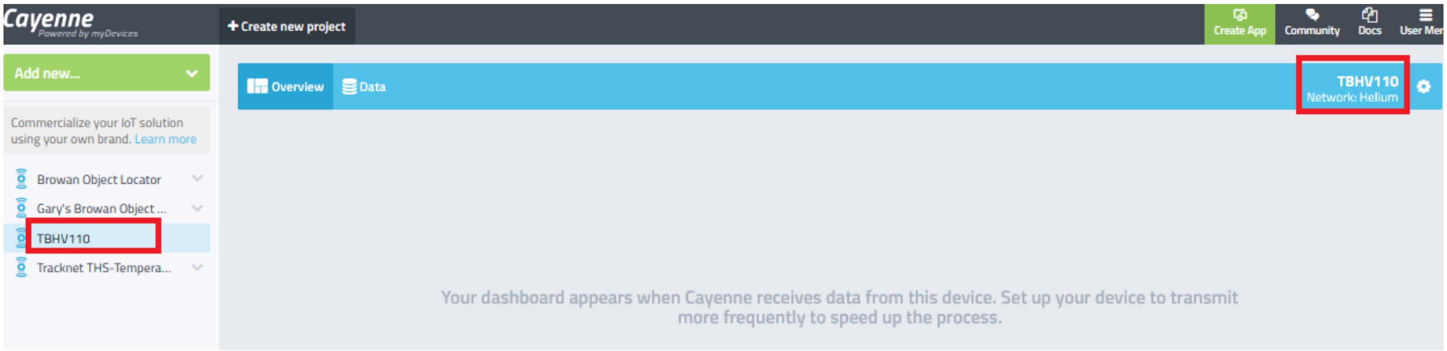
DevEUI: 58A0CB00001'

Activation Mode: Already Registered

Tracking

Location: This device moves

(above) Copy and Paste in the DevEUI information which you can get from the Device tab in the Helium Console, then hit the 'Add device' button

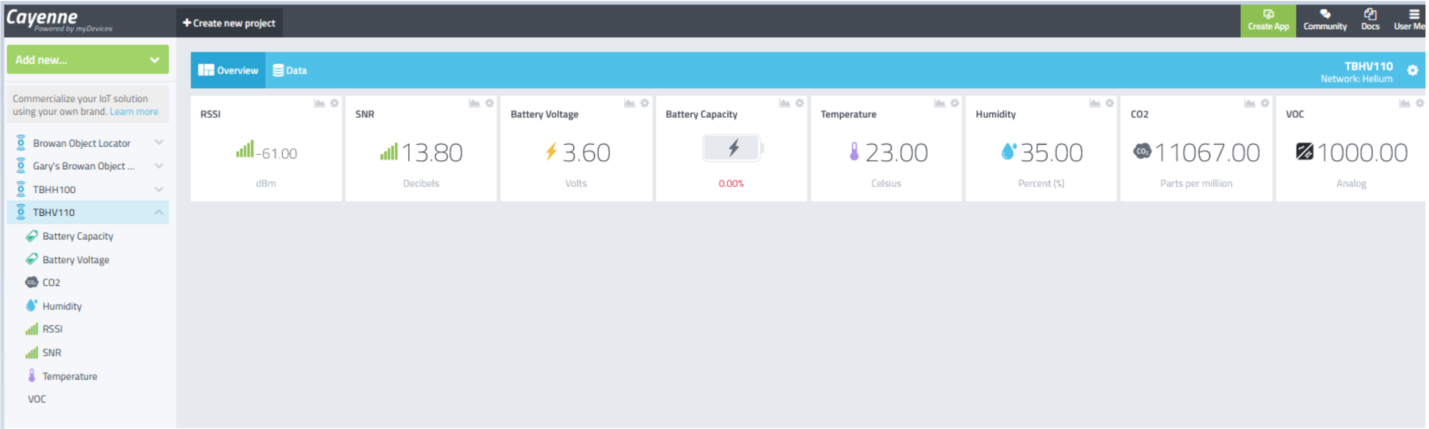


Commercialize your IoT solution using your own brand. Learn more

- Browan Object Locator
- Gary's Browan Object ...
- TBHV110
- Tracknet THS-Tempera...

