Executive Summary

This document walks through the steps to get a Radio Bridge Indoor Temperature and Humidity Sensor (RBS305-ATH-US) connected to the Helium network, connected to Cayenne myDevices, and capturing data in a Google Sheet via a Pipedream workflow. The document also covers how to change the mode of the sensor from the default mode (triggering on temperature or humidity crossing a threshold) to triggering on a change in temperature or humidity.

Steps

1. Add the device to the console and get data flowing
2. Apply a decoder function to the device in the Helium Console so you can see/access the sensor data in the payload
3. Change the mode of the sensor
4. Build a Cayenne myDevices Integration so you can see the sensor data in Cayenne
5. Build a Google Sheet integration so you can see/log/share the sensor data in a spreadsheet

Table of Contents

Contents

References

ID	Topic	Reference	Description
1	Radio Bridge	https://radiobridge.com/products/wireless-air-temperature-and-humidity-sensor 1. How to Connect LoRaWAN Sensors.pdf 2. Wireless Air Temp and Humidity Sensor User Guide.pdf 3. Common Sensor Messages.pdf 4. https://github.com/RadioBridge/Packet-	Documentation Download page 1. Lora Connection Guide 2. User Guide for the RBS305-ATH-US 3. User Guide for sensor messages common to all Radio Bridge sensors 4. Decoder function source code 5. Support Article on Downlink

Author – Mike Boucher, mikeboucher@yahoo.com

		Decoder/blob/master/radio_bridge_packet_decoder.js 5. https://support.radiobridge.com/portal/en/kb/articles/how-do-i-create-a-downlink-configuration-for-an-air-temperature-and-humidity-sensor	messages for Radio Bridge Sensor RBS305-ATH-US
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 5. https://developer.helium.com/console/integrations/http#downlink-tool-example	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 5. Using the Helium Downlink tool
3	Pipedream	1. https://github.com/mikedsp/helium/blob/master/MyDocuments/HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf 2. https://pipedream.com/@dangermikeb/pipedreamint_rbs305-p_OKCdkK	1. HowTo Guide - How to get data from a Browan TBHH100 temperature and humidity sensor to flow in real time to a Google Sheet. 2. Mike's Pipedream workflow for the RBS305-AUTH-US sensor
4	LoRaWAN	https://lora-developers.semtech.com/uploads/documents/files/LoRaWAN_Class_A_Devices_In_Depth_Downloadable.pdf	About Class A Devices
5	Base64 Converter	https://base64.guru/converter/decode/hex	Web tool for converting numbers from one format to another
6	Mike's GitHub Document Repository	https://github.com/mikedsp/helium/tree/master/MyDocuments 1. https://github.com/mikedsp/helium/blob/master/MyDocuments/RBS305-ATH-US_payloadAnalysis.pdf 2. https://github.com/mikedsp/helium/blob/master/MyDocuments/HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf 3. https://github.com/mikedsp/helium/blob/master/MyDocuments/20201016_TempHumidity-SensorCompare.xlsx 4. https://github.com/mikedsp/helium/blob/master/MyDocuments/20201020_RBS305data.xlsx	Document repository for various HowTo and Quick Start documents about working with IoT sensors in the Helium network 1. Manual decoding of some messages from the Radio Bridge sensor 2. Detailed instructions for getting data from a TBHH100 sensor logging into a Google Sheet using a Pipedream workflow 3. Spreadsheet with data collected from the RBS305-ATH-US sensor as well as from a TBHV110 sensor 4. Spreadsheet with data collected from the RBS305-ATH-US sensor

Author – Mike Boucher, mikeboucher@yahoo.com

			over 8 days

Add the Device to the Helium Console

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

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

Section 9 of the Radio Bridge document, *How to Connect LoRaWAN Sensors.pdf*, will help you find the DevEUI, AppEUI, and App Key information. If you read the Radio Bridge document, you'll see that Radio Bridge has its own Console that allows you to register your device and to integrate it with various public networks (Things Network, SENET, LORIOT, and more). Unfortunately, the Helium Network was not supported by the Radio Bridge Console at the time this document was written.

My device came with a small magnet that when placed next to a specific location on the device, triggers the device to send a packet. This is how I triggered the activation. Then I used the in-and-out of the freezer trick to initiate temperature changes to trigger additional uplink messages.

9. THIRD-PARTY NETWORK SERVER

This section provides the information required to connect the LoRaWAN sensors to a third party LoRaWAN network server not otherwise described in this document. The network server may reside on the gateway itself or in the cloud, and the server may push the data to the Radio Bridge console or another third-party application.

9.1. Sensor to Network Server

The LoRaWAN network server must use the connectivity parameters shown in the following table.

Table 7 LoRaWAN Parameters

LoRaWAN Parameter	Description
Activation Method	OTA (over the air activation). The sensor will send a join request and expect a join accept before any other messages can be sent.
Device EUI	This is the ID on the label located on the sensor itself. The barcode provided can also be used to read the Device EUI.
Application EUI	See the section on AppEUI/JoinEUI above. This can be customized in the factory for production orders, but most customers simply use this default.
Application Key	This is the Key on the label located on the sensor itself. The barcode provided can also be used to read the Application Key.

(above) Section 9 from the Radio Bridge manual, *How to Connect LoRaWAN Sensors.pdf*

Author – Mike Boucher, mikeboucher@yahoo.com

+ Import Devices + Add New Device

5 Devices — Quick Action

Device Name	Device EUI	Labels	Integrations	Frame Up	Frame Down	Packets Transferred	DC Used	Date Activated	Last Connected
TBHH100	58A0CB000011C1D4	TBHH100, CayenneInt_TBHH100	pipedreamInt_TBHH100, CayenneInt_TBHH100	313	1	236	234	Sep 4, 2020 5:03 AM	Sep 13, 2020 4:31 AM
TBHV110	58A0CB000011E251	CayenneInt_TBHV110, TBHV110	CayenneInt_TBHV110, pipedreamInt_TBHV110	2267	6	737	729	Sep 4, 2020 5:03 AM	Sep 13, 2020 9:17 AM
Mike's BOL	58A0CB0000202381	CayenneInt_Mike'sBOL, BOL	CayenneInt_Mike'sBOL, pipedreamInt_BOL	4033	1	3495	2451	Sep 4, 2020 4:59 AM	Sep 13, 2020 7:08 AM
Gary's BOL	58A0CB0000202488	CayenneInt_Gary'sBOL, BOL	CayenneInt_Gary'sBOL, pipedreamInt_BOL	4871	1	3874	2286	Sep 4, 2020 5:01 AM	Sep 13, 2020 6:09 AM
RBS305-ATH-US	CCC079000000E5644			3	1	33	29	Sep 12, 2020 12:37 PM	Sep 13, 2020 9:24 AM

(above) RBS305-ATH-US added to the Console and data is flowing

Displaying 2 / 10 Packets


```
{
  "channels": [
    {
      "debug": null,
      "description": "no channels configured",
      "id": "no_channel",
      "name": "no_channel",
      "status": "no_channel"
    }
  ],
  "devaddr": "CF010048",
  "device_name": "RBS305-ATH-US",
  "frame_down": 1,
  "frame_up": 3,
  "hotspots": [
    {
      "frequency": 904.0999755859375,
      "id": "112kk7sLkuPybrPDE4ZPAYcAXzuPZbV3F2MH5adatdGohmRX5zJW",
      "name": "clever-orange-mustang",
      "rssi": -41,
      "snr": 12.199999809265137,
      "spreading": "SF9BW125"
    }
  ],
  "payload": "GAEZWTEHQFmAAAI=",
  "payload_size": 11,
  "port": 2
}
```


(above) Data from the sensor in the Console debug window. Because there is no decoder function, the payload is not decoded.

Author – Mike Boucher, mikeboucher@yahoo.com

Apply a Decoder to the Device

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

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

Fortunately, Radio Bridge has provided decoder source code in the following GitHub repository:

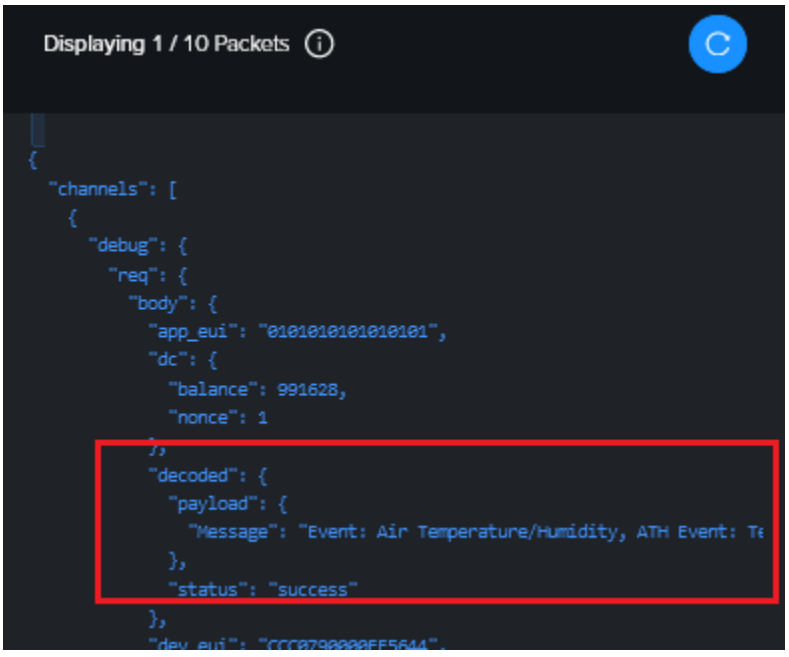
- https://github.com/RadioBridge/Packet-Decoder/blob/master/radio_bridge_packet_decoder.js

Here are the steps to go through:

1. Create a label called 'RBS305'
2. Create a new custom function
    a. Name it 'RBS305decoderFunction' to make it easy to identify in the Console
    b. Copy and paste in the function code from the Radio Bridge GitHub repository at https://github.com/RadioBridge/Packet-Decoder/blob/master/radio_bridge_packet_decoder.js

       Appendix B of this document show my complete custom decoder function source code.

    c. Apply the 'RBS305' label to the function
3. Go back to the Devices tab of the Console and apply the 'RBS305' label from the previous step to the RBS305-ATH-US device
4. While looking at the 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



(above) Decoded data in the Console debugger from the RBS305-ATH-US device.

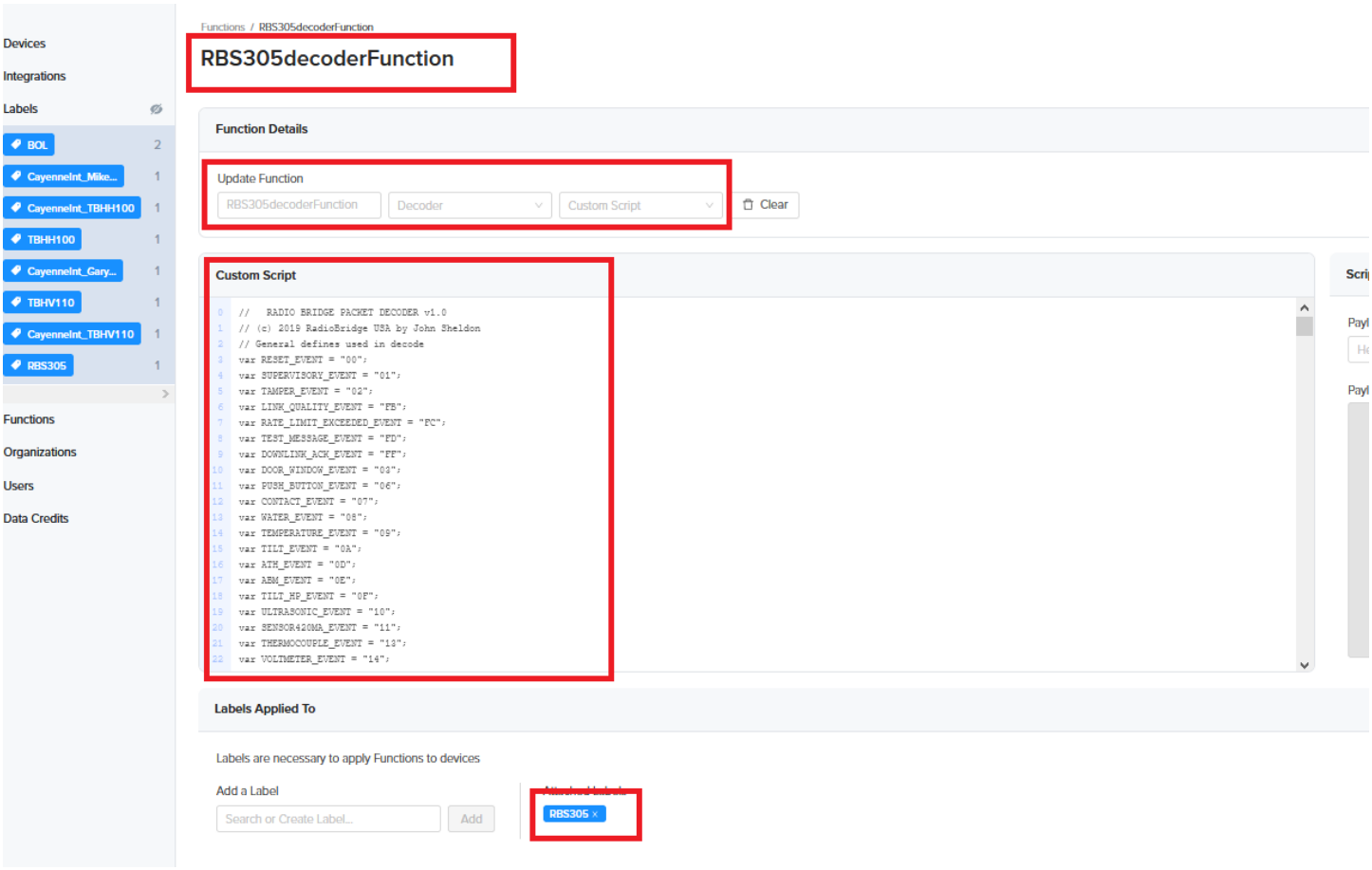
In the console debug screenshot above, part of the deoded message is cut off. Appendix A of this document has a copy of the entire JSON message. The complete decoded Message is shown below.

"decoded": {

Author – Mike Boucher, mikeboucher@yahoo.com

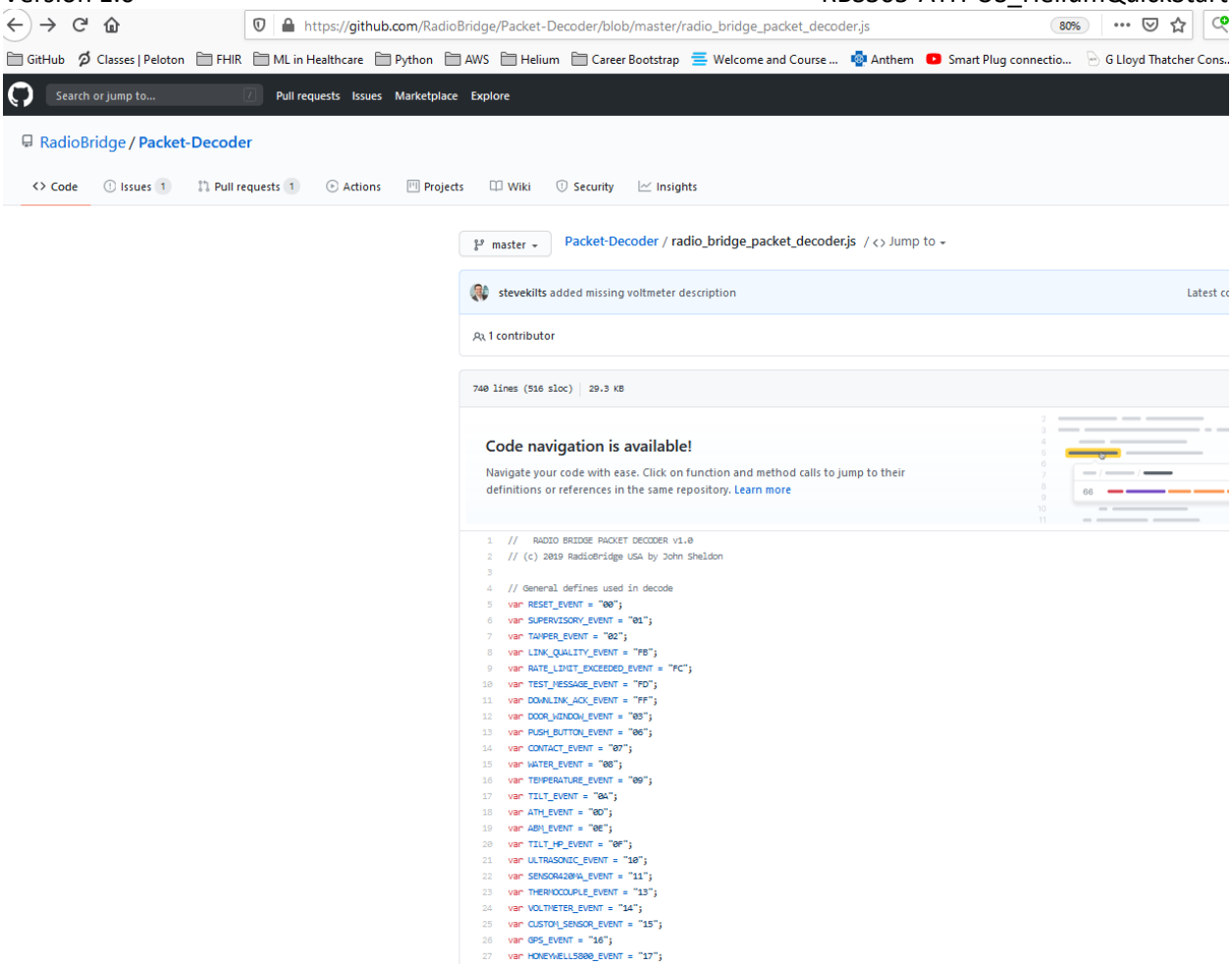
"payload": {
"Message": "Event: Air Temperature/Humidity, ATH Event: Temperature has Fallen Below Lower Threshold, Temperature: 9.9, Humidity: 15.6, Packet Counter: 9, Protocol Version: 1"
},
"status": "success"

The screenshots below illustrate some of the steps above.

