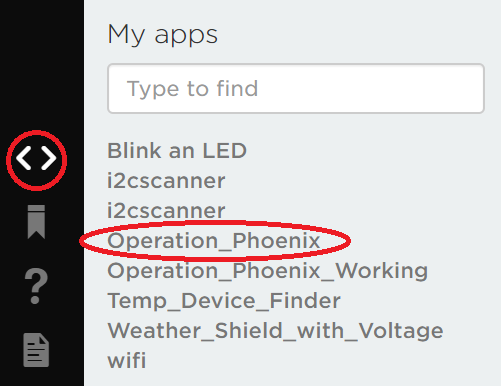
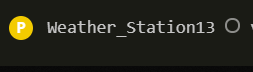
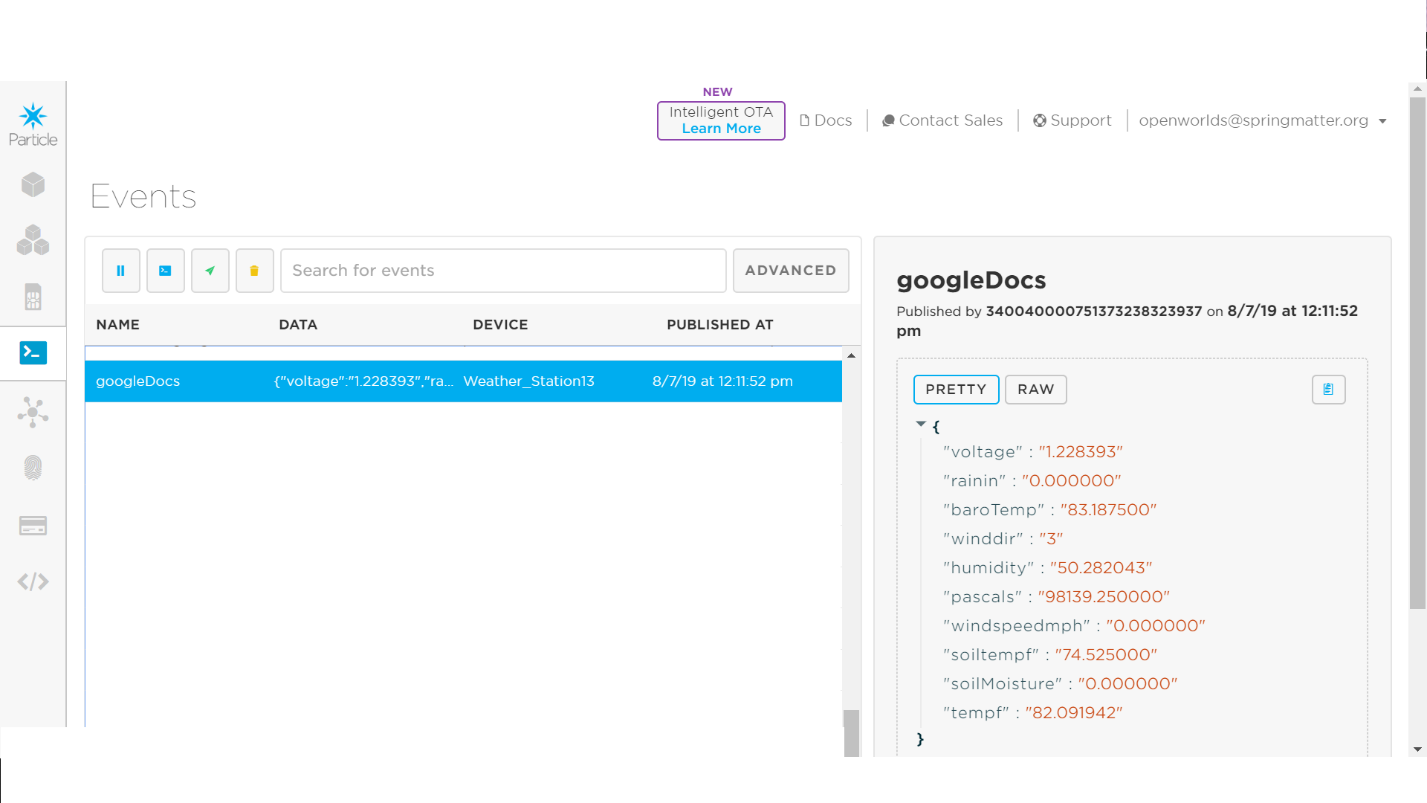
Getting the Data from the Photon

1. Go to build.particle.io
2. First you will need to flash the Operation\_Phoenix code to your device. To do so, you should click the two arrows on the side of the screen then click on Operation\_Phoenix. 
3. Once you have selected the correct device in the bottom right corner, click on the lightning bolt in the top left corner.

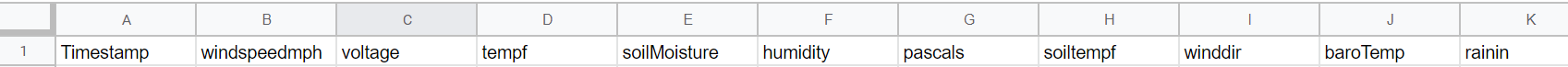
1. Once you have flashed the code to your device, the photon will begin to send data to Particle.io because of the Particle.publish line in the code.