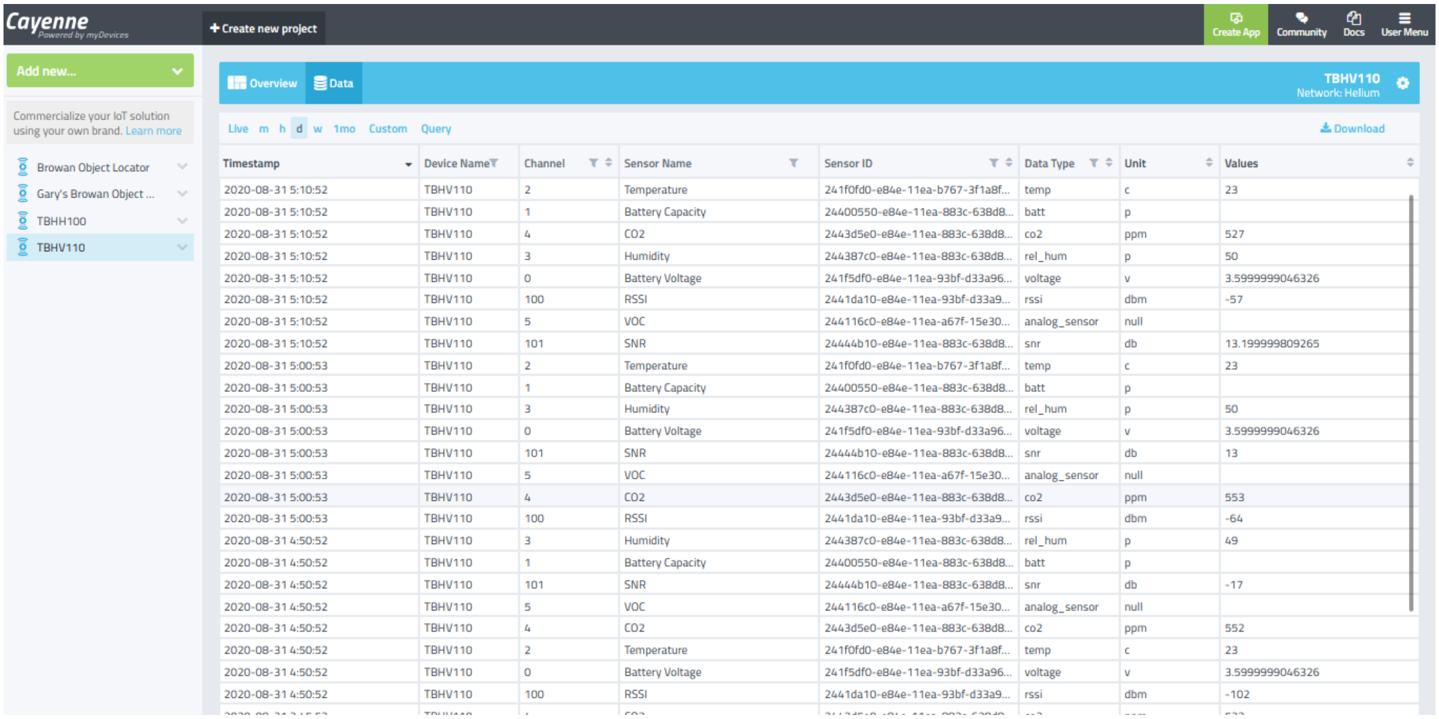
Your dashboard appears when Cayenne receives data from this device. Set up your device to transmit more frequently to speed up the process.

(above) The TBHV110 has been added to your Cayenne dashboard and is waiting for data to come over



RSSI	SNR	Battery Voltage	Battery Capacity	Temperature	Humidity	CO2	VOC
-61.00	13.80	3.60	0.00%	23.00	35.00	11067.00	1000.00
dBm	Decibels	Volts		Celsius	Percent (%)	Parts per milliion	Analog

(above) Cayenne Dashboard showing Overview of data from the TBHV110 – this is the most recent set of received measurements



Timestamp	Device Name	Channel	Sensor Name	Sensor ID	Data Type	Unit	Values
2020-08-31 5:10:52	TBHV110	2	Temperature	241f0fd0-e84e-11ea-b767-3f1a8f...	temp	c	23
2020-08-31 5:10:52	TBHV110	1	Battery Capacity	24400550-e84e-11ea-883c-638d8...	batt	p	
2020-08-31 5:10:52	TBHV110	4	CO2	2443d5e0-e84e-11ea-883c-638d8...	co2	ppm	527
2020-08-31 5:10:52	TBHV110	3	Humidity	244387c0-e84e-11ea-883c-638d8...	rel_hum	p	50
2020-08-31 5:10:52	TBHV110	0	Battery Voltage	241f5df0-e84e-11ea-93bf-d33a96...	voltage	v	3.5999999046326
2020-08-31 5:10:52	TBHV110	100	RSSI	2441da10-e84e-11ea-93bf-d33a9...	rssi	dbm	-57
2020-08-31 5:10:52	TBHV110	5	VOC	244116c0-e84e-11ea-a67f-15e30...	analog_sensor	null	
2020-08-31 5:10:52	TBHV110	101	SNR	24444b10-e84e-11ea-883c-638d8...	snr	db	13.199999809265
2020-08-31 5:00:53	TBHV110	2	Temperature	241f0fd0-e84e-11ea-b767-3f1a8f...	temp	c	23
2020-08-31 5:00:53	TBHV110	1	Battery Capacity	24400550-e84e-11ea-883c-638d8...	batt	p	
2020-08-31 5:00:53	TBHV110	3	Humidity	244387c0-e84e-11ea-883c-638d8...	rel_hum	p	50
2020-08-31 5:00:53	TBHV110	0	Battery Voltage	241f5df0-e84e-11ea-93bf-d33a96...	voltage	v	3.5999999046326
2020-08-31 5:00:53	TBHV110	101	SNR	24444b10-e84e-11ea-883c-638d8...	snr	db	13
2020-08-31 5:00:53	TBHV110	5	VOC	244116c0-e84e-11ea-a67f-15e30...	analog_sensor	null	
2020-08-31 5:00:53	TBHV110	4	CO2	2443d5e0-e84e-11ea-883c-638d8...	co2	ppm	553
2020-08-31 5:00:53	TBHV110	100	RSSI	2441da10-e84e-11ea-93bf-d33a9...	rssi	dbm	-64
2020-08-31 4:50:52	TBHV110	3	Humidity	244387c0-e84e-11ea-883c-638d8...	rel_hum	p	49
2020-08-31 4:50:52	TBHV110	1	Battery Capacity	24400550-e84e-11ea-883c-638d8...	batt	p	
2020-08-31 4:50:52	TBHV110	101	SNR	24444b10-e84e-11ea-883c-638d8...	snr	db	-17
2020-08-31 4:50:52	TBHV110	5	VOC	244116c0-e84e-11ea-a67f-15e30...	analog_sensor	null	
2020-08-31 4:50:52	TBHV110	4	CO2	2443d5e0-e84e-11ea-883c-638d8...	co2	ppm	552
2020-08-31 4:50:52	TBHV110	2	Temperature	241f0fd0-e84e-11ea-b767-3f1a8f...	temp	c	23
2020-08-31 4:50:52	TBHV110	0	Battery Voltage	241f5df0-e84e-11ea-93bf-d33a96...	voltage	v	3.5999999046326
2020-08-31 4:50:52	TBHV110	100	RSSI	2441da10-e84e-11ea-93bf-d33a9...	rssi	dbm	-102