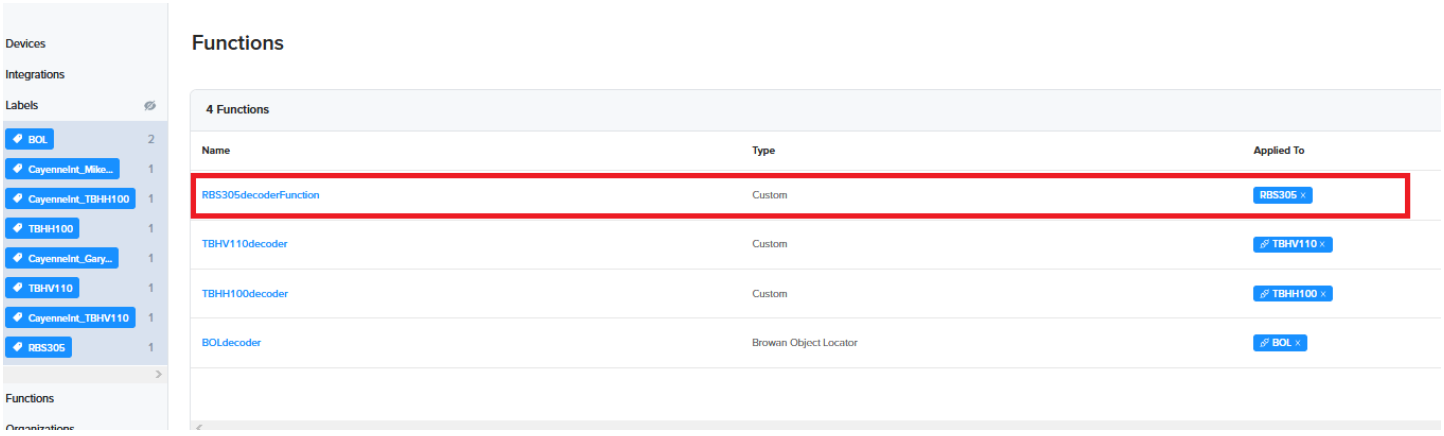


(above) Creating new decoder function called, RBS305decoderFunction, with label, *RBS305*.

Author – Mike Boucher, mikeboucher@yahoo.com



(above) Looking at the Generic decoder source code from the Radio Bridge GitHub repository



(above) The RBS305decoderFunction in the Function list of the Helium Console

Author – Mike Boucher, mikeboucher@yahoo.com

Devices

5 Devices

Device Name	Device EUI	Labels	Integrations	Frame Up	Frame Down	Packets Transferred	DC Used
TBHH100	58A0CB000011C1D4	TBHH100 × CayenneInt_TBHH100 ×	pipedreamInt_TBHH100, CayenneInt_TBHH100	319	1	237	235
TBHV110	58A0CB000011E251	CayenneInt_TBHV110 × TBHV110 ×	CayenneInt_TBHV110, pipedreamInt_TBHV110	2283	6	741	733
Mike's BOL	58A0CB0000202381	CayenneInt_Mike'sBOL × BOL ×	CayenneInt_Mike'sBOL, pipedreamInt_BOL	4033	1	3495	2451
Gary's BOL	58A0CB00002024B8	CayenneInt_Gary'sBOL × BOL ×	CayenneInt_Gary'sBOL, pipedreamInt_BOL	4871	1	3874	2286
RBS305-ATH-US	CCC0790000EE5644	RBS305 ×		4	1	34	30

(above) RBS305 label applied to the RBS305-ATH-US device

Author – Mike Boucher, mikeboucher@yahoo.com

Radio Bridge Decoder Debug

From 9/13 – 9/16/20, the Helium Console debug window was showing some received messages with a decoded function status of error. There was a discussion on this topic in the console Discord channel, with the following suggestion:
try{ your code goes here } catch (e) { decoded.err = e.message; }

I added this the try/catch block to my decoder function, but since I did that, I haven't seen any more decoder errors in the debug window, so I am not sure *why* I was getting the errors in the first place.

The screenshot below shows the try/catch block added to the decoder function.
The document, *RBS305-ATH-US_payloadAnalysis.doc*, shows some of the Json with the errors (and a success).

(above) A try/catch structure put around the contents of the Decoder function

Author – Mike Boucher, mikeboucher@yahoo.com

Radio Bridge Console post to Helium Discord Console Channel

Below is the message I posted to the Helium Discord server querying if anyone had been able to interface the Helium Console to the Radio Bridge Console. Unfortunately, I didn't see any responses.

Q: Has anyone been able to successfully connect the Helium network to the Radio Bridge console using a Helium HTTP integration to a Radio Bridge Console 3rd Party Integration (see section 9.2 in the attached document - How to Connect LoRaWAN Sensors.pdf) and/or the Radio Bridge Console Callback API?

For the past few weeks, I've been working with a Radio Bridge indoor temp/humidity sensor (RBS305-AUTH-US). After constructing a downlink message by hand, I was able to use the Helium Console downlink capability to configure the device to send measurements once an hour rather than the default threshold mode. I have not been able to get the device to send message on changes in temperature and so far Radio Bridge Support has not been much help. What would make it ***much*** *easier to control the device would be to use Radio Bridge's own console which handles message decoding and encoding. Unfortunately, while the Radio Bridge Console is set up for integrations to sigfox, machineQ, the Things Network, and more – it is* ***NOT*** *set up for integrations to the Helium Network. A support person from Radio Bridge indicated that is coming soon, but that doesn't help me now.*

Change Mode of Sensor from 'Threshold' to 'Report on Change'

The Browan temperature and humidity sensors are defaulted to the following triggers:

- 60-minute inactivity, ±2 °C delta, ±5 %RH Delta

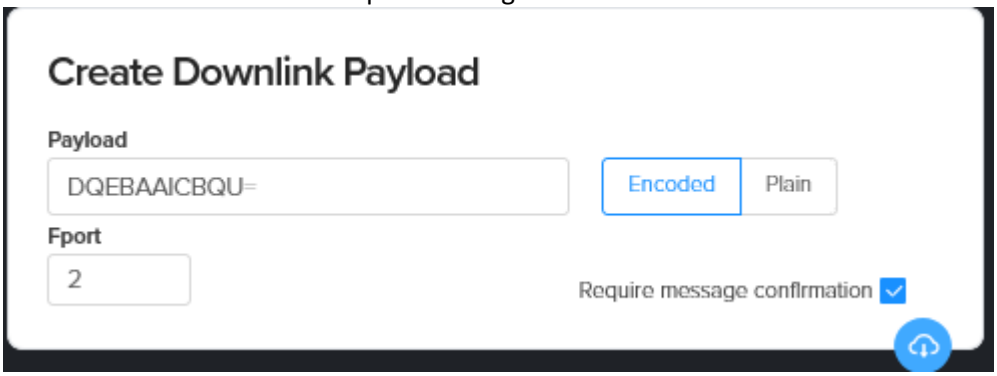
The RBS305-ATH-US sensor is defaulted to a threshold trigger mode rather than a report on change configuration mode. From section 6.2.1 in the Sensor User guide, the default thresholds for the radio bridge sensor are as follows;

- Lower temperature threshold. Default 10 degrees C.
- Upper temperature threshold. Default 40 degrees C.
- Lower humidity threshold. Default 40% relative humidity.
- Upper humidity threshold. Default 60% relative humidity.

So that we can compare the performance of the sensors, we're going to program the RBS305 to have the same trigger as the Browan devices. To accomplish this, we're going to do the 6 steps listed below. Details on the steps are provided **after** step 6.

1. Construct the downlink configuration message content
    a. Construct the message content in hex format
        i. 0D 01 01 00 02 02 05 05 = 60-minute inactivity, ±2 °C delta, ±5 %RH Delta
    b. Convert the message content to Base 64
        i. DQEBAAICBQU=

2. Prepare to send the Message
    a. Open up a new instance of the Helium Console and go to the Downlink form for the device
    b. Fill in the 'Create Downlink Payload' form in the Helium Console
        i. Payload = DQEBAAICBQU=
        ii. Set Fport = 2
        iii. Check the 'Require message confirmation' box



3. Reset the device with the magnet
    a. In a different Helium Console window, go to the device and enable the debugger
    b. Put the magnet by the appropriate place on the side of the device
    c. Watch the Helium debugger, waiting for a Supervisory event to appear
4. Send the downlink message
    a. When the supervisory event appears in the debugger window, trigger the downlink message in the Helium Console window from step 2
    b. If you don't see a response in the debug window within 10 seconds, trigger the downlink message again
5. Validate uplink response

Author – Mike Boucher, mikeboucher@yahoo.com

If the device accepted and understood the downlink message, you should receive a Downlink Ack message with a 'Message Valid' response in the Debug window. The message I received in the Console debug window is shown below.

```
"Message": "Event: Downlink Acknowledge, Downlink: Message Valid, Packet
Counter: 4, Protocol Version: 1"
```

6. Initiate a temperature change to validate the device is operating as expected

   To test whether the device is sending messages upon a change in temperature or humidity, put the device into (or take it out of) your freezer. The message I received in the Console debug window is shown below.

```
"Message": "Event: Air Temperature/Humidity, ATH Event: Temperature Report-
on-Change Decrease, Temperature: 14.5, Humidity: 11.2, Packet Counter: 14,
Protocol Version: 1"
```

The sub sections below provide useful details on the previous 6 steps.

Understanding when the RBS305-ATH-US device is listening for Downlink Messages

The paragraph below is from the LoRaWAN reference. It says that a Class A device listens for a downlink message one or two seconds after the uplink.

End devices in a LoRaWAN network come in three classes: Class A, Class B and Class C. While end devices can always send uplinks at will, the device's class determines when it can receive downlinks. The class also determines a device's energy efficiency. The more energy efficient a device, the longer the battery life. An In-depth Look at LoRaWAN® Class A Devices semtech.com/LoRa Page 3 of 12 Technical Paper Proprietary November 2019 Semtech All end devices must support Class A ("Aloha") communications. Class A end devices spend most of their time in sleep mode. Because LoRaWAN is not a "slotted" protocol, end devices can communicate with the network server any time there is a change in a sensor reading or when a timer fires. Basically, they can wake up and talk to the server at any moment. After the device sends an uplink, it "listens" for a message from the network one and two seconds after the uplink (receive windows) before going back to sleep. Class A is the most energy efficient and results in the longest battery life.

Constructing the Downlink Message

To construct the downlink message, we're going to use the information in this Radio Bridge Support Article with additional context from sections 6.2.2 and 6.2.3 in *Wireless Air Temp and Humidity Sensor User Guide.pdf.*

Report on Change Configuration Example:

0D 01 8F 00 0A 0F 05 14

0D	Downlink Message Type (Air Temp & Humidity = 0x0D)
01	Reporting Mode (Report On Change = 0x01)
8F	Periodic Reporting (15 minutes = 0x8F)
00	Not Used
0A	Temperature Increase (10 degrees C = 0x0A)
0F	Temperature Decrease (15 degrees C = 0x0F)
05	Humidity Increase (5% = 0x05)
14	Humidity Decrease (20% = 0x14)

If the configuration is successful the sensor will respond with a ***Downlink Acknowledge*** message:

(above) Screenshot from [this Radio Bridge Support Article]

0D 01 01 00 02 02 05 05 = 60-minute inactivity, ±2 °C delta, ±5 %RH Delta

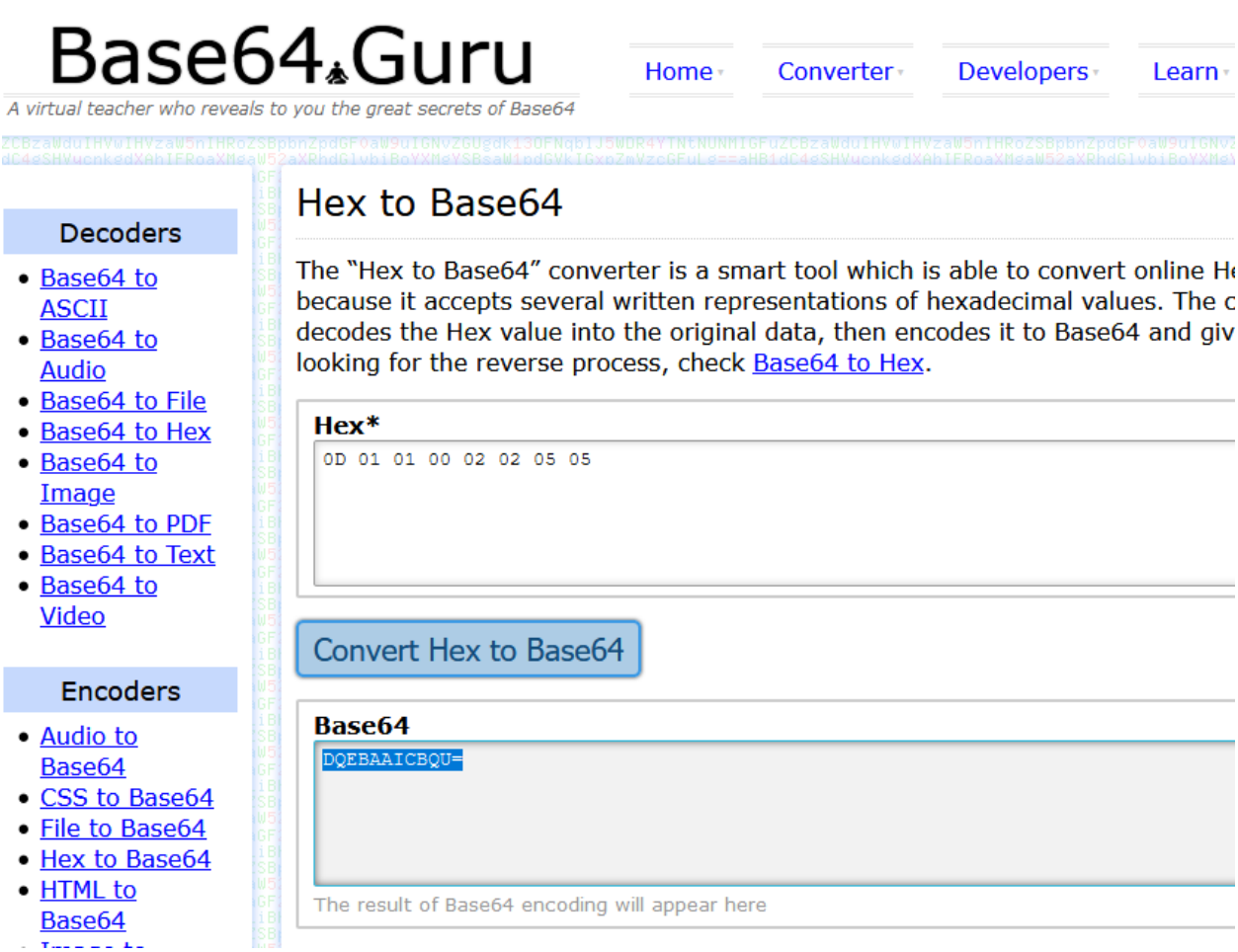
Byte Value	Description
0D	Downlink Message Type (Air Temp & Humidity = 0x0D)
01	Reporting Mode (Report On Change = 0x01)
01	Periodic Reporting (1 hour = 0x01)
00	Not used
02	Temperature Increase (2 degrees C = 0x02)
02	Temperature Decrease (2 degrees C = 0x02)
05	Humidity Increase (5% = 0x05)
05	Humidity Decrease (5% = 0x05)

Converting the Downlink Message from Hex to Base64 in Preparation for Putting the Message into the Helium Downlink form

ALERT - Be careful when using the Base64 Converter (Reference 5).

This is where I got into trouble the first time I reprogrammed the device. I did an incorrect conversion of 0D 01 01 00 02 02 05 05 into Base64. Specifically, the first time I added an extra 0 at the end of my hex string when I placed the string into the converter to make the converter work. The sensor device accepted the Base64 message I generated this way, but was only sending uplink messages every hour; not on temperature or humidity changes. I tried the conversion again several days later and this time was able to do it without adding the extra 0 at the end of the hex string, and that gave me a different Base64 payload to download.