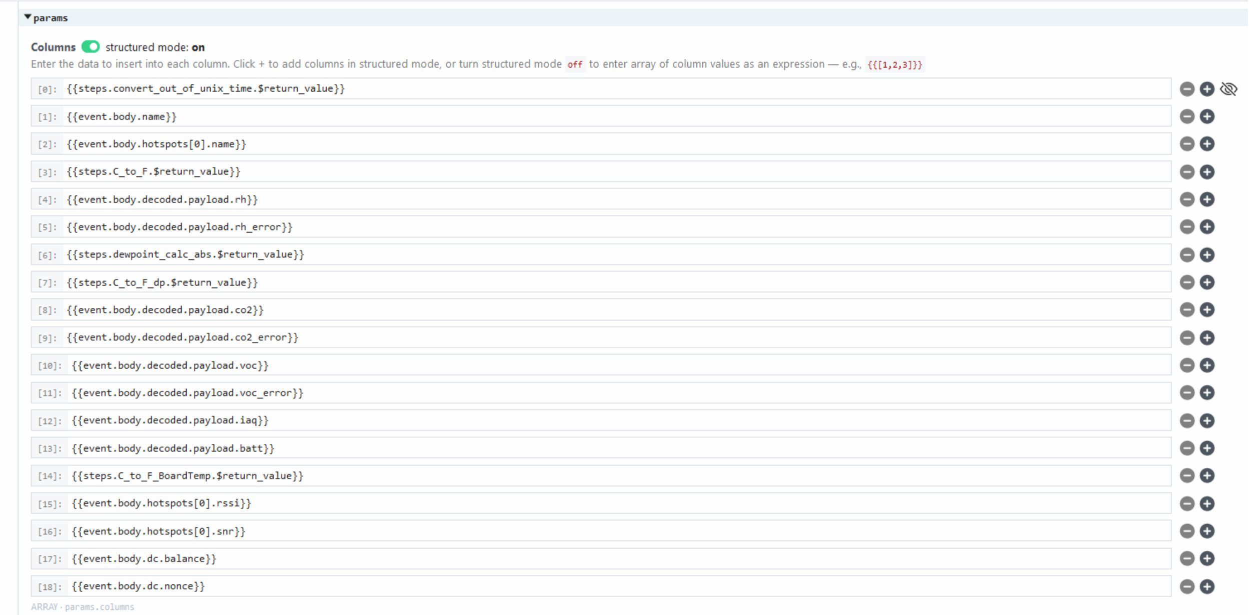
(above) Cayenne Dashboard showing measurements received from the TBHV110

Build a Google Sheet integration

Using the instructions in *HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf* , build the following data flow:

- TBHV110 > Helium Hotspot > Helium Console > Pipedream > Google Sheet

The only difference between the TBHH100 and the TBV110 is that the TBHV110 has a few more data fields, namely CO2, VO, and IAQ, which stands for ‘Indoor Air Quality’ and is a calculated value from the sensor measurements. The TBHV110 reference manual explains this in more detail.



▾params

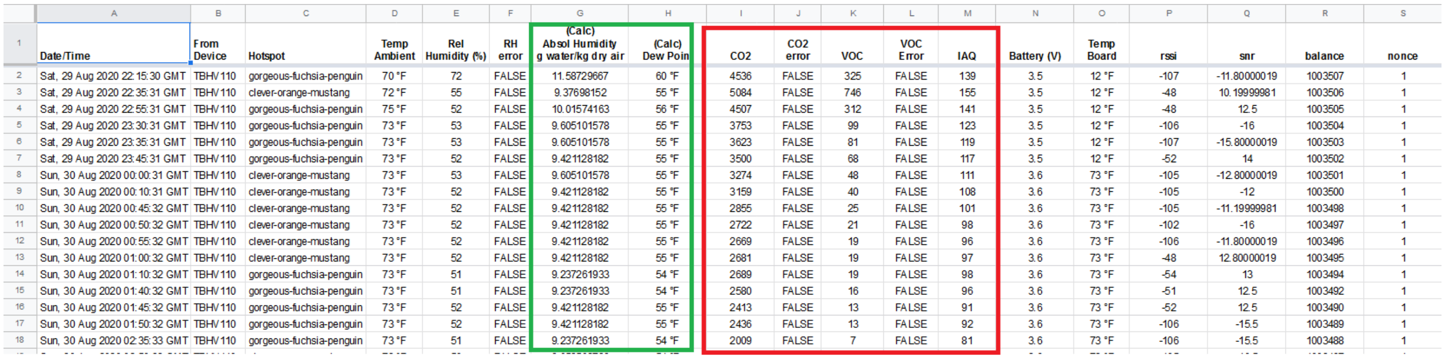
Columns structured mode: **on**

Enter the data to insert into each column. Click + to add columns in structured mode, or turn structured mode `off` to enter array of column values as an expression — e.g., `{{[1,2,3]}}`

```
[0]:  {{steps.convert_out_of_unix_time.$return_value}}
[1]:  {{event.body.name}}
[2]:  {{event.body.hotspots[0].name}}
[3]:  {{steps.C_to_F.$return_value}}
[4]:  {{event.body.decoded.payload.rh}}
[5]:  {{event.body.decoded.payload.rh_error}}
[6]:  {{steps.dewpoint_calc_abs.$return_value}}
[7]:  {{steps.C_to_F_dp.$return_value}}
[8]:  {{event.body.decoded.payload.co2}}
[9]:  {{event.body.decoded.payload.co2_error}}
[10]:  {{event.body.decoded.payload.voc}}
[11]:  {{event.body.decoded.payload.voc_error}}
[12]:  {{event.body.decoded.payload.iaq}}
[13]:  {{event.body.decoded.payload.batt}}
[14]:  {{steps.C_to_F_BoardTemp.$return_value}}
[15]:  {{event.body.hotspots[0].rssi}}
[16]:  {{event.body.hotspots[0].snr}}
[17]:  {{event.body.dc.balance}}
[18]:  {{event.body.dc.nonce}}
```