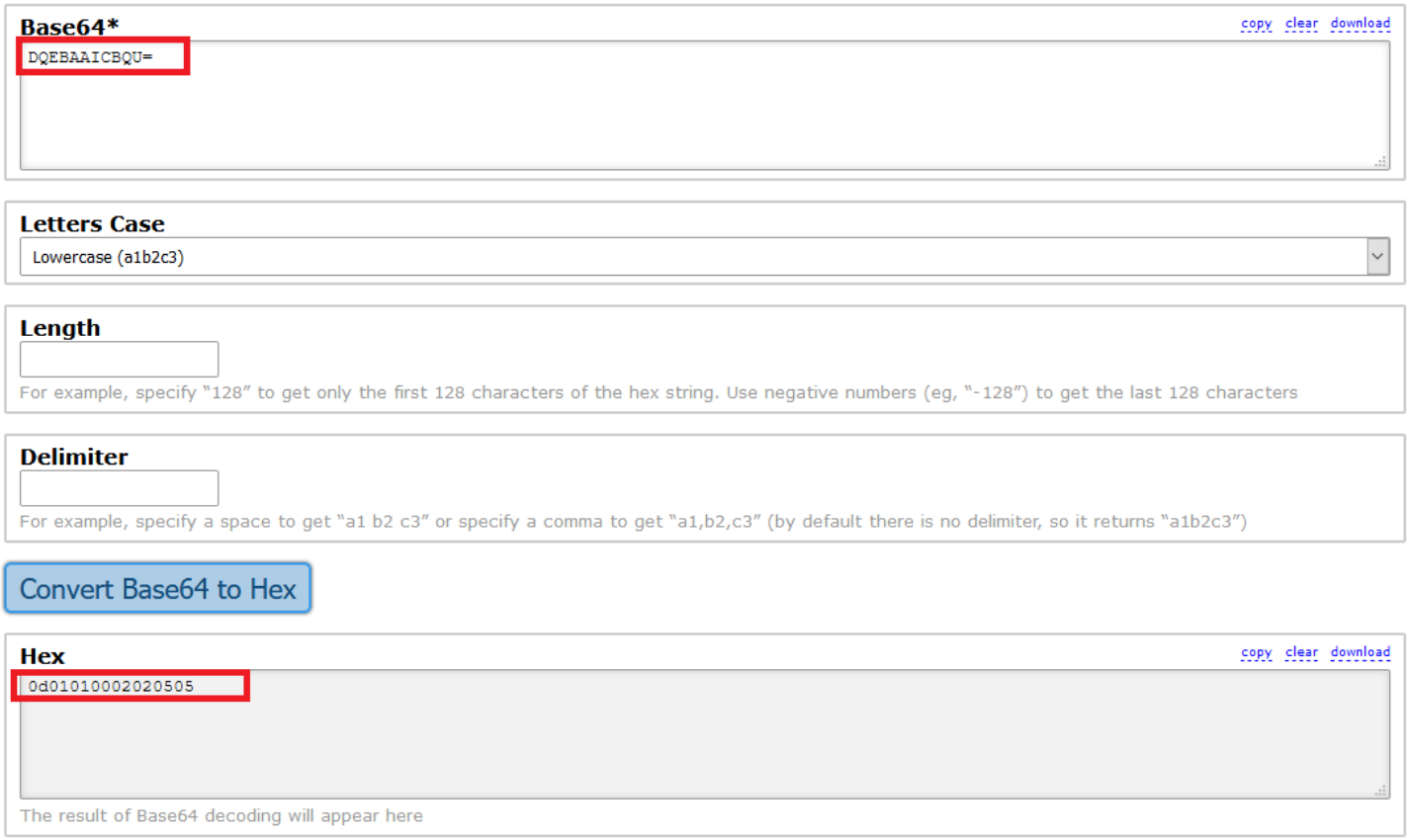
Take the hex string from above, 0D 01 01 00 02 02 05 05, and paste it into Hex to Base 64 to convert to Base64 format. The Base64 formatted string is what will be put into the Helium Console downlink form.



(above) Converting the hex string, 0D 01 01 00 02 02 05 05, into a Base64 string, DQEBAAICBQU=

Confirm the result by converting the Base64 back into Hex using Base64 to Hex

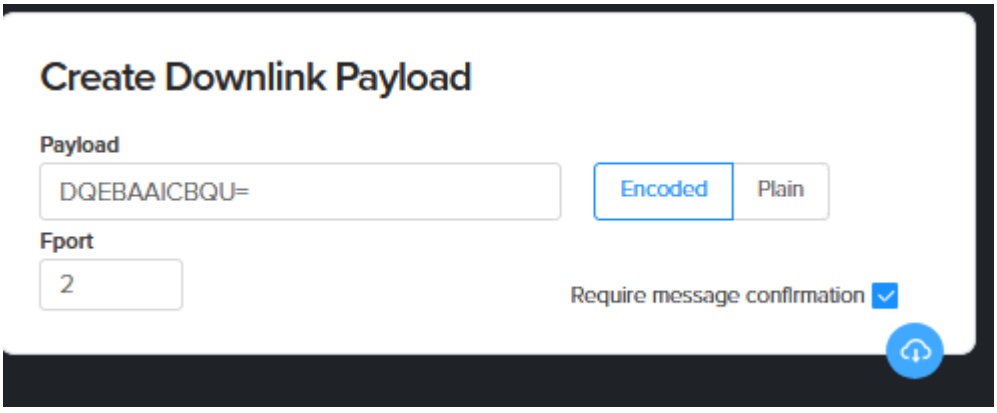
Author – Mike Boucher, mikeboucher@yahoo.com



(above) Converting Base64 string back into Hex format to double check the Base64 value is legit

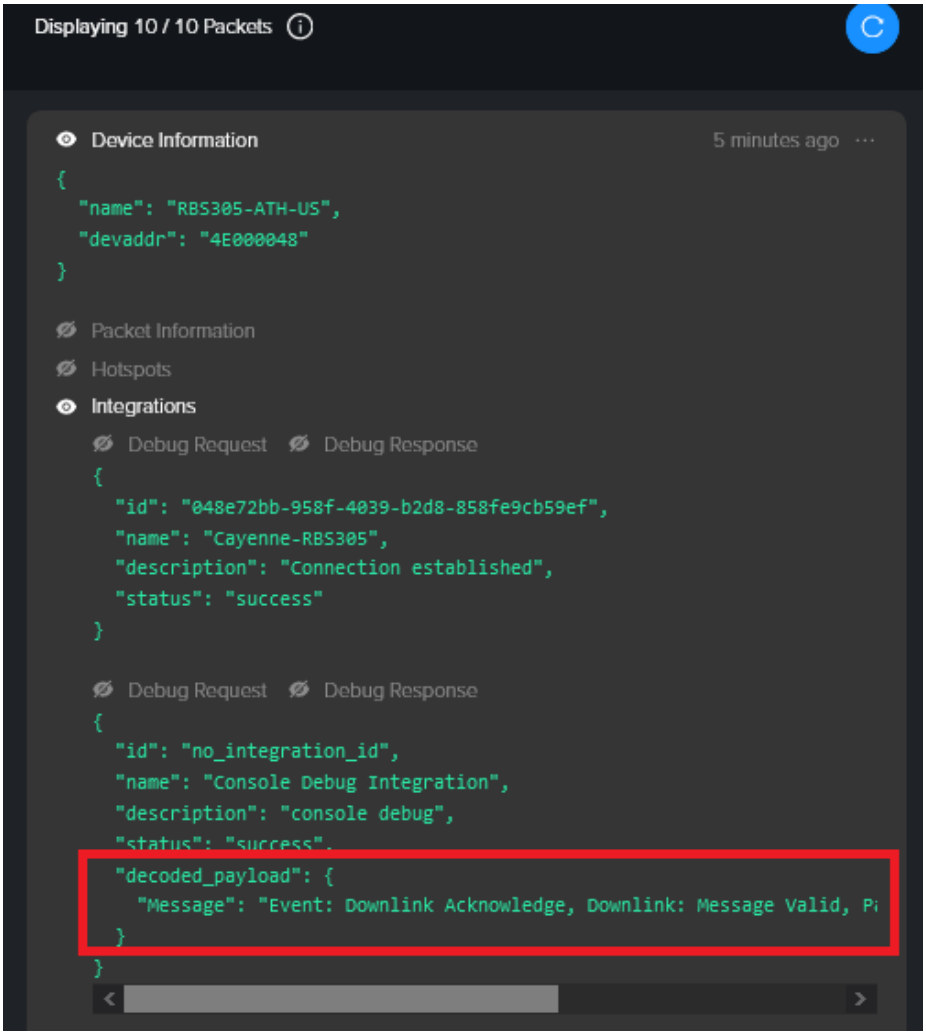
Preparing to Send the Downlink Message

Using the Helium instructions here: https://developer.helium.com/console/integrations/http#downlink-tool-example



Author – Mike Boucher, mikeboucher@yahoo.com

Helium Debug Window – Download Ack Message



(above) Downlink Ack message with a 'message valid' confirmation of the downlink.

Debug Response

```
{
  "id": "048e72bb-958f-4039-b2d8-858fe9cb59ef",
  "name": "Cayenne-RBS305",
  "description": "Connection established",
  "status": "success"
}
```

Debug RequestDebug Response

```
{
  "id": "no_integration_id",
  "name": "Console Debug Integration",
  "description": "console debug",
  "status": "success",
  "decoded_payload": {
    "Message": "Event: Downlink Acknowledge, Downlink: Message Valid,
Packet Counter: 4, Protocol Version: 1"
  }
}
```

(above) Complete text of the Debug Response from the Helium Debug Window

Author – Mike Boucher, mikeboucher@yahoo.com

Build a Cayenne myDevices Integration

Using the instructions at the following URL, build a Cayenne myDevices 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 label called, *RBS305-cayenne-int*
2. In the Helium Console, apply the integration label, '*RBS305-cayenne-int*' to the RBS305-ATH-US device

RBS305-AUTH-US to Cayenne Console Settings		
Console Object	**Name**	**Applied Label(s)**
Integration	Cayenne-RBS305	RBS305-cayenne-int
Device	RBS305-ATH-US	RBS305-cayenne-int

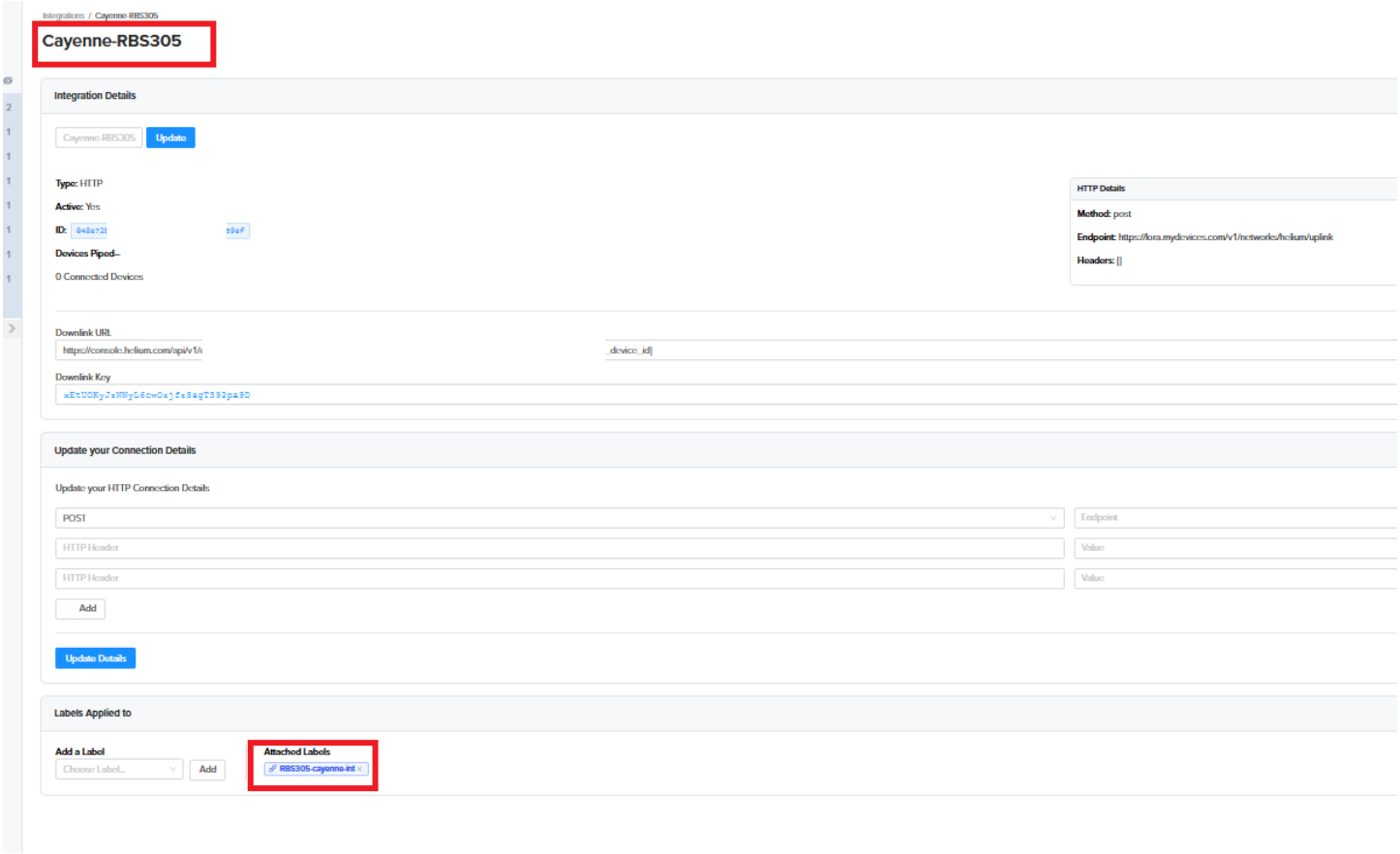
(above) Helium Console settings for the RBS305-to-Cayenne Integration

3. In the Helium Console, create a new Cayenne Integration
    a. Name it 'Cayenne-RBS305'
    b. Apply the label called 'RBS305-cayenne-int'

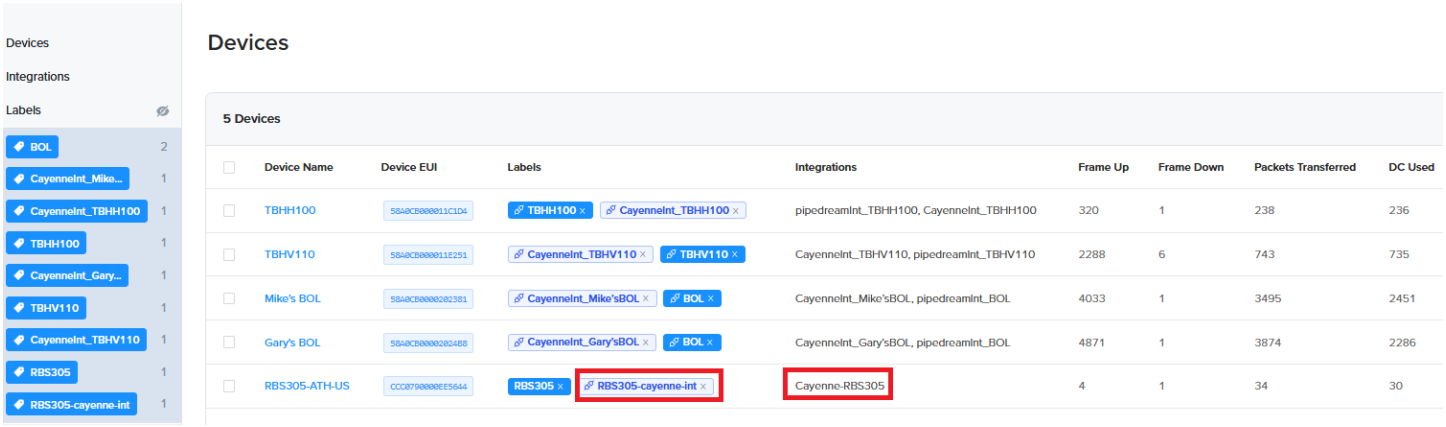
4. In the Cayenne Dashboard, add/create a new Device/Widget
    a. Add new… > Devices & Widgets > Lora > Helium > Radio Bridge Air Temperature and Humidity Sensor
    b. Change the default name to 'RBS305-ATH-US'
    c. Add the DevEUI from the Helium Console
5. Wait for the data to flow into Cayenne

The screenshots below illustrate the steps above.

Author – Mike Boucher, mikeboucher@yahoo.com

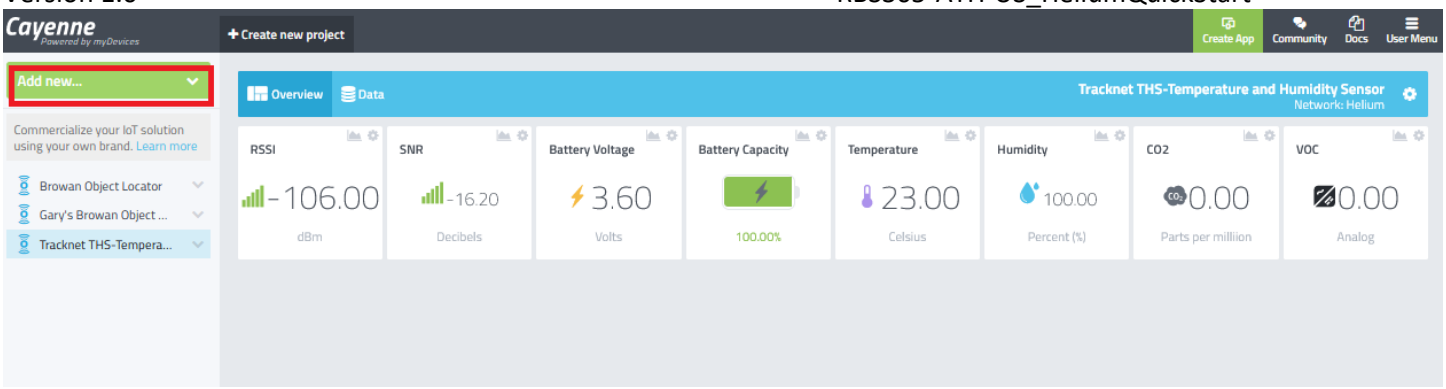


(above) the Cayenne-RBS305 integration has been created!

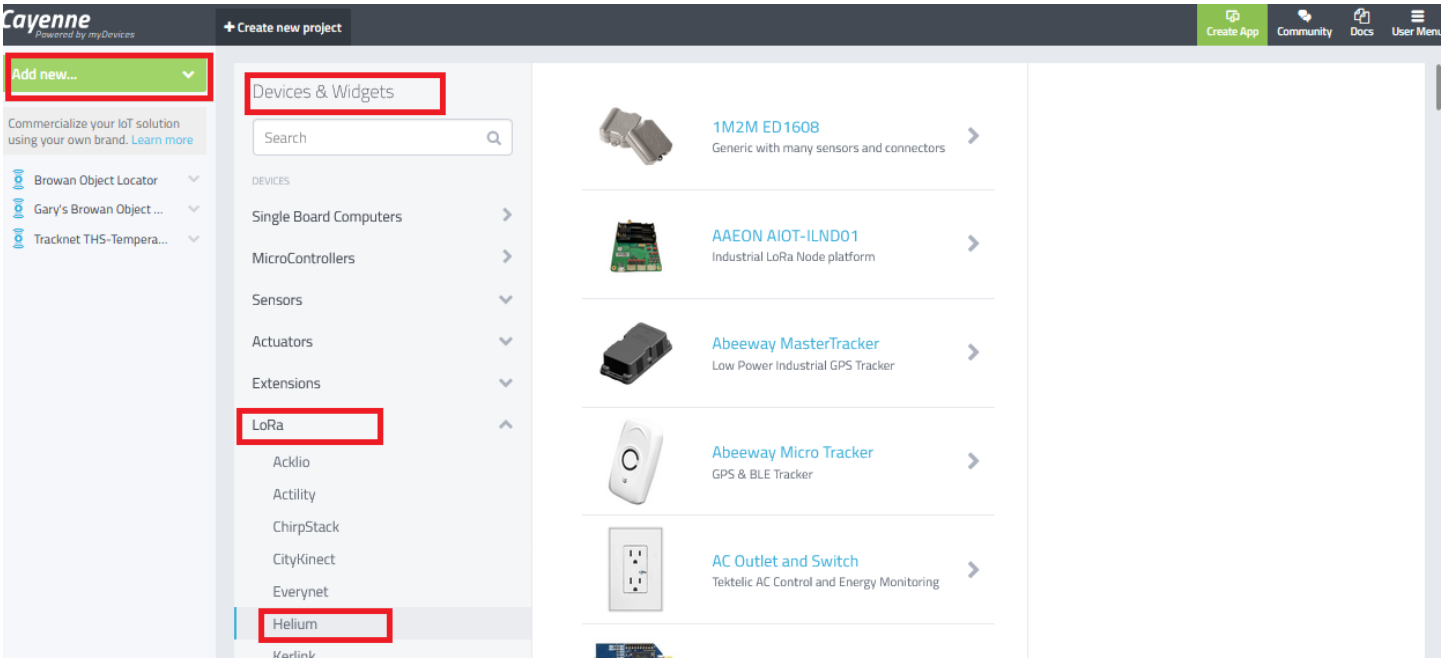


(above) RBS305-ATH-US device with the Cayenne integration applied through the RBS305_cayenne_int label

Author – Mike Boucher, mikeboucher@yahoo.com

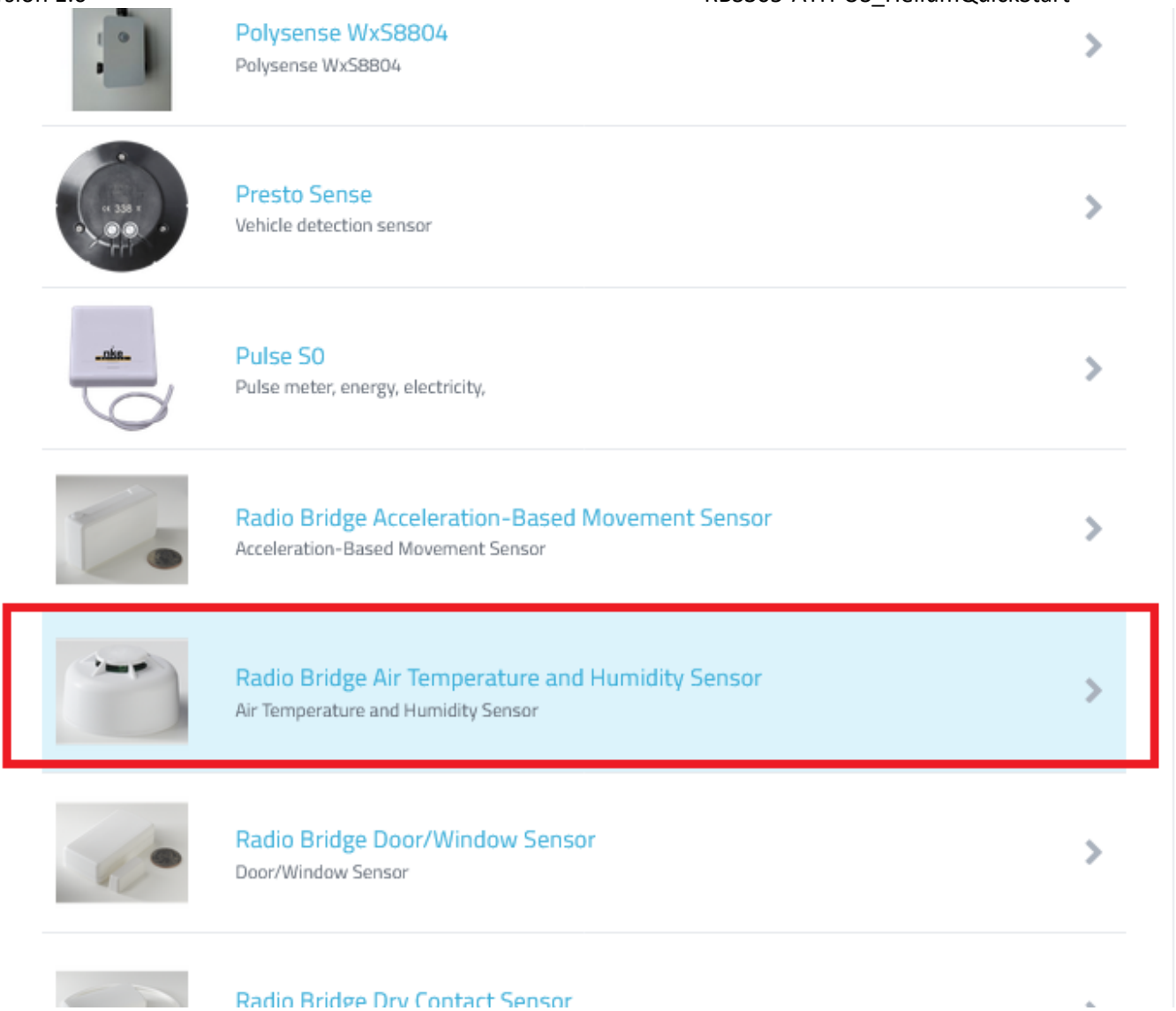


(above) Log into your Cayenne Dashboard and select the 'Add new…' button



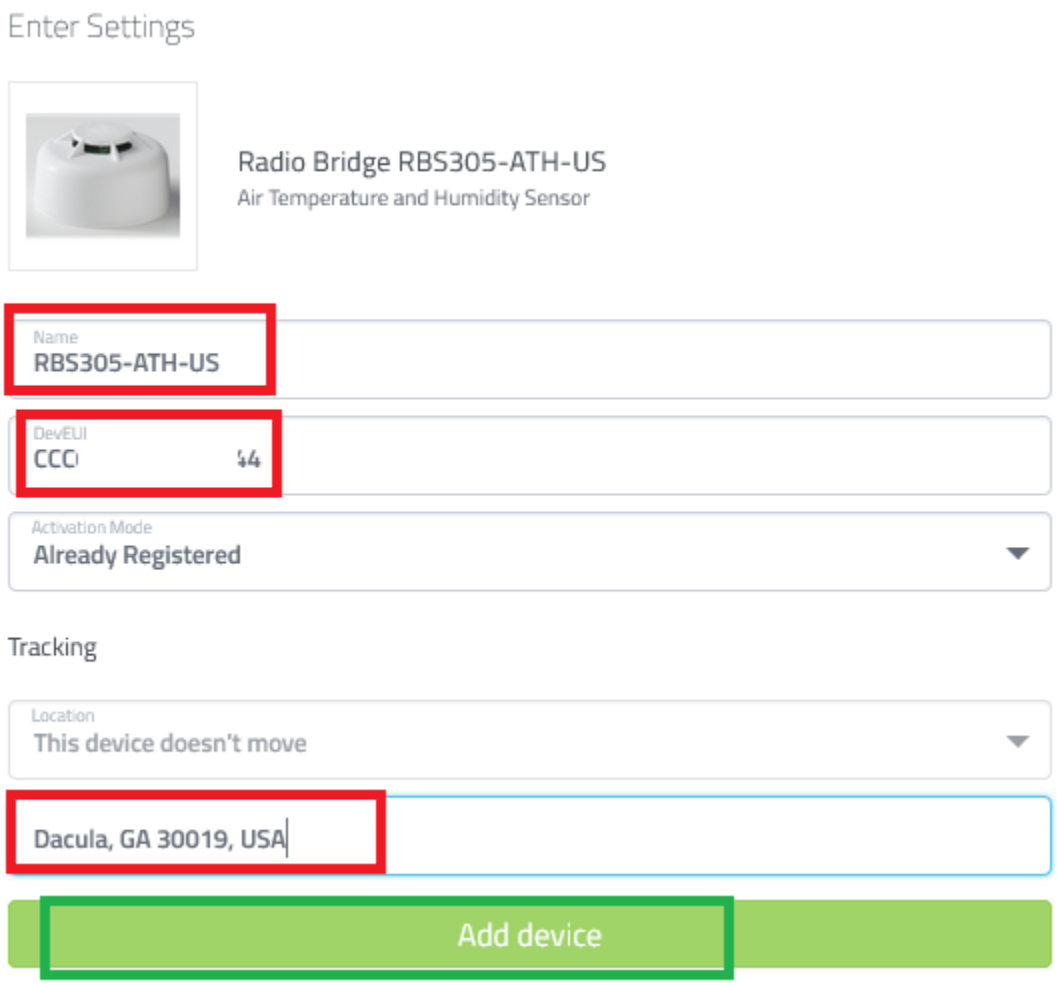
(above) Add new… > Devices & Widgets > Lora > Helium

Author – Mike Boucher, mikeboucher@yahoo.com

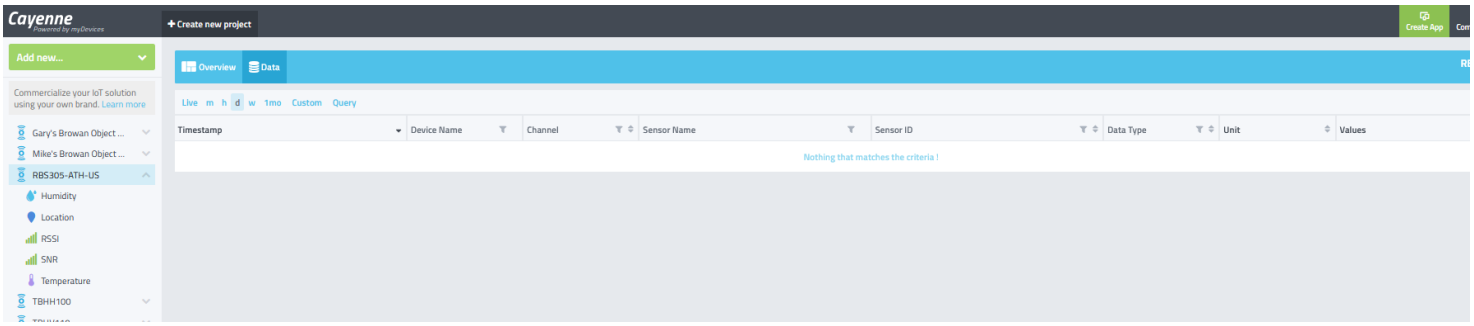


(above) Select the 'Radio Bridge Air Temperature and Humidity Sensor'

Author – Mike Boucher, mikeboucher@yahoo.com

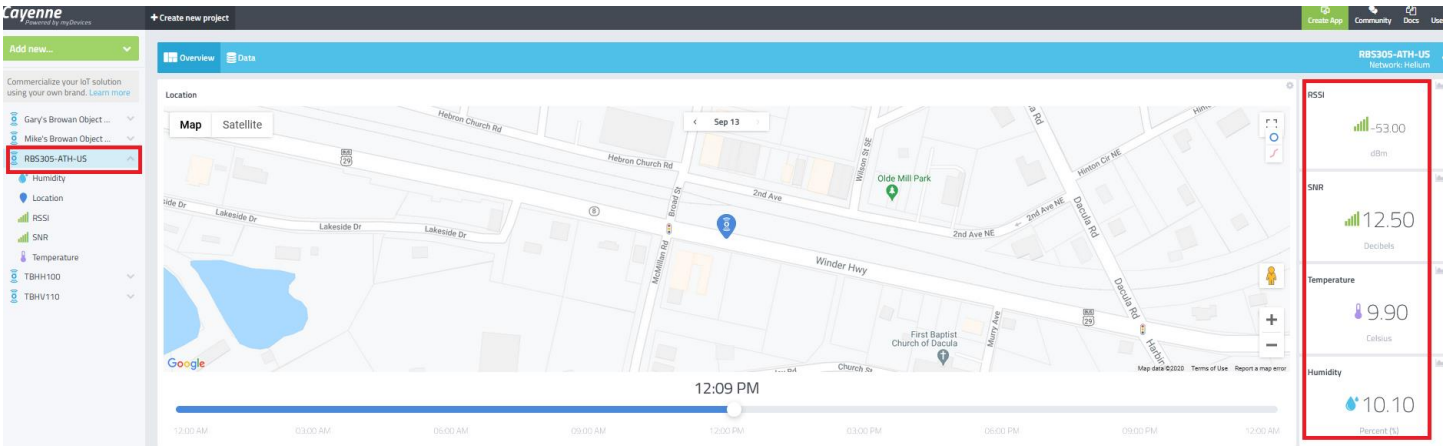


(above) Entering the settings for the device

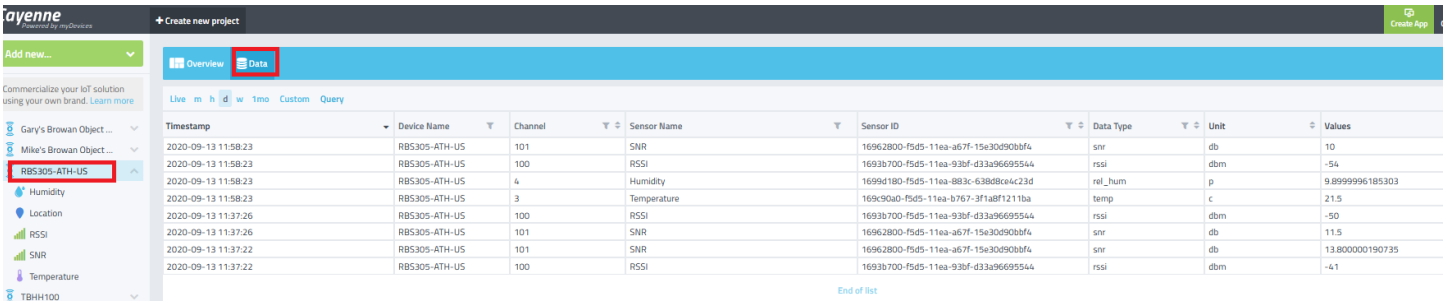


(above) The RBS305-ATH-US device has been added to your Cayenne dashboard and is waiting for data to come over

Author – Mike Boucher, mikeboucher@yahoo.com



(above) Cayenne Dashboard showing Overview of data from the RBS305-ATH-US device – this is the most recent set of received measurements. Note – the device does not send location information, by the Cayenne Device Setup insisted I enter a location, so I put in 'Dacula, GA 30019 USA' and that is what is showing up in the map above.



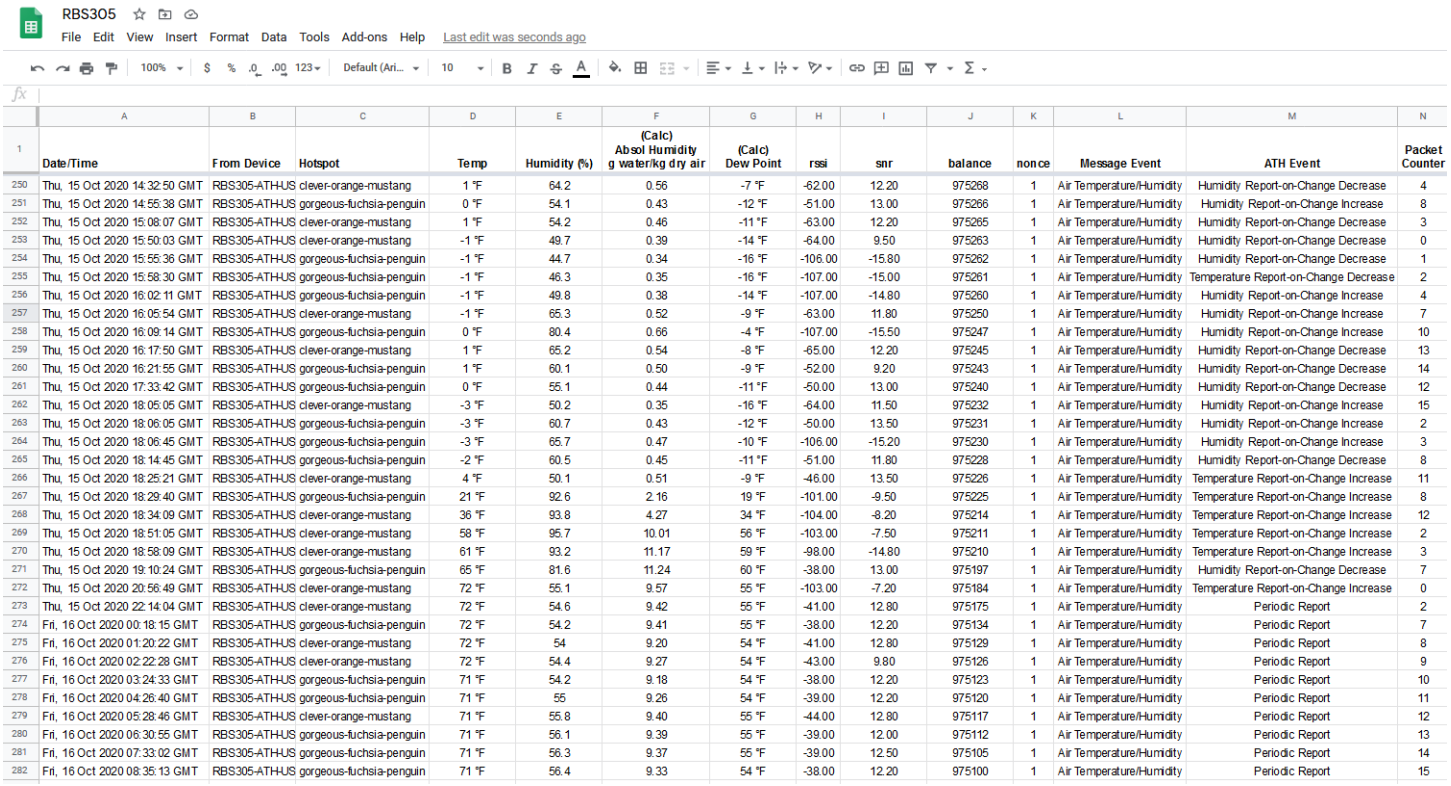
(above) Cayenne Dashboard showing measurements received from the RBS305-AUTH-US device. The device sends different types of events; some are status events and don't contain temperature or humidity information. You can see that in the data sent at 11:37 in the screenshot above.