ARRAY · params.columns

(above) Pipedream workflow – showing the contents of the steps.add_single_row_to_sheet



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Date/Time	From Device	Hotspot	Temp Ambient	Rel Humidity (%)	RH error	(Calc) Absol Humidity g water/kg dry air	(Calc) Dew Poin	CO2	CO2 error	VOC	VOC Error	IAQ	Battery (V)	Temp Board	rssi	snr	balance	nonce
2	Sat, 29 Aug 2020 22:15:30 GMT	TBHV110	gorgeous-fuchsia-penguin	70 °F	72	FALSE	11.58729667	60 °F	4536	FALSE	325	FALSE	139	3.5	12 °F	-107	-11.80000019	1003507	1
3	Sat, 29 Aug 2020 22:35:31 GMT	TBHV110	clever-orange-mustang	72 °F	55	FALSE	9.37698152	55 °F	5084	FALSE	746	FALSE	155	3.5	12 °F	-48	10.19999981	1003506	1
4	Sat, 29 Aug 2020 22:55:31 GMT	TBHV110	gorgeous-fuchsia-penguin	75 °F	52	FALSE	10.01574163	56 °F	4507	FALSE	312	FALSE	141	3.5	12 °F	-48	12.5	1003505	1
5	Sat, 29 Aug 2020 23:30:31 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	53	FALSE	9.605101578	55 °F	3753	FALSE	99	FALSE	123	3.5	12 °F	-106	-16	1003504	1
6	Sat, 29 Aug 2020 23:35:31 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	53	FALSE	9.605101578	55 °F	3623	FALSE	81	FALSE	119	3.5	12 °F	-107	-15.80000019	1003503	1
7	Sat, 29 Aug 2020 23:45:31 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	52	FALSE	9.421128182	55 °F	3500	FALSE	68	FALSE	117	3.5	12 °F	-52	14	1003502	1
8	Sun, 30 Aug 2020 00:00:31 GMT	TBHV110	clever-orange-mustang	73 °F	53	FALSE	9.605101578	55 °F	3274	FALSE	48	FALSE	111	3.6	73 °F	-105	-12.80000019	1003501	1
9	Sun, 30 Aug 2020 00:10:31 GMT	TBHV110	clever-orange-mustang	73 °F	52	FALSE	9.421128182	55 °F	3159	FALSE	40	FALSE	108	3.6	73 °F	-105	-12	1003500	1
10	Sun, 30 Aug 2020 00:45:32 GMT	TBHV110	clever-orange-mustang	73 °F	52	FALSE	9.421128182	55 °F	2855	FALSE	25	FALSE	101	3.6	73 °F	-105	-11.19999981	1003498	1
11	Sun, 30 Aug 2020 00:50:32 GMT	TBHV110	clever-orange-mustang	73 °F	52	FALSE	9.421128182	55 °F	2722	FALSE	21	FALSE	98	3.6	73 °F	-102	-16	1003497	1
12	Sun, 30 Aug 2020 00:55:32 GMT	TBHV110	clever-orange-mustang	73 °F	52	FALSE	9.421128182	55 °F	2669	FALSE	19	FALSE	96	3.6	73 °F	-106	-11.80000019	1003496	1
13	Sun, 30 Aug 2020 01:00:32 GMT	TBHV110	clever-orange-mustang	73 °F	52	FALSE	9.421128182	55 °F	2681	FALSE	19	FALSE	97	3.6	73 °F	-48	12.80000019	1003495	1
14	Sun, 30 Aug 2020 01:10:32 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	51	FALSE	9.237261933	54 °F	2689	FALSE	19	FALSE	98	3.6	73 °F	-54	13	1003494	1
15	Sun, 30 Aug 2020 01:40:32 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	51	FALSE	9.237261933	54 °F	2580	FALSE	16	FALSE	96	3.6	73 °F	-51	12.5	1003492	1
16	Sun, 30 Aug 2020 01:45:32 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	52	FALSE	9.421128182	55 °F	2413	FALSE	13	FALSE	91	3.6	73 °F	-52	12.5	1003490	1
17	Sun, 30 Aug 2020 01:50:32 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	52	FALSE	9.421128182	55 °F	2436	FALSE	13	FALSE	92	3.6	73 °F	-106	-15.5	1003489	1
18	Sun, 30 Aug 2020 02:35:33 GMT	TBHV110	gorgeous-fuchsia-penguin	73 °F	51	FALSE	9.237261933	54 °F	2009	FALSE	7	FALSE	81	3.6	73 °F	-106	-15.5	1003488	1

(above) Data from the TBHV110 in a Google Sheet. The data in columns I-M are data that the TBHV110 has but the TBHH100 does not. The data in columns G and H is calculated (in a Pipedream workflow step) from the data in columns D and E.