Author – Mike Boucher, mikeboucher@yahoo.com

Build a Google Sheet integration

Using the process described in Reference 6.2, *HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf* , build the following data flow to produce a spreadsheet like the one shown below.

- RBS305-ATH-US > Helium Hotspot > Helium Console/Network > Pipedream > Google Sheet



RBS305

File Edit View Insert Format Data Tools Add-ons Help Last edit was seconds ago

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date/Time	From Device	Hotspot	Temp	Humidity (%)	(Calc) Absol Humidity g water/kg dry air	(Calc) Dew Point	rssi	snr	balance	nonce	Message Event	ATH Event	Packet Counter
250	Thu, 15 Oct 2020 14:32:50 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	64.2	0.56	-7 °F	-62.00	12.20	975268	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	4
251	Thu, 15 Oct 2020 14:55:38 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	54.1	0.43	-12 °F	-51.00	13.00	975266	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	8
252	Thu, 15 Oct 2020 15:08:07 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	54.2	0.46	-11 °F	-63.00	12.20	975265	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	3
253	Thu, 15 Oct 2020 15:50:03 GMT	RBS305-ATH-US	clever-orange-mustang	-1 °F	49.7	0.39	-14 °F	-64.00	9.50	975263	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	0
254	Thu, 15 Oct 2020 15:55:36 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	44.7	0.34	-16 °F	-106.00	-15.80	975262	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	1
255	Thu, 15 Oct 2020 15:58:30 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	46.3	0.35	-16 °F	-107.00	-15.00	975261	1	Air Temperature/Humidity	Temperature Report-on-Change Decrease	2
256	Thu, 15 Oct 2020 16:02:11 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	49.8	0.38	-14 °F	-107.00	-14.80	975260	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	4
257	Thu, 15 Oct 2020 16:05:54 GMT	RBS305-ATH-US	clever-orange-mustang	-1 °F	65.3	0.52	-9 °F	-63.00	11.80	975250	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	7
258	Thu, 15 Oct 2020 16:09:14 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	80.4	0.66	-4 °F	-107.00	-15.50	975247	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	10
259	Thu, 15 Oct 2020 16:17:50 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	65.2	0.54	-8 °F	-65.00	12.20	975245	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	13
260	Thu, 15 Oct 2020 16:21:55 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	1 °F	60.1	0.50	-9 °F	-52.00	9.20	975243	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	14
261	Thu, 15 Oct 2020 17:33:42 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	55.1	0.44	-11 °F	-50.00	13.00	975240	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	12
262	Thu, 15 Oct 2020 18:05:05 GMT	RBS305-ATH-US	clever-orange-mustang	-3 °F	50.2	0.35	-16 °F	-64.00	11.50	975232	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	15
263	Thu, 15 Oct 2020 18:06:05 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-3 °F	60.7	0.43	-12 °F	-50.00	13.50	975231	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	2
264	Thu, 15 Oct 2020 18:06:45 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-3 °F	65.7	0.47	-10 °F	-106.00	-15.20	975230	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	3
265	Thu, 15 Oct 2020 18:14:45 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-2 °F	60.5	0.45	-11 °F	-51.00	11.80	975228	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	8
266	Thu, 15 Oct 2020 18:25:21 GMT	RBS305-ATH-US	clever-orange-mustang	4 °F	50.1	0.51	-9 °F	-46.00	13.50	975226	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	11
267	Thu, 15 Oct 2020 18:29:40 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	21 °F	92.6	2.16	19 °F	-101.00	-9.50	975225	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	8
268	Thu, 15 Oct 2020 18:34:09 GMT	RBS305-ATH-US	clever-orange-mustang	36 °F	93.8	4.27	34 °F	-104.00	-8.20	975214	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	12
269	Thu, 15 Oct 2020 18:51:05 GMT	RBS305-ATH-US	clever-orange-mustang	58 °F	95.7	10.01	56 °F	-103.00	-7.50	975211	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	2
270	Thu, 15 Oct 2020 18:58:09 GMT	RBS305-ATH-US	clever-orange-mustang	61 °F	93.2	11.17	59 °F	-98.00	-14.80	975210	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	3
271	Thu, 15 Oct 2020 19:10:24 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	65 °F	81.6	11.24	60 °F	-38.00	13.00	975197	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	7
272	Thu, 15 Oct 2020 20:56:49 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	55.1	9.57	55 °F	-103.00	-7.20	975184	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	0
273	Thu, 15 Oct 2020 22:14:04 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54.6	9.42	55 °F	-41.00	12.80	975175	1	Air Temperature/Humidity	Periodic Report	2
274	Fri, 16 Oct 2020 00:18:15 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	72 °F	54.2	9.41	55 °F	-38.00	12.20	975134	1	Air Temperature/Humidity	Periodic Report	7
275	Fri, 16 Oct 2020 01:20:22 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54	9.20	54 °F	-41.00	12.80	975129	1	Air Temperature/Humidity	Periodic Report	8
276	Fri, 16 Oct 2020 02:22:28 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54.4	9.27	54 °F	-43.00	9.80	975126	1	Air Temperature/Humidity	Periodic Report	9
277	Fri, 16 Oct 2020 03:24:33 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	54.2	9.18	54 °F	-38.00	12.20	975123	1	Air Temperature/Humidity	Periodic Report	10
278	Fri, 16 Oct 2020 04:26:40 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	55	9.26	54 °F	-39.00	12.20	975120	1	Air Temperature/Humidity	Periodic Report	11
279	Fri, 16 Oct 2020 05:28:46 GMT	RBS305-ATH-US	clever-orange-mustang	71 °F	55.8	9.40	55 °F	-44.00	12.80	975117	1	Air Temperature/Humidity	Periodic Report	12
280	Fri, 16 Oct 2020 06:30:55 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	56.1	9.39	55 °F	-39.00	12.00	975112	1	Air Temperature/Humidity	Periodic Report	13
281	Fri, 16 Oct 2020 07:33:02 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	56.3	9.37	55 °F	-39.00	12.50	975105	1	Air Temperature/Humidity	Periodic Report	14
282	Fri, 16 Oct 2020 08:35:13 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	56.4	9.33	54 °F	-38.00	12.20	975100	1	Air Temperature/Humidity	Periodic Report	15

(above) Google Sheet receiving data from the RBS305-ATH-US sensor

Very Important: When creating the Custom HTTP Integration in the Helium Console 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 device on the Helium Console, Pipedream will NOT get decoded data. I made this mistake when building Pipedream integrations for my other sensors, the TBHH100 and the TBHV110.

The HowTo instructions for the TBHH100 device can be found in GitHub here:
https://github.com/mikedsp/helium/blob/master/MyDocuments/HowTo_BrowanTBHH100_to_GoogleSheet-SHARE.pdf

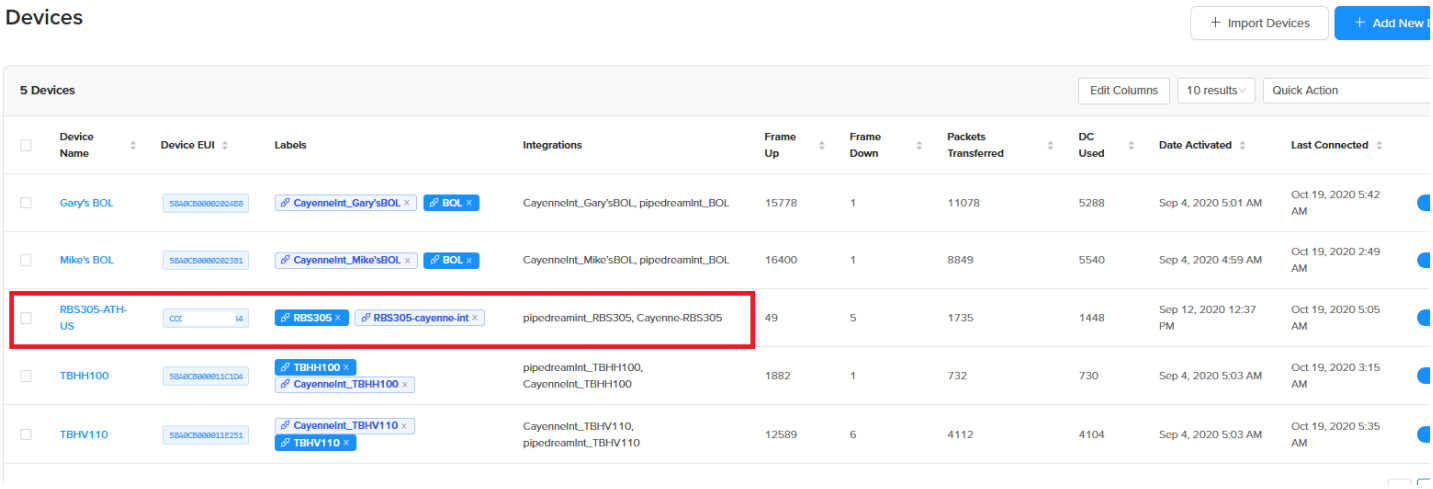
Two Excel files with some collected data can be found in GitHub at the URL below. The excel file was created by saving a copy of the Google Sheet that is capturing the sensor data.
https://github.com/mikedsp/helium/blob/master/MyDocuments/20201016_TempHumidity-SensorCompare.xlsx
https://github.com/mikedsp/helium/blob/master/MyDocuments/20201020_RBS305data.xlsx

A public version of the Pipedream workflow can be found here:
https://pipedream.com/@dangermikeb/pipedreamint_rbs305-p_OKCdkK

High level steps are as follows

1. Get the RBS305-ATH-US set up in the Helium Console
    a. Create then apply a decoder function to the device so you can read/decode the data from the device
        i. Use a label called RBS305 to apply the decoder to the device
2. In Pipedream, create a pipedream endpoint (via a new Pipedream Workflow)
    a. Go to the Workflow tab in Pipedream
    b. Create a new workflow and set “HTTP/Webhook” as the trigger
        i. Name = RBS305-ATH-US to Google Sheet
3. Create an HTTP Integration in the Helium Console
    a. Integration Name = pipedreamint_RBS305
    b. Use the URL from the pipedream endpoint in the previous step
    c. Use the RBS305 label to connect the integration to the RBS305-ATH-US device and its decoder function
4. Verify that Data is flowing from the RBS305-ATH-US > Helium Console/Network > Pipedream workflow
5. Create a Google Sheet
    a. Make 1 column for each sensor payload data element you see (or are interested in) when looking at the device in the Helium Console with debug turned on
6. Complete the development of the Pipedream workflow
    a. Make the connection to your Google Sheet
    b. Drop the data received from the trigger step into a step that connects to your Google Sheet
7. (optional) Do some data manipulation of the sensor data in the Pipedream workflow to make it read better in the Google sheet, e.g. the sensor is sending the timestamp in Unix
8. (optional) Enjoy the sensor data flowing into your Google Sheet. Share the Sheet with a friend.

The next few pages contain screenshots of some of the steps above.



(above) RBS305-ATH-US Sensor set up in the Helium Console with integrations for Cayenne and Pipedream

Author – Mike Boucher, mikeboucher@yahoo.com

Functions

4 Functions

Name	Type	Applied To
RBS305decoderFunction	Custom	RBS305 ×
TBHV110decoder	Custom	TBHV110 ×
TBHH100decoder	Custom	TBHH100 ×
BOLdecoder	Browan Object Locator	BOL ×

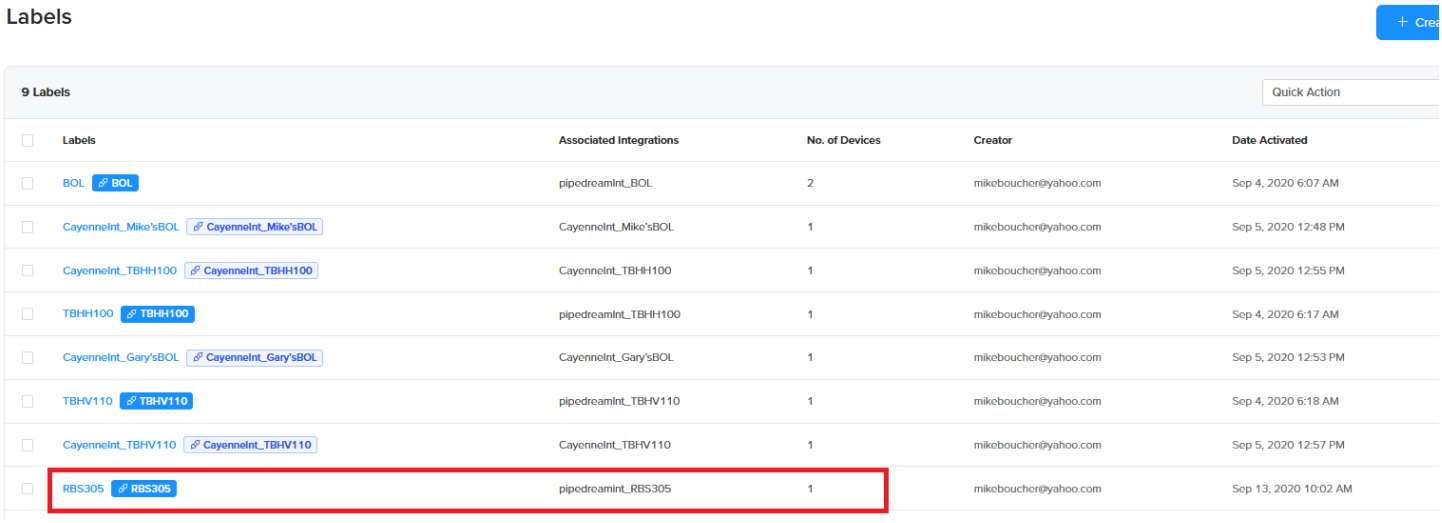
(above) Functions tab of Helium Console – showing the custom decoder created for the RBS305-ATH-US device, attached to the device with the RBS305 label

My Integrations

Name	Type	Labels	Devices
CayenneInt_Mike'sBOL	HTTP	CayenneInt_Mike'sBOL	1
CayenneInt_Gary'sBOL	HTTP	CayenneInt_Gary'sBOL	1
CayenneInt_TBHH100	HTTP	CayenneInt_TBHH100	1
CayenneInt_TBHV110	HTTP	CayenneInt_TBHV110	1
pipedreamInt_TBHV110	HTTP	TBHV110	1
pipedreamInt_BOL	HTTP	BOL	2
Cayenne-RBS305	HTTP	RBS305-cayenne-int	1
pipedreamInt_TBHH100	HTTP	TBHH100	1
pipedreamint_RBS305	HTTP	RBS305	1

(above) Integrations tab of Helium Console – showing the custom HTTP integration created for the RBS305-ATH-US device, attached with the RBS305 label

Author – Mike Boucher, mikeboucher@yahoo.com

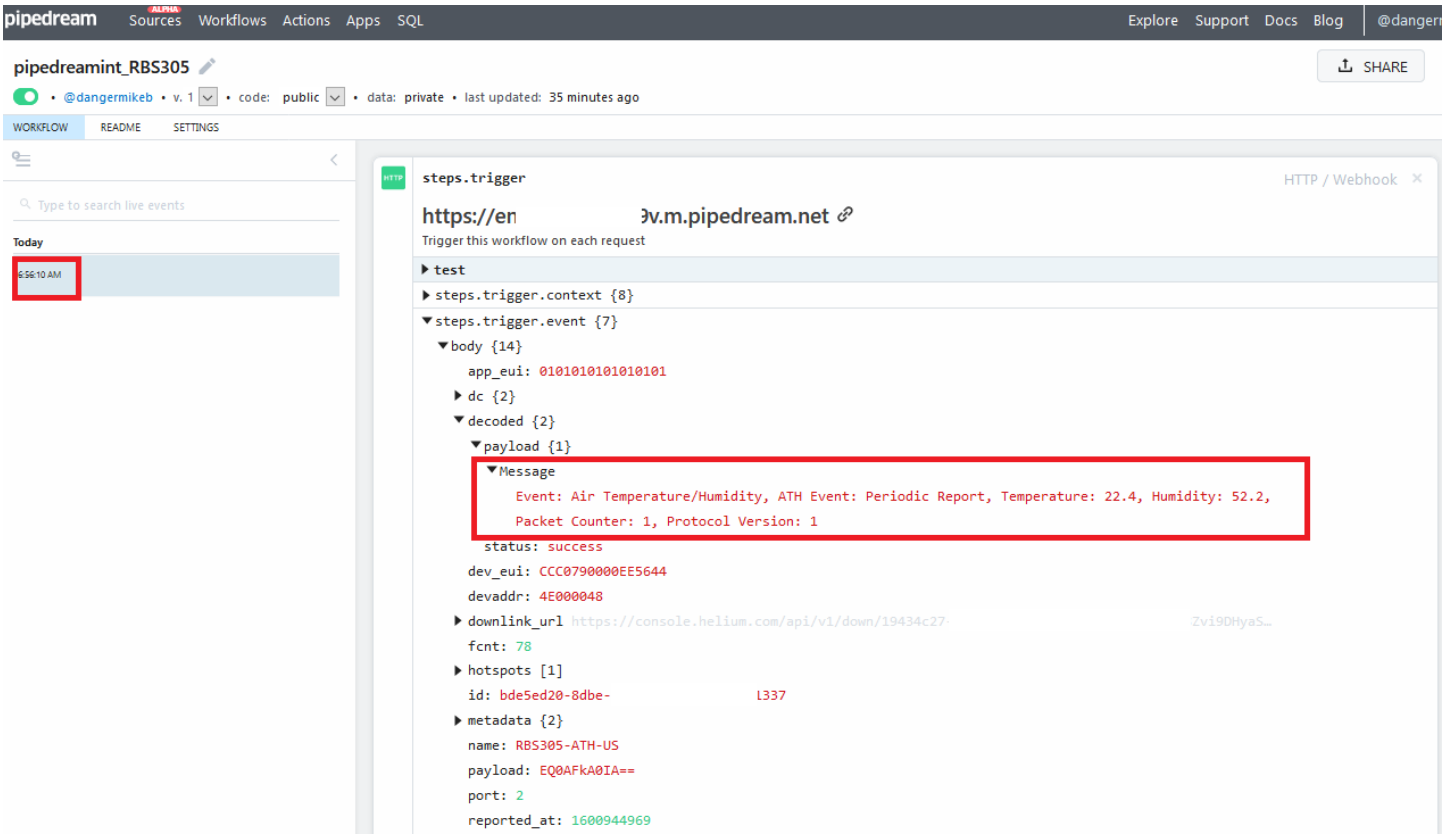


9 Labels

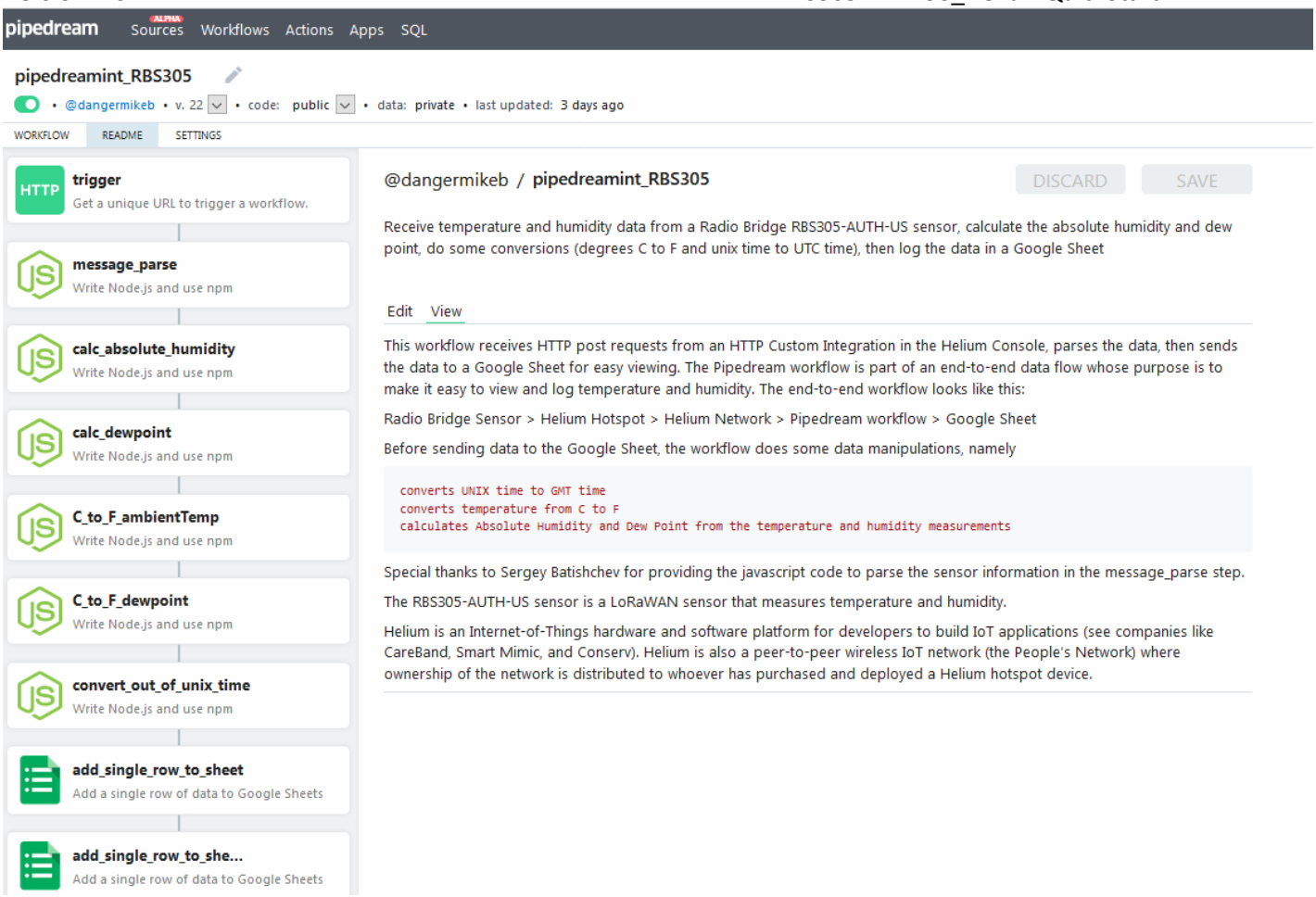
Quick Action

Labels	Associated Integrations	No. of Devices	Creator	Date Activated
BOL BOL	pipedreamInt_BOL	2	mikeboucher@yahoo.com	Sep 4, 2020 6:07 AM
CayenneInt_Mike'sBOL CayenneInt_Mike'sBOL	CayenneInt_Mike'sBOL	1	mikeboucher@yahoo.com	Sep 5, 2020 12:48 PM
CayenneInt_TBHH100 CayenneInt_TBHH100	CayenneInt_TBHH100	1	mikeboucher@yahoo.com	Sep 5, 2020 12:55 PM
TBHH100 TBHH100	pipedreamInt_TBHH100	1	mikeboucher@yahoo.com	Sep 4, 2020 6:17 AM
CayenneInt_Gary'sBOL CayenneInt_Gary'sBOL	CayenneInt_Gary'sBOL	1	mikeboucher@yahoo.com	Sep 5, 2020 12:53 PM
TBHV110 TBHV110	pipedreamInt_TBHV110	1	mikeboucher@yahoo.com	Sep 4, 2020 6:18 AM
CayenneInt_TBHV110 CayenneInt_TBHV110	CayenneInt_TBHV110	1	mikeboucher@yahoo.com	Sep 5, 2020 12:57 PM
RBS305 RBS305	pipedreamint_RBS305	1	mikeboucher@yahoo.com	Sep 13, 2020 10:02 AM

(above) Labels tab of Helium Console – showing the label, RBS305, associated to the pipedreamint_RBS305 integration



(above) Pipedream workflow after High Level Step 4 above – i.e. decoded data is flowing into the Pipedream workflow. Message ATH Event type = Periodic Report. Notice that the payload is 1 long message String. It's not easy to extract the individual data elements from this format, so we'll use a subsequent step to convert the string into an array.



(above) Pipedream workflow after high level step 7 – showing all the steps in the workflow. A brief explanation of each step is provided below:

- trigger – receives the message from the Helium network
- message_parse – converts the sensor payload information from 1 long string into an array for easy access to each informational element. Special thanks to Sergey Batishchev for providing the JavaScript code.
- Calc_absolute_humidity – calculates the absolute humidity from sensor temperature and relative humidity values
- Calc_dewpoint – calculates the dew point from the sensor's temperature and relative humidity values
- C_to_F_ambientTemp – converts the sensor's temperature value from Celsius to Fahrenheit
- C_to_F_dewpoint – converts the dew point calculation from Celsius to Fahrenheit
- Convert_out_of_unix_time – converts the sensor time stamp value from a Unix value to a GMT value
- Add_single_row_to_sheet – pushes data to my Radio Bridge Google Sheet
- Add_single_row_to_sheet-SensorCompare – pushes data to a different Google Sheet called TempHumidity-SensorCompare, which is also receiving data from TBHV110 sensor

Author – Mike Boucher, mikeboucher@yahoo.com

steps.add_single_row_to_sheet

Add a single row of data to Google Sheets

auth

Google Sheets (auths.google_sheets): @gmail.com

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.$return_value}}
[1]: {{event.body.name}}
[2]: {{event.body.hotspots[0].name}}
[3]: {{steps.C_to_F_ambientTemp.$return_value}}
[4]: {{steps.message_parse.$return_value.Humidity}}
[5]: {{steps.calc_absolute_humidity.$return_value}}
[6]: {{steps.C_to_F_dewpoint.$return_value}}
[7]: {{event.body.hotspots[0].rssi}}
[8]: {{event.body.hotspots[0].snr}}
[9]: {{event.body.dc.balance}}
[10]: {{event.body.dc.nonce}}
[11]: {{steps.message_parse.$return_value.Event}}
[12]: {{steps.message_parse.$return_value["ATH Event"]}}
[13]: {{steps.message_parse.$return_value["Packet Counter"]}}
```

(above) Pipedream workflow step used to drop the data into a row in the Google Sheet. The step, message_parse, was where the data received in the trigger step was converted from a single String into an array.

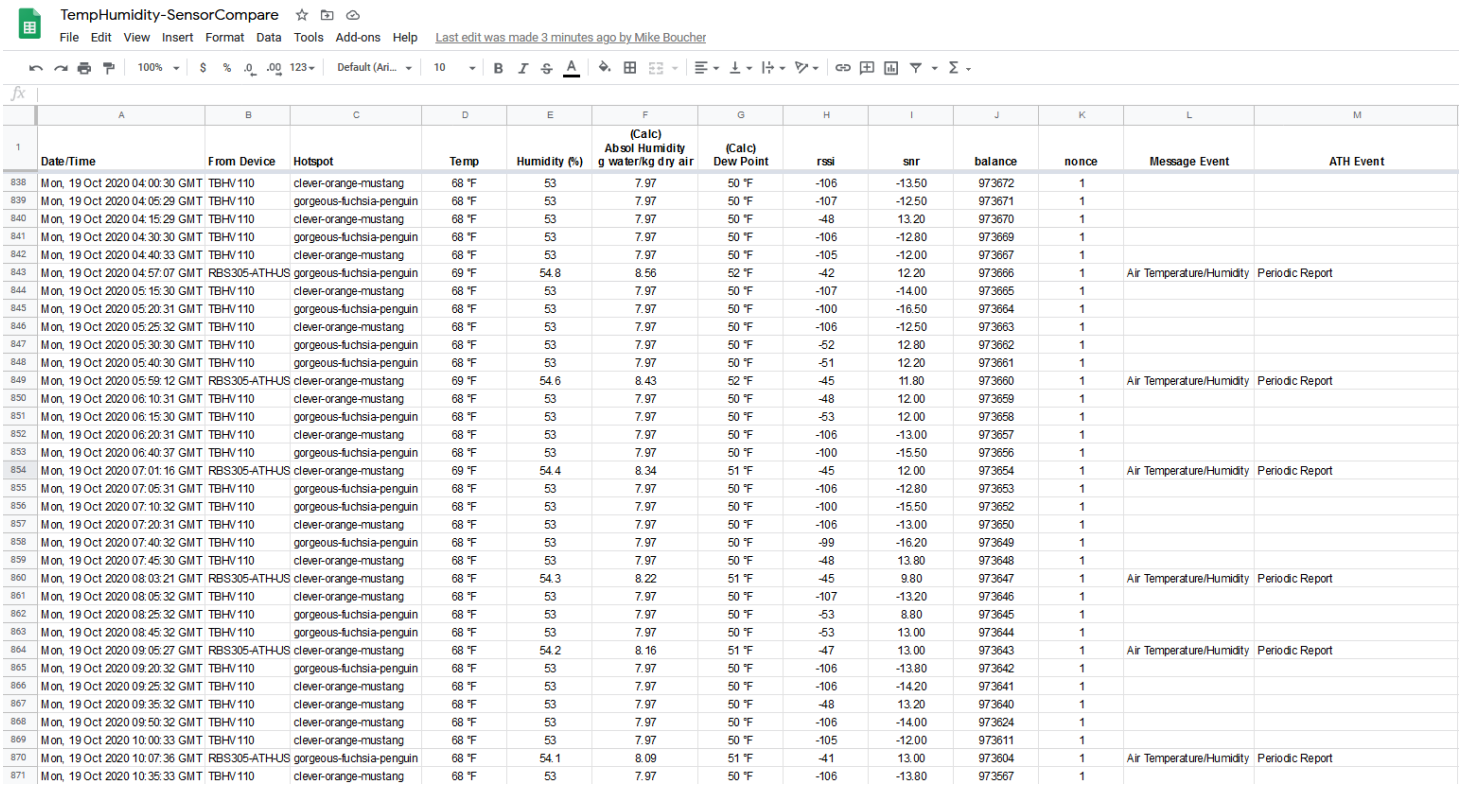
RBS305

File Edit View Insert Format Data Tools Add-ons Help Last edit was made 12 minutes ago by Mike Boucher

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Date/Time	From Device	Hotspot	Temp	Humidity (%)	(Calc) Absol Humidity g water/kg dry air	(Calc) Dew Point	rssi	snr	balance	nonce	Message Event	ATH Event	Packet Counter
250	Thu, 15 Oct 2020 14:32:50 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	64.2	0.561600242	-7 °F	-62	12.19999981	975268	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	4
251	Thu, 15 Oct 2020 14:55:38 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	54.1	0.4347547481	-12 °F	-51	13	975266	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	8
252	Thu, 15 Oct 2020 15:08:07 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	54.2	0.4608905937	-11 °F	-63	12.19999981	975265	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	3
253	Thu, 15 Oct 2020 15:50:03 GMT	RBS305-ATH-US	clever-orange-mustang	-1 °F	49.7	0.3918953367	-14 °F	-64	9.5	975263	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	0
254	Thu, 15 Oct 2020 15:55:36 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	44.7	0.3393430107	-16 °F	-106	-15.80000019	975262	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	1
255	Thu, 15 Oct 2020 15:58:30 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	46.3	0.3481761638	-16 °F	-107	-15	975261	1	Air Temperature/Humidity	Temperature Report-on-Change Decrease	2
256	Thu, 15 Oct 2020 16:02:11 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-1 °F	49.8	0.3780835334	-14 °F	-107	-14.80000019	975260	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	4
257	Thu, 15 Oct 2020 16:05:54 GMT	RBS305-ATH-US	clever-orange-mustang	-1 °F	65.3	0.5150065913	-9 °F	-63	11.80000019	975250	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	7
258	Thu, 15 Oct 2020 16:09:14 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	80.4	0.6586427138	-4 °F	-107	-15.5	975247	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	10
259	Thu, 15 Oct 2020 16:17:50 GMT	RBS305-ATH-US	clever-orange-mustang	1 °F	65.2	0.5441759338	-8 °F	-65	12.19999981	975245	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	13
260	Thu, 15 Oct 2020 16:21:55 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	1 °F	60.1	0.501575699	-9 °F	-52	9.199999809	975243	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	14
261	Thu, 15 Oct 2020 17:33:42 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	0 °F	55.1	0.4386329542	-11 °F	-50	13	975240	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	12
262	Thu, 15 Oct 2020 18:05:05 GMT	RBS305-ATH-US	clever-orange-mustang	-3 °F	50.2	0.3498241119	-16 °F	-64	11.5	975232	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	15
263	Thu, 15 Oct 2020 18:06:05 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-3 °F	60.7	0.4271045633	-12 °F	-50	13.5	975231	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	2
264	Thu, 15 Oct 2020 18:06:45 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-3 °F	65.7	0.4667462619	-10 °F	-106	-15.19999981	975230	1	Air Temperature/Humidity	Humidity Report-on-Change Increase	3
265	Thu, 15 Oct 2020 18:14:45 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	-2 °F	60.5	0.4507359892	-11 °F	-51	11.80000019	975228	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	8
266	Thu, 15 Oct 2020 18:25:21 GMT	RBS305-ATH-US	clever-orange-mustang	4 °F	50.1	0.5133624664	-9 °F	-46	13.5	975226	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	11
267	Thu, 15 Oct 2020 18:29:40 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	21 °F	92.6	2.16308726	19 °F	-101	-9.5	975225	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	8
268	Thu, 15 Oct 2020 18:34:09 GMT	RBS305-ATH-US	clever-orange-mustang	36 °F	93.8	4.274803699	34 °F	-104	-8.199999809	975214	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	12
269	Thu, 15 Oct 2020 18:51:05 GMT	RBS305-ATH-US	clever-orange-mustang	58 °F	95.7	10.00772115	56 °F	-103	-7.5	975211	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	2
270	Thu, 15 Oct 2020 18:58:09 GMT	RBS305-ATH-US	clever-orange-mustang	61 °F	93.2	11.17289491	59 °F	-98	-14.80000019	975210	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	3
271	Thu, 15 Oct 2020 19:10:24 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	65 °F	81.6	11.24067165	60 °F	-38	13	975197	1	Air Temperature/Humidity	Humidity Report-on-Change Decrease	7
272	Thu, 15 Oct 2020 20:56:49 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	55.1	9.570096135	55 °F	-103	-7.199999809	975184	1	Air Temperature/Humidity	Temperature Report-on-Change Increase	0
273	Thu, 15 Oct 2020 22:14:04 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54.6	9.423553838	55 °F	-41	12.80000019	975175	1	Air Temperature/Humidity	Periodic Report	2
274	Fri, 16 Oct 2020 00:18:15 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	72 °F	54.2	9.411413479	55 °F	-38	12.19999981	975134	1	Air Temperature/Humidity	Periodic Report	7
275	Fri, 16 Oct 2020 01:20:22 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54	9.203968052	54 °F	-41	12.80000019	975129	1	Air Temperature/Humidity	Periodic Report	8
276	Fri, 16 Oct 2020 02:22:28 GMT	RBS305-ATH-US	clever-orange-mustang	72 °F	54.4	9.273162059	54 °F	-43	9.800000191	975126	1	Air Temperature/Humidity	Periodic Report	9
277	Fri, 16 Oct 2020 03:24:33 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	54.2	9.181579442	54 °F	-38	12.19999981	975123	1	Air Temperature/Humidity	Periodic Report	10
278	Fri, 16 Oct 2020 04:26:40 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	71 °F	55	9.261600209	54 °F	-39	12.19999981	975120	1	Air Temperature/Humidity	Periodic Report	11
279	Fri, 16 Oct 2020 05:28:46 GMT	RBS305-ATH-US	clever-orange-mustang	71 °F	55.8	9.398349976	55 °F	-44	12.80000019	975117	1	Air Temperature/Humidity	Periodic Report	12

(above) Screenshot of the RBS305 Google Sheet receiving data from the Pipedream workflow. In rows 250-261, the sensor was in a freezer. The rows highlighted in red show how the sensor behaved when the sensor was removed from the freezer and placed in the living room.

Author – Mike Boucher, mikeboucher@yahoo.com



TempHumidity-SensorCompare

File Edit View Insert Format Data Tools Add-ons Help Last edit was made 3 minutes ago by Mike Boucher

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Date/Time	From Device	Hotspot	Temp	Humidity (%)	(Calc) Absol Humidity g water/kg dry air	(Calc) Dew Point	rssi	snr	balance	nonce	Message Event	ATH Event
838	Mon, 19 Oct 2020 04:00:30 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-13.50	973672	1		
839	Mon, 19 Oct 2020 04:05:29 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-107	-12.50	973671	1		
840	Mon, 19 Oct 2020 04:15:29 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-48	13.20	973670	1		
841	Mon, 19 Oct 2020 04:30:30 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-106	-12.80	973669	1		
842	Mon, 19 Oct 2020 04:40:33 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-105	-12.00	973667	1		
843	Mon, 19 Oct 2020 04:57:07 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	69 °F	54.8	8.56	52 °F	-42	12.20	973666	1	Air Temperature/Humidity	Periodic Report
844	Mon, 19 Oct 2020 05:15:30 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-107	-14.00	973665	1		
845	Mon, 19 Oct 2020 05:20:31 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-100	-16.50	973664	1		
846	Mon, 19 Oct 2020 05:25:32 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-12.50	973663	1		
847	Mon, 19 Oct 2020 05:30:30 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-52	12.80	973662	1		
848	Mon, 19 Oct 2020 05:40:30 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-51	12.20	973661	1		
849	Mon, 19 Oct 2020 05:59:12 GMT	RBS305-ATH-US	clever-orange-mustang	69 °F	54.6	8.43	52 °F	-45	11.80	973660	1	Air Temperature/Humidity	Periodic Report
850	Mon, 19 Oct 2020 06:10:31 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-48	12.00	973659	1		
851	Mon, 19 Oct 2020 06:15:30 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-53	12.00	973658	1		
852	Mon, 19 Oct 2020 06:20:31 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-13.00	973657	1		
853	Mon, 19 Oct 2020 06:40:37 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-100	-15.50	973656	1		
854	Mon, 19 Oct 2020 07:01:16 GMT	RBS305-ATH-US	clever-orange-mustang	69 °F	54.4	8.34	51 °F	-45	12.00	973654	1	Air Temperature/Humidity	Periodic Report
855	Mon, 19 Oct 2020 07:05:31 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-106	-12.80	973653	1		
856	Mon, 19 Oct 2020 07:10:32 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-100	-15.50	973652	1		
857	Mon, 19 Oct 2020 07:20:31 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-13.00	973650	1		
858	Mon, 19 Oct 2020 07:40:32 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-99	-16.20	973649	1		
859	Mon, 19 Oct 2020 07:45:30 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-48	13.80	973648	1		
860	Mon, 19 Oct 2020 08:03:21 GMT	RBS305-ATH-US	clever-orange-mustang	68 °F	54.3	8.22	51 °F	-45	9.80	973647	1	Air Temperature/Humidity	Periodic Report
861	Mon, 19 Oct 2020 08:05:32 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-107	-13.20	973646	1		
862	Mon, 19 Oct 2020 08:25:32 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-53	8.80	973645	1		
863	Mon, 19 Oct 2020 08:45:32 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-53	13.00	973644	1		
864	Mon, 19 Oct 2020 09:05:27 GMT	RBS305-ATH-US	clever-orange-mustang	68 °F	54.2	8.16	51 °F	-47	13.00	973643	1	Air Temperature/Humidity	Periodic Report
865	Mon, 19 Oct 2020 09:20:32 GMT	TBHV110	gorgeous-fuchsia-penguin	68 °F	53	7.97	50 °F	-106	-13.80	973642	1		
866	Mon, 19 Oct 2020 09:25:32 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-14.20	973641	1		
867	Mon, 19 Oct 2020 09:35:32 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-48	13.20	973640	1		
868	Mon, 19 Oct 2020 09:50:32 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-14.00	973624	1		
869	Mon, 19 Oct 2020 10:00:33 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-105	-12.00	973611	1		
870	Mon, 19 Oct 2020 10:07:36 GMT	RBS305-ATH-US	gorgeous-fuchsia-penguin	68 °F	54.1	8.09	51 °F	-41	13.00	973604	1	Air Temperature/Humidity	Periodic Report
871	Mon, 19 Oct 2020 10:35:33 GMT	TBHV110	clever-orange-mustang	68 °F	53	7.97	50 °F	-106	-13.80	973567	1		

(above) Screenshot of the Google Sheet used to capture data from the RBS305-ATH-US and TBHV110 sensors. The purpose of this sheet is to compare sensor performance.

Author – Mike Boucher, mikeboucher@yahoo.com

Appendix A – RBS305-ATH-US Decoder Function

Radio Bridge provides a generic decoder function (i.e. a decoder that supports multiple sensors) at the following GitHub repository: https://github.com/RadioBridge/Packet-Decoder/blob/master/radio_bridge_packet_decoder.js

That is what I am using – with the addition of a try/catch structure inside the Decoder function.

Below is a copy-n-paste of the contents of the Script I am using. Note that I chose **not** to try to remove the portions of the function that were for different sensors. I chose this approach simply because I didn't want to fix something that was already working.