The HowTo instructions can be found in GitHub here:
https://github.com/mikedsp/helium/blob/master/MyDocuments/HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf

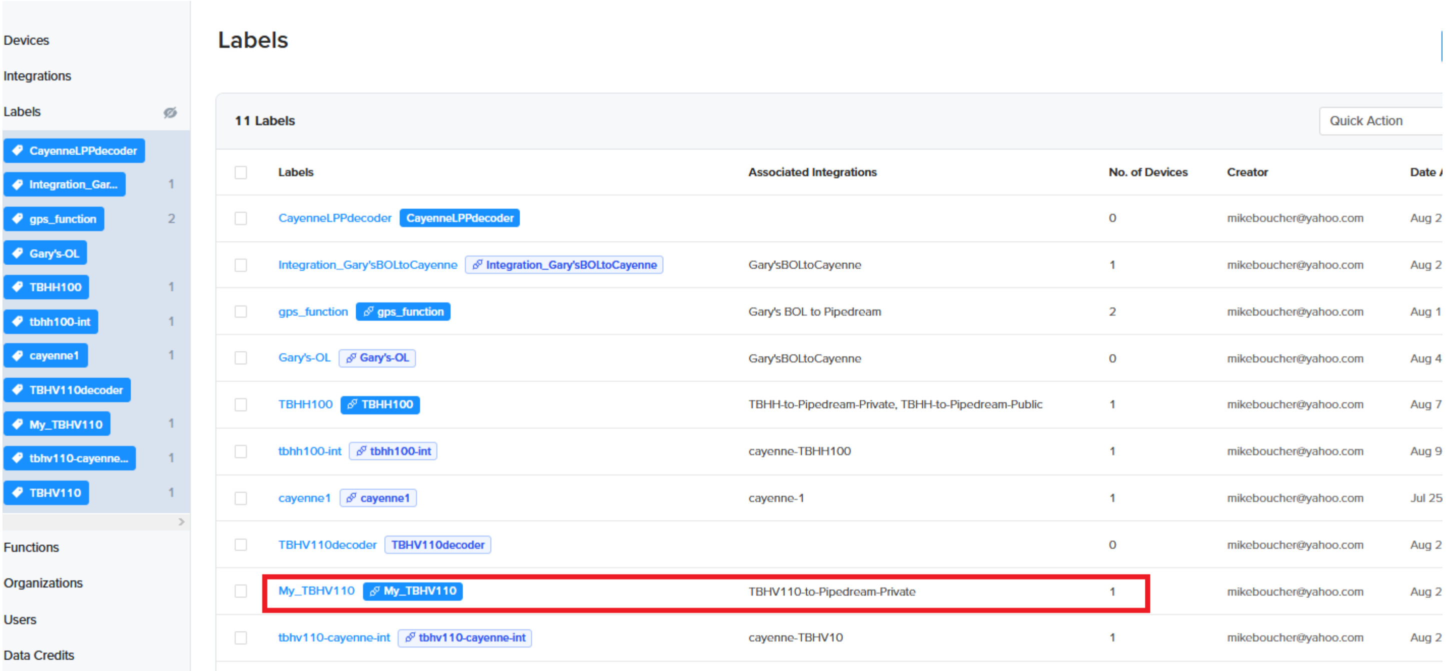
An Excel file with some collected data can be found in GitHub here:
https://github.com/mikedsp/helium/blob/master/MyDocuments/20200831_TBHV110%20data.xlsx

A public version of the Pipedream workflow can be found here:
https://pipedream.com/@dangermikeb/tbhv110-to-google-sheet-private-p_ezCn1j
Note – the shared workflow at the link above does not show the parameters being pulled from the event and steps and being populated into the Google Sheet step. See the Pipedream workflow screenshot above for what that looks like.

Very Important: When creating the Custom HTTP Integration to send data to Pipedream, it is critical that the label you apply to the Integration be created BEFORE you attach the label to the Integration. If you create the label for the Integration while you are creating the Integration, and then apply that label to the TBHV110 device, Pipedream will NOT get decoded data. I made this mistake when building the TBHH100 and the TBHV110 integrations with Pipedream.

Debugging Tips

If you're working on your Pipedream workflow and trigger data is being received in the workflow, but you can't see or find the readings returned from the Decoder function you created in the Helium Console, then you most likely have your labels wrong. You'll see the decoded values in the Helium Console debugger when you look at your TBHV110 device so you'll know your decode is working, but the sensor values are not showing up in the Workflow event or trigger. If you're in this state, look very closely at your label assignments in the Console.