```
//   RADIO BRIDGE PACKET DECODER v1.0
// (c) 2019 RadioBridge USA by John Sheldon

// General defines used in decode
var RESET_EVENT = "00";
var SUPERVISORY_EVENT = "01";
var TAMPER_EVENT = "02";
var LINK_QUALITY_EVENT = "FB";
var RATE_LIMIT_EXCEEDED_EVENT = "FC";
var TEST_MESSAGE_EVENT = "FD";
var DOWNLINK_ACK_EVENT = "FF";
var DOOR_WINDOW_EVENT = "03";
var PUSH_BUTTON_EVENT = "06";
var CONTACT_EVENT = "07";
var WATER_EVENT = "08";
var TEMPERATURE_EVENT = "09";
var TILT_EVENT = "0A";
var ATH_EVENT = "0D";
var ABM_EVENT = "0E";
var TILT_HP_EVENT = "0F";
var ULTRASONIC_EVENT = "10";
var SENSOR420MA_EVENT = "11";
var THERMOCOUPLE_EVENT = "13";
var VOLTMETER_EVENT = "14";
var CUSTOM_SENSOR_EVENT = "15";
var GPS_EVENT = "16";
var HONEYWELL5800_EVENT = "17";
var MAGNETOMETER_EVENT = "18";
var VIBRATION_LB_EVENT = "19";
var VIBRATION_HB_EVENT = "1A";



// Different network servers have different callback functions
// Each of these is mapped to the generic decoder function

// ------------------------------------------------

// function called by ChirpStack
function Decode(fPort, bytes, variables) {
   return Generic_Decoder(bytes, fPort);
}

// function called by TTN
function Decoder(bytes, port) {
try{
// try out tray-catch structure
return Generic_Decoder(bytes, port);
}
```

```
catch(e){
Decoder.err =e.message;
}

}
// ------------------------------------------------



// The generic decode function called by one of the above network server specific callbacks
function Generic_Decoder(bytes, port) {

   // data structure which contains decoded messages
   var decoded = {};

   // The first byte contains the protocol version (upper nibble) and packet counter (lower nibble)
   ProtocolVersion = (bytes[0] >> 4) & 0x0f;
   PacketCounter = bytes[0] & 0x0f;


   // the event type is defined in the second byte
   EventType = Hex(bytes[1]);

   // the rest of the message decode is dependent on the type of event
   switch (EventType) {

      // ==================    RESET EVENT    ====================
      case RESET_EVENT:

         decoded.Message = "Event: Reset";

         // third byte is device type, convert to hex format for case statement
         DeviceTypeByte = Hex(bytes[2]);

         // device types are enumerated below
         switch (DeviceTypeByte) {
            case "01": DeviceType = "Door/Window Sensor"; break;
            case "02": DeviceType = "Door/Window High Security"; break;
            case "03": DeviceType = "Contact Sensor"; break;
            case "04": DeviceType = "No-Probe Temperature Sensor"; break;
            case "05": DeviceType = "External-Probe Temperature Sensor"; break;
            case "06": DeviceType = "Single Push Button"; break;
            case "07": DeviceType = "Dual Push Button"; break;
            case "08": DeviceType = "Acceleration-Based Movement Sensor"; break;
            case "09": DeviceType = "Tilt Sensor"; break;
            case "0A": DeviceType = "Water Sensor"; break;
            case "0B": DeviceType = "Tank Level Float Sensor"; break;
            case "0C": DeviceType = "Glass Break Sensor"; break;
            case "0D": DeviceType = "Ambient Light Sensor"; break;
            case "0E": DeviceType = "Air Temperature and Humidity Sensor"; break;
            case "0F": DeviceType = "High-Precision Tilt Sensor"; break;
            case "10": DeviceType = "Ultrasonic Level Sensor"; break;
            case "11": DeviceType = "4-20mA Current Loop Sensor"; break;
            case "12": DeviceType = "Ext-Probe Air Temp and Humidity Sensor"; break;
            case "13": DeviceType = "Thermocouple Temperature Sensor"; break;
            case "14": DeviceType = "Voltage Sensor"; break;
            case "15": DeviceType = "Custom Sensor"; break;
            case "16": DeviceType = "GPS"; break;
            case "17": DeviceType = "Honeywell 5800 Bridge"; break;
            case "18": DeviceType = "Magnetometer"; break;
            case "19": DeviceType = "Vibration Sensor - Low Frequency"; break;
```

```
        case "1A": DeviceType = "Vibration Sensor - High Frequency"; break;
        default: DeviceType = "Device Undefined"; break;
      }

      decoded.Message += ", Device Type: " + DeviceType;

      // the hardware version has the major version in the upper nibble, and the minor version in the lower nibble
      HardwareVersion = ((bytes[3] >> 4) & 0x0f) + "." + (bytes[3] & 0x0f);

      decoded.Message += ", Hardware Version: v" + HardwareVersion;

      // the firmware version has two different formats depending on the most significant bit
      FirmwareFormat = (bytes[4] >> 7) & 0x01;

      // FirmwareFormat of 0 is old format, 1 is new format
      // old format is has two sections x.y
      // new format has three sections x.y.z
      if (FirmwareFormat == 0)
        FirmwareVerison = bytes[4] + "." + bytes[5];
      else
        FirmwareVerison = ((bytes[4] >> 2) & 0x1F) + "." + ((bytes[4] & 0x03) + ((bytes[5] >> 5) & 0x07)) + "." + (bytes[5] & 0x1F);

      decoded.Message += ", Firmware Version: v" + FirmwareVerison;

      break;

    // ================  SUPERVISORY EVENT  ==================

    case SUPERVISORY_EVENT:
      decoded.Message = "Event: Supervisory";

      // note that the sensor state in the supervisory message is being depreciated, so those are not decoded here

      // battery voltage is in the format x.y volts where x is upper nibble and y is lower nibble
      BatteryLevel = ((bytes[4] >> 4) & 0x0f) + "." + (bytes[4] & 0x0f);

      decoded.Message += ", Battery Voltage: " + BatteryLevel + "V";

      // the accumulation count is a 16-bit value
      AccumulationCount = (bytes[9] * 256) + bytes[10];
      decoded.Message += ", Accumulation Count: " + AccumulationCount;


      // decode bits for error code byte
      TamperSinceLastReset = (bytes[2] >> 4) & 0x01;
      decoded.Message += ", Tamper Since Last Reset: " + TamperSinceLastReset;

      CurrentTamperState = (bytes[2] >> 3) & 0x01;
      decoded.Message += ", Current Tamper State: " + CurrentTamperState;

      ErrorWithLastDownlink = (bytes[2] >> 2) & 0x01;
      decoded.Message += ", Error With Last Downlink: " + ErrorWithLastDownlink;

      BatteryLow = (bytes[2] >> 1) & 0x01;
      decoded.Message += ", Battery Low: " + BatteryLow;

      RadioCommError = bytes[2] & 0x01;
      decoded.Message += ", Radio Comm Error: " + RadioCommError;

      break;
```

```
    // ==================   TAMPER EVENT    ====================
    case TAMPER_EVENT:
      decoded.Message = "Event: Tamper";

      TamperState = bytes[2];

      // tamper state is 0 for open, 1 for closed
      if (TamperState == 0)
        decoded.Message += ", State: Open";
      else
        decoded.Message += ", State: Closed";

      break;

    // ==================   LINK QUALITY EVENT    ====================
    case LINK_QUALITY_EVENT:
      decoded.Message = "Event: Link Quality";

      CurrentSubBand = bytes[2];
      decoded.Message += ", Current Sub-Band: " + CurrentSubBand;

      RSSILastDownlink = bytes[3];
      decoded.Message += ", RSSI of Last Downlink: " + RSSILastDownlink;

      SNRLastDownlink = bytes[4];
      decoded.Message += ", SNR of Last Downlink: " + SNRLastDownlink;

      break;

    // ==================   RATE LIMIT EXCEEDED EVENT    ====================
    case RATE_LIMIT_EXCEEDED_EVENT:

      // this feature is depreciated so it is not decoded here
      decoded.Message = "Event: Rate Limit Exceeded. Depreciated Event And Not Decoded Here";

      break;

    // ==================   TEST MESSAGE EVENT    ====================
    case TEST_MESSAGE_EVENT:

      // this feature is depreciated so it is not decoded here
      decoded.Message = "Event: Test Message. Depreciated Event And Not Decoded Here";

      break;


    // ================  DOOR/WINDOW EVENT  ====================
    case DOOR_WINDOW_EVENT:

      decoded.Message = "Event: Door/Window";

      SensorState = bytes[2];

      // 0 is closed, 1 is open
      if (SensorState == 0)
        decoded.Message += ", State: Closed";
      else
        decoded.Message += ", State: Open";

      break;
```


Author – Mike Boucher, mikeboucher@yahoo.com

```
    // ===============  PUSH BUTTON EVENT   ===================
    case PUSH_BUTTON_EVENT:

      decoded.Message = "Event: Push Button";

      ButtonID = Hex(bytes[2]);

      switch (ButtonID) {
        // 01 and 02 used on two button
        case "01": ButtonReference = "Button 1"; break;
        case "02": ButtonReference = "Button 2"; break;
        // 03 is single button
        case "03": ButtonReference = "Button 1"; break;
        // 12 when both buttons pressed on two button
        case "12": ButtonReference = "Both Buttons"; break;
        default: ButtonReference = "Undefined"; break;
      }

      decoded.Message += ", Button ID: " + ButtonReference;

      ButtonState = bytes[3];

      switch (ButtonState) {
        case 0: SensorStateDescription = "Pressed"; break;
        case 1: SensorStateDescription = "Released"; break;
        case 2: SensorStateDescription = "Held"; break;
        default: SensorStateDescription = "Undefined"; break;
      }

      decoded.Message += ", Button State: " + SensorStateDescription;

      break;

    // =================   CONTACT EVENT   =====================
    case CONTACT_EVENT:

      decoded.Message = "Event: Dry Contact";

      ContactState = bytes[2];

      // if state byte is 0 then shorted, if 1 then opened
      if (ContactState == 0)
        SensorState = "Contacts Shorted";
      else
        SensorState = "Contacts Opened";

      decoded.Message += ", Sensor State: " + SensorState;

      break;

    // ===================  WATER EVENT  ======================
    case WATER_EVENT:

      decoded.Message = "Event: Water";

      SensorState = bytes[2];

      if (SensorState == 0)
        decoded.Message += ", State: Water Present";
      else
        decoded.Message += ", State: Water Not Present";
```

```
    WaterRelativeResistance = bytes[3];

    decoded.Message += ", Relative Resistance: " + WaterRelativeResistance;

    break;

  // ================== TEMPERATURE EVENT ====================
  case TEMPERATURE_EVENT:

    decoded.Message = "Event: Temperature";

    TemperatureEvent = bytes[2];

    switch (TemperatureEvent) {
      case 0: TemperatureEventDescription = "Periodic Report"; break;
      case 1: TemperatureEventDescription = "Temperature Over Upper Threshold"; break;
      case 2: TemperatureEventDescription = "Temperature Under Lower Threshold"; break;
      case 3: TemperatureEventDescription = "Temperature Report-on-Change Increase"; break;
      case 4: TemperatureEventDescription = "Temperature Report-on-Change Decrease"; break;
      default: TemperatureEventDescription = "Undefined"; break;
    }

    decoded.Message += ", Temperature Event: " + TemperatureEventDescription;

    // current temperature reading
    CurrentTemperature = Convert(bytes[3], 0);
    decoded.Message += ", Current Temperature: " + CurrentTemperature;

    // relative temp measurement for use with an alternative calibration table
    RelativeMeasurement = Convert(bytes[4], 0);
    decoded.Message += ", Relative Measurement: " + RelativeMeasurement;

    break;

  // ====================  TILT EVENT  ======================
  case TILT_EVENT:

    decoded.Message = "Event: Tilt";

    TiltEvent = bytes[2];

    switch (TiltEvent) {
      case 0: TiltEventDescription = "Transitioned to Vertical"; break;
      case 1: TiltEventDescription = "Transitioned to Horizontal"; break;
      case 2: TiltEventDescription = "Report-on-Change Toward Vertical"; break;
      case 3: TiltEventDescription = "Report-on-Change Toward Horizontal"; break;
      default: TiltEventDescription = "Undefined"; break;
    }

    decoded.Message += ", Tilt Event: " + TiltEventDescription;

    TiltAngle = bytes[3];

    decoded.Message += ", Tilt Angle: " + TiltAngle;

    break;

  // =============  AIR TEMP & HUMIDITY EVENT  ===============
  case ATH_EVENT:
```

```
    decoded.Message = "Event: Air Temperature/Humidity";

    ATHEvent = bytes[2];

    switch (ATHEvent) {
      case 0: ATHDescription = "Periodic Report"; break;
      case 1: ATHDescription = "Temperature has Risen Above Upper Threshold"; break;
      case 2: ATHDescription = "Temperature has Fallen Below Lower Threshold"; break;
      case 3: ATHDescription = "Temperature Report-on-Change Increase"; break;
      case 4: ATHDescription = "Temperature Report-on-Change Decrease"; break;
      case 5: ATHDescription = "Humidity has Risen Above Upper Threshold"; break;
      case 6: ATHDescription = "Humidity has Fallen Below Lower Threshold"; break;
      case 7: ATHDescription = "Humidity Report-on-Change Increase"; break;
      case 8: ATHDescription = "Humidity Report-on-Change Decrease"; break;
      default: ATHDescription = "Undefined"; break;
    }

    decoded.Message += ", ATH Event: " + ATHDescription;

    // integer and fractional values between two bytes
    Temperature = Convert((bytes[3]) + ((bytes[4] >> 4) / 10), 1);
    decoded.Message += ", Temperature: " + Temperature;

    // integer and fractional values between two bytes
    Humidity = +(bytes[5] + ((bytes[6]>>4) / 10)).toFixed(1);
    decoded.Message += ", Humidity: " + Humidity;

    break;

  // ============  ACCELERATION MOVEMENT EVENT  ==============
  case ABM_EVENT:

    decoded.Message = "Event: Acceleration-Based Movement";

    ABMEvent = bytes[2];

    if (ABMEvent == 0)
      ABMEventDescription = "Movement Started";
    else
      ABMEventDescription = "Movement Stopped";

    decoded.Message += ", ABM Event: " + ABMEventDescription;

    break;

  // =============  HIGH-PRECISION TILT EVENT  ===============
  case TILT_HP_EVENT:

    decoded.Message = "Event: High-Precision Tilt";

    TiltEvent = bytes[2];

    switch (TiltEvent) {
      case 0: TiltEventDescription = "Periodic Report"; break;
      case 1: TiltEventDescription = "Transitioned Toward 0-Degree Vertical Orientation"; break;
      case 2: TiltEventDescription = "Transitioned Away From 0-Degree Vertical Orientation"; break;
      case 3: TiltEventDescription = "Report-on-Change Toward 0-Degree Vertical Orientation"; break;
      case 4: TiltEventDescription = "Report-on-Change Away From 0-Degree Vertical Orientation"; break;
      default: TiltEventDescription = "Undefined"; break;
    }
```

```
        decoded.Message += ", Tilt HP Event: " + TiltEventDescription;

        // integer and fractional values between two bytes
        Angle = +(bytes[3] + (bytes[4] / 10)).toFixed(1);
        decoded.Message = ", Angle: " + Angle;

        Temperature = Convert(bytes[5], 0);
        decoded.Message = ", Temperature: " + Temperature;

        break;

    // ===============  ULTRASONIC LEVEL EVENT  ================
    case ULTRASONIC_EVENT:

        decoded.Message = "Event: Ultrasonic Level";

        UltrasonicEvent = bytes[2];

        switch (UltrasonicEvent) {
            case 0: UltrasonicEventDescription = "Periodic Report"; break;
            case 1: UltrasonicEventDescription = "Distance has Risen Above Upper Threshold"; break;
            case 2: UltrasonicEventDescription = "Distance has Fallen Below Lower Threshold"; break;
            case 3: UltrasonicEventDescription = "Report-on-Change Increase"; break;
            case 4: UltrasonicEventDescription = "Report-on-Change Decrease"; break;
            default: UltrasonicEventDescription = "Undefined"; break;
        }

        decoded.Message += ", Ultrasonic Event: " + UltrasonicEventDescription;

        // distance is calculated across 16-bits
        Distance = ((bytes[3] * 256) + bytes[4]);

        decoded.Message += ", Distance: " + Distance;
        break;

    // ===============  4-20mA ANALOG EVENT  ==================
    case SENSOR420MA_EVENT:

        decoded.Message = "Event: 4-20mA";

        Sensor420mAEvent = bytes[2];

        switch (Sensor420mAEvent) {
            case 0: Sensor420mAEventDescription = "Periodic Report"; break;
            case 1: Sensor420mAEventDescription = "Analog Value has Risen Above Upper Threshold"; break;
            case 2: Sensor420mAEventDescription = "Analog Value has Fallen Below Lower Threshold"; break;
            case 3: Sensor420mAEventDescription = "Report on Change Increase"; break;
            case 4: Sensor420mAEventDescription = "Report on Change Decrease"; break;
            default: Sensor420mAEventDescription = "Undefined"; break;
        }

        decoded.Message += ", 4-20mA Event: " + Sensor420mAEventDescription;

        // calculatec across 16-bits, convert from units of 10uA to mA
        Analog420Measurement = ((bytes[3] * 256) + bytes[4]) / 100;

        decoded.Message += ", Current Measurement in mA: " + Analog420Measurement;

        break;

    // ================  THERMOCOUPLE EVENT  ==================
```

```
    case THERMOCOUPLE_EVENT:

      decoded.Message = "Event: Thermocouple";

      ThermocoupleEvent = bytes[2];

      switch (ThermocoupleEvent) {
        case 0: ThermocoupleEventDescription = "Periodic Report"; break;
        case 1: ThermocoupleEventDescription = "Analog Value has Risen Above Upper Threshold"; break;
        case 2: ThermocoupleEventDescription = "Analog Value has Fallen Below Lower Threshold"; break;
        case 3: ThermocoupleEventDescription = "Report on Change Increase"; break;
        case 4: ThermocoupleEventDescription = "Report on Change Decrease"; break;
        default: ThermocoupleEventDescription = "Undefined"; break;
      }

      decoded.Message += ", Thermocouple Event: " + ThermocoupleEventDescription;

      // decode is across 16-bits
      Temperature = parseInt(((bytes[3] * 256) + bytes[4]) / 16);

      decoded.Message += ", Temperature: " + Temperature + "°C";

      Faults = bytes[5];

      // decode each bit in the fault byte
      FaultColdOutsideRange = (Faults >> 7) & 0x01;
      FaultHotOutsideRange = (Faults >> 6) & 0x01;
      FaultColdAboveThresh = (Faults >> 5) & 0x01;
      FaultColdBelowThresh = (Faults >> 4) & 0x01;
      FaultTCTooHigh = (Faults >> 3) & 0x01;
      FaultTCTooLow = (Faults >> 2) & 0x01;
      FaultVoltageOutsideRange = (Faults >> 1) & 0x01;
      FaultOpenCircuit = Faults & 0x01;

      // Decode faults
      if (Faults == 0)
        decoded.Message += ", Fault: None";
      else {
        if (FaultColdOutsideRange)
          decoded.Message += ", Fault: The cold-Junction temperature is outside of the normal operating range";

        if (FaultHotOutsideRange)
          decoded.Message += ", Fault: The hot junction temperature is outside of the normal operating range";

        if (FaultColdAboveThresh)
          decoded.Message += ", Fault: The cold-Junction temperature is at or above than the cold-junction temperature high threshold";

        if (FaultColdBelowThresh)
          decoded.Message += ", Fault: The Cold-Junction temperature is lower than the cold-junction temperature low threshold";

        if (FaultTCTooHigh)
          decoded.Message += ", Fault: The thermocouple temperature is too high";

        if (FaultTCTooLow)
          decoded.Message += ", Fault: Thermocouple temperature is too low";

        if (FaultVoltageOutsideRange)
          decoded.Message += ", Fault: The input voltage is negative or greater than VDD";

        if (FaultOpenCircuit)
          decoded.Message += ", Fault: An open circuit such as broken thermocouple wires has been detected";
```

```
        }

        break;

    // ================  VOLTMETER ANALOG EVENT  ==================
    case VOLTMETER_EVENT:

        decoded.Message = "Event: Voltage Sensor";

        VoltmeterEvent = bytes[2];

        switch (VoltmeterEvent) {
            case 0: VoltmeterEventDescription = "Periodic Report"; break;
            case 1: VoltmeterEventDescription = "Voltage has Risen Above Upper Threshold"; break;
            case 2: VoltmeterEventDescription = "Voltage has Fallen Below Lower Threshold"; break;
            case 3: VoltmeterEventDescription = "Report on Change Increase"; break;
            case 4: VoltmeterEventDescription = "Report on Change Decrease"; break;
            default: VoltmeterEventDescription = "Undefined";
        }

        decoded.Message += ", Voltage Sensor Event: " + VoltmeterEventDescription;

        // voltage is measured across 16-bits, convert from units of 10mV to V
        VoltageMeasurement = ((bytes[3] * 256) + bytes[4]) / 100;

        decoded.Message += ", Voltage: " + VoltageMeasurement + "V";
        break;


    // ================  CUSTOM SENSOR EVENT  ==================
    case CUSTOM_SENSOR_EVENT:

        decoded.Message = "Event: Custom Sensor";

        // Custom sensors are not decoded here

        break;


    // ================  VOLTMETER ANALOG EVENT  ==================
    case GPS_EVENT:

        decoded.Message = "Event: GPS";

        GPSStatus = bytes[2];

        // decode status byte
        GPSValidFix = GPSStatus & 0x01;

        if (GPSValidFix == 0)
            GPSValidFixDescription = ", No Valid Fix";
        else
            GPSValidFixDescription = ", Valid Fix";


        decoded.Message += ", GPS Status: " + GPSValidFixDescription;

        // latitude and longitude calculated across 32 bits each, show 12 decimal places
        Latitude = toFixed((((bytes[3] * (2 ^ 24)) + (bytes[4] * (2 ^ 16)) + (bytes[5] * (2 ^ 8)) + bytes[6]) / (10 ^ 7)), 12);
        Latitude = toFixed((((bytes[7] * (2 ^ 24)) + (bytes[8] * (2 ^ 16)) + (bytes[9] * (2 ^ 8)) + bytes[10]) / (10 ^ 7)), 12);
```

```
        decoded.Message += ", Latitude: " + Latitude + ", Longitude: " + Longitude;

        break;


    // ================  HONEYWELL 5800 EVENT  ==================
    case HONEYWELL5800_EVENT:

        decoded.Message = "Event: Honeywell 5800 Sensor Message";

        // honeywell sensor ID, 24-bits
        HWSensorID = (bytes[2] * (2 ^ 16)) + (bytes[3] * (2 ^ 8)) + bytes[4];

        decoded.Message += ", Honeywell Sensor ID: " + HWSensorID;

        HWEvent = bytes[5];

        switch (HWEvent) {
            case 0: HWEventDescription = "Status code"; break;
            case 1: HWEventDescription = "Error Code"; break;
            case 2: HWEventDescription = "Sensor Data Payload"; break;
            default: HWEventDescription = "Undefined"; break;
        }

        decoded.Message += ", Honeywell Sensor Event: " + HWEventDescription;

        // represent the honeywell sensor payload in hex
        HWSensorPayload = Hex((bytes[6] * 256) + bytes[7]);

        decoded.Message += ", Sensor Payload: 0x" + HWSensorPayload;

        break;


    // ================  MAGNETOMETER EVENT  ==================
    case MAGNETOMETER_EVENT:

        // TBD

        break;


    // ================  VIBRATION LOW BANDWIDTH EVENT  ==================
    case VIBRATION_LB_EVENT:

        decoded.Message = "Event: Vibration Low-Bandwidth";

        VibeEvent = bytes[2];

        switch (VibeEvent) {
            case 0: VibeEventDescription = "Low Frequency Periodic Report"; break;
            case 4: VibeEventDescription = "Low Frequency X-Axis Has Risen Above Upper Threshold"; break;
            case 5: VibeEventDescription = "Low Frequency X-Axis Has Fallen Below Lower Threshold"; break;
            case 6: VibeEventDescription = "Low Frequency Y-Axis Has Risen Above Upper Threshold"; break;
            case 7: VibeEventDescription = "Low Frequency Y-Axis Has Fallen Below Lower Threshold"; break;
            case 8: VibeEventDescription = "Low Frequency Z-Axis Has Risen Above Upper Threshold"; break;
            case 9: VibeEventDescription = "Low Frequency Z-Axis Has Fallen Below Lower Threshold"; break;
            case 11: VibeEventDescription = "Low Frequency Exceeded G-Force Range"; break;
            default: VibeEventDescription = "Undefined"; break;
        }
```

```
        decoded.Message += ", Vibration Event: " + VibeEventDescription;

        // X, Y, and Z velocities are 16-bits
        XVelocity = (bytes[3] * 256) + bytes[4];
        YVelocity = (bytes[5] * 256) + bytes[6];
        ZVelocity = (bytes[7] * 256) + bytes[8];

        decoded.Message += ", X-Axis Velocity: " + XVelocity + " inches/second";
        decoded.Message += ", Y-Axis Velocity: " + YVelocity + " inches/second";
        decoded.Message += ", Z-Axis Velocity: " + ZVelocity + " inches/second";

        // capture sign of temp
        VibeTemp = parseInt(bytes[9]);

        decoded.Message = ", Internal Temperature: " + VibeTemp + "°C";

        break;

    // ================  VIBRATION HIGH BANDWIDTH EVENT  ==================
    case VIBRATION_HB_EVENT:

        decoded.Message = "Event: Vibration Low-Bandwidth";

        VibeEvent = bytes[2];

        switch (VibeEvent) {
          case 1: VibeEventDescription = "High Frequency Periodic Report"; break;
          case 2: VibeEventDescription = "High Frequency Vibration Above Upper Threshold"; break;
          case 3: VibeEventDescription = "High Frequency Vibration Below Lower Threshold"; break;
          case 10: VibeEventDescription = "High Frequency Exceeded G-Force Range"; break;
          default: VibeEventDescription = "Undefined"; break;
        }

        decoded.Message += ", Vibration Event: " + VibeEventDescription;

        // peak g-force
        PeakGForce = (bytes[3] * 256) + bytes[4];

        decoded.Message += ", Peak G-Force: " + PeakGForce;

        // capture sign of temp
        VibeTemp = parseInt(bytes[5]);

        decoded.Message = ", Internal Temperature: " + VibeTemp + "°C";

        break;


    // ================  DOWNLINK EVENT  ==================
    case DOWNLINK_ACK_EVENT:

        decoded.Message = "Event: Downlink Acknowledge";

        DownlinkEvent = bytes[2];

        if (DownlinkEvent == 1)
          DownlinkEventDescription = "Message Invalid";
        else
          DownlinkEventDescription = "Message Valid";

        decoded.Message += ", Downlink: " + DownlinkEventDescription;
```

```
        break;

      // end of EventType Case
    }

    // add packet counter and protocol version to the end of the decode
    decoded.Message += ", Packet Counter: " + PacketCounter;
    decoded.Message += ", Protocol Version: " + ProtocolVersion;

    return decoded;
}

function Hex(decimal) {
    decimal = ('0' + decimal.toString(16).toUpperCase()).slice(-2);
    return decimal;
}

function Convert(number, mode) {
    switch (mode) {
      // for EXT-TEMP and NOP
      case 0: if (number > 127) { result = number - 256 } else { result = number }; break
      //for ATH temp
      case 1: if (number > 127) { result = -+(number - 128).toFixed(1) } else { result = +number.toFixed(1) }; break
    }
    return result;
}
```