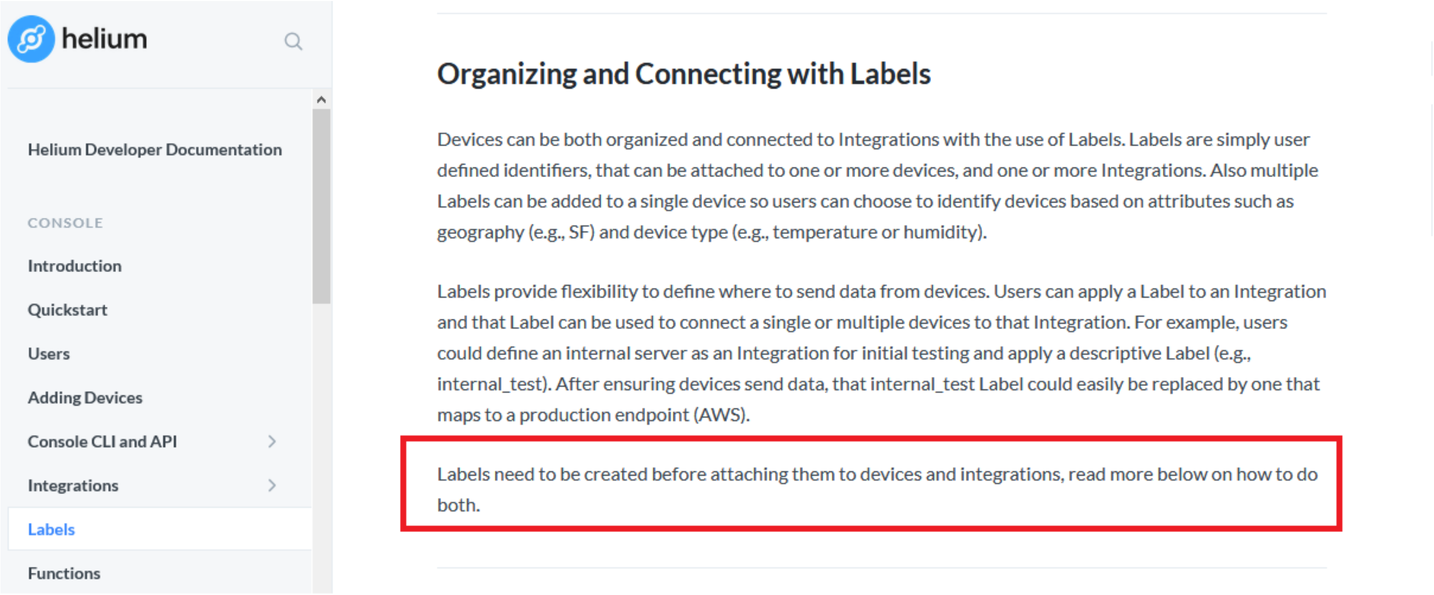


Labels

11 Labels

□	Labels	Associated Integrations	No. of Devices	Creator	Date /
□	CayenneLPPdecoder CayenneLPPdecoder		0	mikeboucher@yahoo.com	Aug 2
□	Integration_Gary'sBOLtoCayenne Integration_Gary'sBOLtoCayenne	Gary'sBOLtoCayenne	1	mikeboucher@yahoo.com	Aug 2
□	gps_function gps_function	Gary's BOL to Pipedream	2	mikeboucher@yahoo.com	Aug 1
□	Gary's-OL Gary's-OL	Gary'sBOLtoCayenne	0	mikeboucher@yahoo.com	Aug 4
□	TBHH100 TBHH100	TBHH-to-Pipedream-Private, TBHH-to-Pipedream-Public	1	mikeboucher@yahoo.com	Aug 7
□	tbhh100-int tbhh100-int	cayenne-TBHH100	1	mikeboucher@yahoo.com	Aug 9
□	cayenne1 cayenne1	cayenne-1	1	mikeboucher@yahoo.com	Jul 25
□	TBHV110decoder TBHV110decoder		0	mikeboucher@yahoo.com	Aug 2
□	My_TBHV110 My_TBHV110	TBHV110-to-Pipedream-Private	1	mikeboucher@yahoo.com	Aug 2
□	tbhv110-cayenne-int tbhv110-cayenne-int	cayenne-TBHV10	1	mikeboucher@yahoo.com	Aug 2

(above) Looking at Labels in the Helium Console. The label, 'My_TBHV110' is the one that I am using to get decoded data to my Pipedream workflow. The label above that is the one I tried first, but notice how it is not associated with an Integration.



Organizing and Connecting with Labels

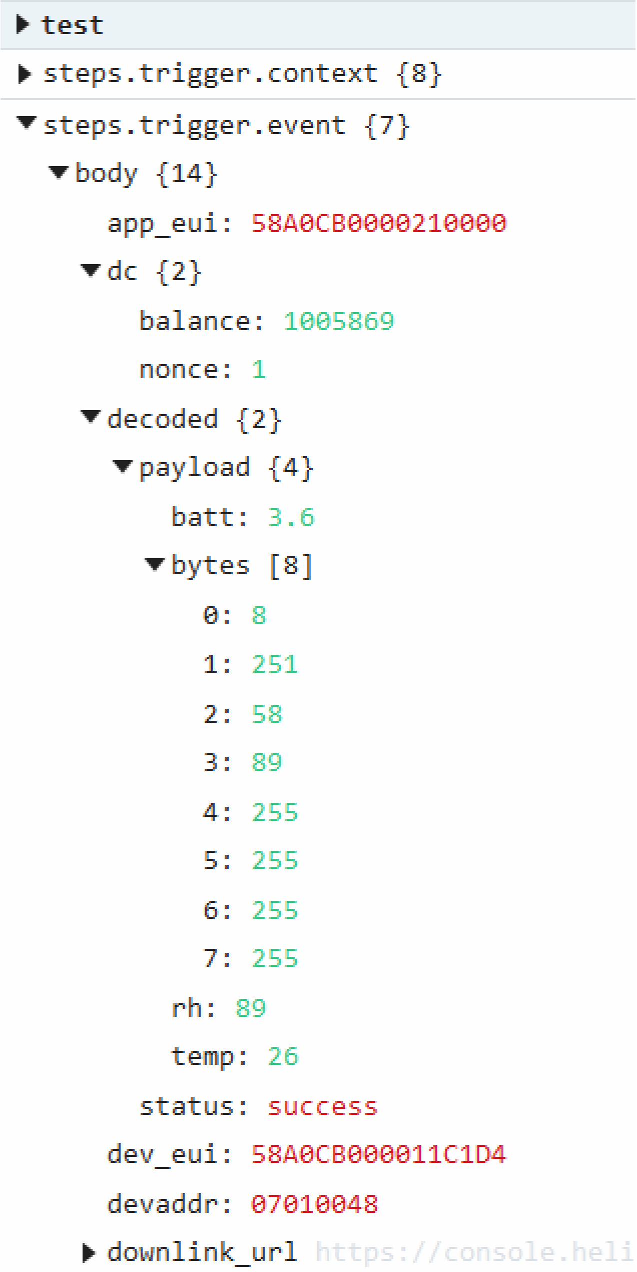
Devices can be both organized and connected to Integrations with the use of Labels. Labels are simply user defined identifiers, that can be attached to one or more devices, and one or more Integrations. Also multiple Labels can be added to a single device so users can choose to identify devices based on attributes such as geography (e.g., SF) and device type (e.g., temperature or humidity).

Labels provide flexibility to define where to send data from devices. Users can apply a Label to an Integration and that Label can be used to connect a single or multiple devices to that Integration. For example, users could define an internal server as an Integration for initial testing and apply a descriptive Label (e.g., internal_test). After ensuring devices send data, that internal_test Label could easily be replaced by one that maps to a production endpoint (AWS).

Labels need to be created before attaching them to devices and integrations, read more below on how to do both.

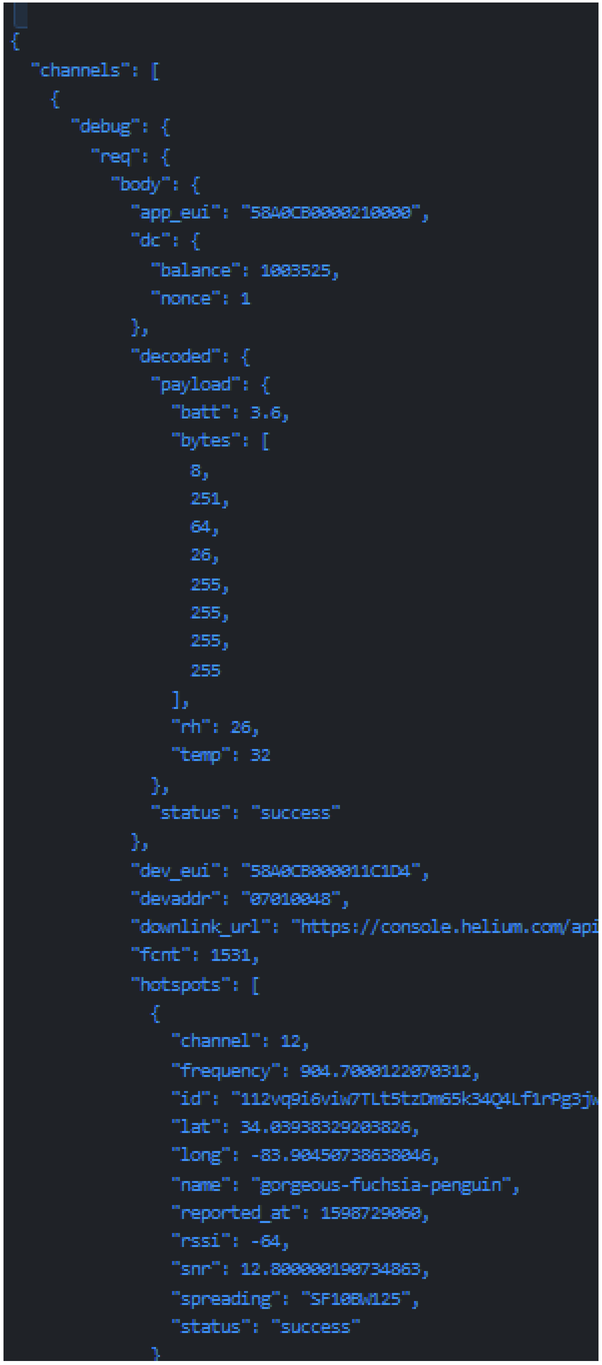
(above) Screenshot of the Helium Label documentation – notice the warning about creating labels before attaching them.

The 2 side-by-side screenshots below compare sensor data views in the Pipedream workflow and in the Helium Console debugger (note – rh and temp values are different because the reading are from different times). If in your Pipedream workflow you see decoded payload data, then your Decoder function in the Helium Console is properly applied to the Pipedream integration. If you cannot see decoded payload data in your Pipedream workflow trigger event, check your labels. The screenshots below are for the TBHH100 device, but the concept is the same for the TBHV110.



```
▶ test
▶ steps.trigger.context {8}
▼ steps.trigger.event {7}
  ▼ body {14}
      app_eui: 58A0CB0000210000
    ▼ dc {2}
        balance: 1005869
        nonce: 1
    ▼ decoded {2}
      ▼ payload {4}
          batt: 3.6
        ▼ bytes [8]
            0: 8
            1: 251
            2: 58
            3: 89
            4: 255
            5: 255
            6: 255
            7: 255
          rh: 89
          temp: 26
        status: success
      dev_eui: 58A0CB000011C1D4
      devaddr: 07010048
    ▶ downlink_url https://console.heli
```

(above) TBHH100 in Pipedream



```
{
  "channels": [
    {
      "debug": {
        "req": {
          "body": {
            "app_eui": "58A0CB0000210000",
            "dc": {
              "balance": 1003525,
              "nonce": 1
            },
            "decoded": {
              "payload": {
                "batt": 3.6,
                "bytes": [
                  8,
                  251,
                  64,
                  26,
                  255,
                  255,
                  255,
                  255
                ],
                "rh": 26,
                "temp": 32
              },
              "status": "success"
            },
            "dev_eui": "58A0CB000011C1D4",
            "devaddr": "07010048",
            "downlink_url": "https://console.helium.com/api
            "fcnt": 1531,
            "hotspots": [
              {
                "channel": 12,
                "frequency": 904.7000122070312,
                "id": "112vq9i6viw7TLt5tzDm65k34Q4Lf1rPg3jw
                "lat": 34.03938329203826,
                "long": -83.90450738638046,
                "name": "gorgeous-fuchsia-penguin",
                "reported_at": 1598729060,
                "rssi": -64,
                "snr": 12.800000190734863,
                "spreading": "SF10BW125",
                "status": "success"
              }
```

(above) TBHH100 in Console debugger

Appendix A – JSON Data from Console Debugger

The JSON structure below was copied from the Helium Debugger once the TBHV110 was successfully added to the Console and a decoder function was applied.

```
{
  "channels": [
    {
      "debug": {
        "req": {
          "body": {
            "app_eui": "58A0CB0000210000",
            "dc": {
              "balance": 1006185,
              "nonce": 1
            },
            "decoded": {
              "payload": {
                "batt": 3.6,
                "bytes": [
                  0,
                  11,
                  55,
                  57,
                  244,
                  1,
                  0,
                  0,
                  25,
                  0,
                  55
                ],
                "co2": 500,
                "co2_error": false,
                "iaq": 25,
                "rh": 57,
                "rh_error": false,
                "temp_ambient": 23,
                "temp_board": 23,
                "voc": 0,
                "voc_error": false
              },
              "status": "success"
            },
            "dev_eui": "58A0CB000011E251",
            "devaddr": "0A010048",
            "fcnt": 50,
            "hotspots": [
              {
                "channel": 65,
                "frequency": 904.5999755859375,
                "id": "112kk7sLkuPybrPDE4ZPAYcAXzuPZbV3F2MH5adatdGohmRX5zJW",
                "lat": 34.04490799366432,
                "long": -83.90869057221556,
                "name": "clever-orange-mustang",
                "reported_at": 1598450375,
                "rssi": -37,
                "snr": 12.199999809265137,
                "spreading": "SF8BW500",
                "status": "success"
              }
            ],
            "id": "4271cf9a-024a-4d49-a782-99f9c19fd5a5",
            "metadata": {
              "labels": [
                {
                  "id": "85e80d87-4d40-4b82-ba0a-993d13860c42",
                  "name": "TBHV110decoder",
                  "organization_id": "27706bf3-96b3-4ccf-ae94-439c44858866"
                }
              ],
              "organization_id": "27706bf3-96b3-4ccf-ae94-439c44858866"
            },
            "name": "TBHV110",
            "payload": "AAs3OfQBAAAZADc=",
            "port": 103,
            "reported_at": 1598450375
          }
        }
      },
      "description": "console debug",
      "id": "no_integration_id",
      "name": "Console Debug Integration",
      "status": "success"
    }
  ],
  "devaddr": "0A010048",
  "device_name": "TBHV110",
  "frame_down": 7,
  "frame_up": 50,
  "hotspots": [
    {
      "frequency": 904.5999755859375,
      "id": "112kk7sLkuPybrPDE4ZPAYcAXzuPZbV3F2MH5adatdGohmRX5zJW",
      "name": "clever-orange-mustang",
      "rssi": -37,
      "snr": 12.199999809265137,
      "spreading": "SF8BW500"
    }
  ],
  "id": "5b0f480b-9ef1-4cb0-a835-fd1600159b37",
  "payload": "AAs3OfQBAAAZADc=",
  "payload_size": 11,
  "port": 103
}
{
  "channels": [],
  "devaddr": "0A010048",
  "device_name": "TBHV110",
  "frame_down": 7,
  "frame_up": 50,
  "hotspots": [
    {
      "frequency": 923.9,
      "id": "112kk7sLkuPybrPDE4ZPAYcAXzuPZbV3F2MH5adatdGohmRX5zJW",
      "name": "clever-orange-mustang",
      "rssi": 27,
      "snr": 0,
      "spreading": "SF7BW500"
    }
  ],
  "id": "a6f1c0d4-19e3-49e1-b555-e6fca124f912",
  "payload": null,
  "payload_size": 0,
  "port": 103
}
{
  "channels": [
    {
      "debug": {
        "req": {
          "body": {
            "app_eui": "58A0CB0000210000",
            "dc": {
              "balance": 1006186,
              "nonce": 1
            },
            "decoded": {
              "payload": {
                "batt": 3.6,
                "bytes": [
                  0,
                  11,
                  17,
                  75,
                  54,
                  2,
                  0,
                  0,
                  37,
                  0,
                53
              ],
              "co2": 566,
              "co2_error": false,
              "iaq": 37,
              "rh": 75,
              "rh_error": false,
              "temp_ambient": 21,
              "temp_board": -15,
              "voc": 0,
              "voc_error": false
            },
            "status": "success"
          },
          "dev_eui": "58A0CB000011E251",
          "devaddr": "0A010048",
          "fcnt": 42,
          "hotspots": [
            {
              "channel": 65,
              "frequency": 904.5999755859375,
              "id": "112vq9i6viw7TLt5tzDm65k34Q4Lf1rPg3jwgYHd9CVxwadcNW4g",
              "lat": 34.03938329203826,
              "long": -83.90450738638046,
              "name": "gorgeous-fuchsia-penguin",
              "reported_at": 1598447974,
              "rssi": -47,
              "snr": 12,
              "spreading": "SF8BW500",
              "status": "success"
            }
          ],
          "id": "4271cf9a-024a-4d49-a782-99f9c19fd5a5",
          "metadata": {
            "labels": [
              {
                "id": "85e80d87-4d40-4b82-ba0a-993d13860c42",
                "name": "TBHV110decoder",
                "organization_id": "27706bf3-96b3-4ccf-ae94-439c44858866"
              }
            ],
            "organization_id": "27706bf3-96b3-4ccf-ae94-439c44858866"
          },
          "name": "TBHV110",
          "payload": "AAsRSzYCAAAlADU=",
          "port": 103,
          "reported_at": 1598447974
        }
      }
    },
      "description": "console debug",
      "id": "no_integration_id",
      "name": "Console Debug Integration",
      "status": "success"
    }
  ],
  "devaddr": "0A010048",
  "device_name": "TBHV110",
  "frame_down": 6,
  "frame_up": 42,
  "hotspots": [
    {
      "frequency": 904.5999755859375,
      "id": "112vq9i6viw7TLt5tzDm65k34Q4Lf1rPg3jwgYHd9CVxwadcNW4g",
      "name": "gorgeous-fuchsia-penguin",
      "rssi": -47,
      "snr": 12,
      "spreading": "SF8BW500"
    }
  ],
  "id": "d135192e-787b-4ffd-8255-dde5b84688c7",
  "payload": "AAsRSzYCAAAlADU=",
  "payload_size": 11,
  "port": 103
```