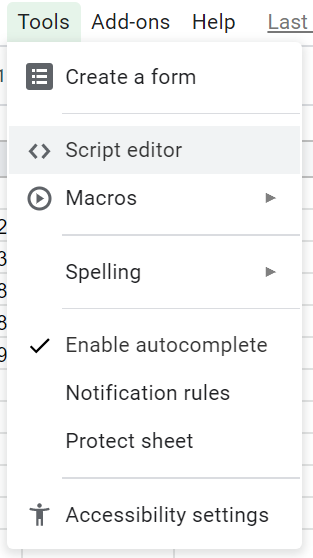
1. You can see data being published in the Particle.io Console

Setting Up Google Sheets

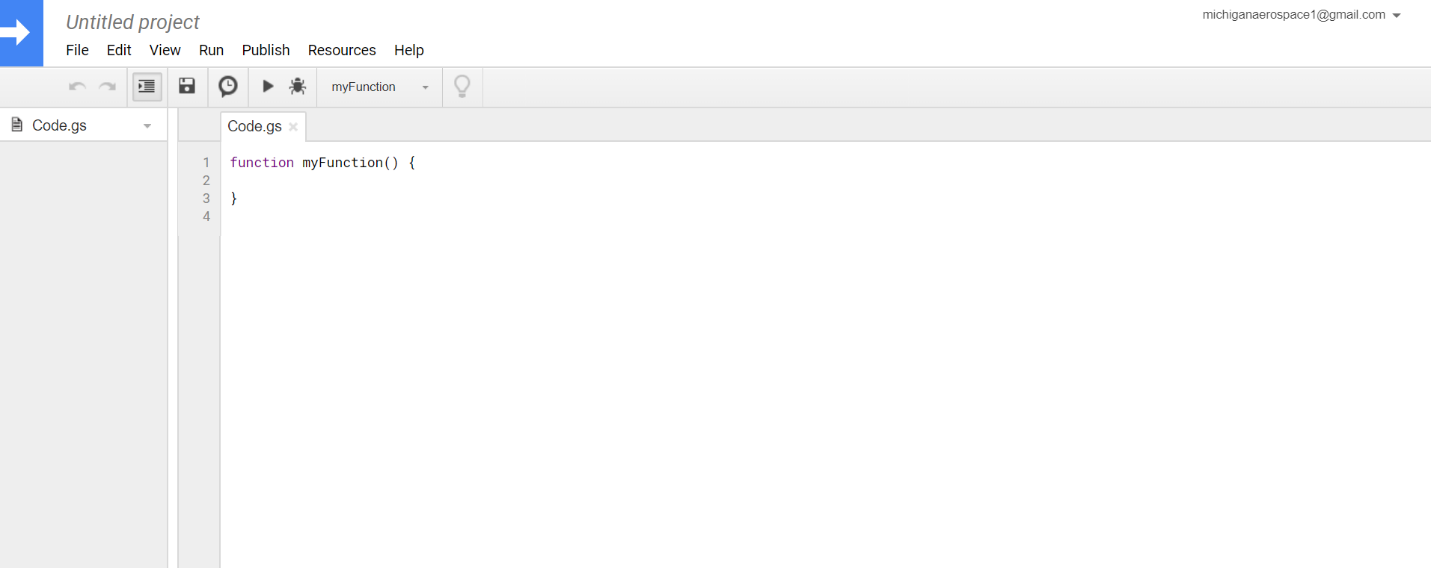
1. Once there is data streaming into Particle.io you can set up a Google sheet for a webhook to send this data to.
2. Login to the email: [michiganaerospace1@gmail.com](mailto:michiganaerospace1@gmail.com) password: michaero
3. Create a new Google Sheet in Google Drive and Name it (Weather Station #)
4. Once you open your google sheet, you need to name the columns as shown below



1. Next you need to open the script editor in Google Sheets. To do this click on the Tools tab at the top, the click on Script editor



1. A Screen like this will pop up. Give the project a name (Weather Station # Project)



1. Delete what is currently under Code.gs and paste this code:

// original from: http://mashe.hawksey.info/2014/07/google-sheets-as-a-database-insert-with-apps-script-using-postget-methods-with-ajax-example/

function doGet(e){

return handleResponse(e);

}

// Usage

// 1. Enter sheet name where data is to be written below

var SHEET\_NAME = "Sheet1";

// 2. Run > setup

//

// 3. Publish > Deploy as web app

// - enter Project Version name and click 'Save New Version'

// - set security level and enable service (most likely execute as 'me' and access 'anyone, even anonymously)

//

// 4. Copy the 'Current web app URL' and post this in your form/script action

//

// 5. Insert column names on your destination sheet matching the parameter names of the data you are passing in (exactly matching case)

var SCRIPT\_PROP = PropertiesService.getScriptProperties(); // new property service

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

function doPost(e){

return handleResponse(e);

}

function handleResponse(e) {

// shortly after my original solution Google announced the LockService[1]

// this prevents concurrent access overwritting data

// [1] http://googleappsdeveloper.blogspot.co.uk/2011/10/concurrency-and-google-apps-script.html

// we want a public lock, one that locks for all invocations

var lock = LockService.getPublicLock();

lock.waitLock(30000); // wait 30 seconds before conceding defeat.

try {

// next set where we write the data - you could write to multiple/alternate destinations

var doc = SpreadsheetApp.openById(SCRIPT\_PROP.getProperty("key"));

var sheet = doc.getSheetByName(SHEET\_NAME);

// we'll assume header is in row 1 but you can override with header\_row in GET/POST data

var headRow = e.parameter.header\_row || 1;

var headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

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

var row = [];

// loop through the header columns

for (i in headers){

if (headers[i] == "Timestamp"){ // special case if you include a 'Timestamp' column

row.push(new Date());

} else { // else use header name to get data

row.push(e.parameter[headers[i]]);

}

}

// more efficient to set values as [][] array than individually

sheet.getRange(nextRow, 1, 1, row.length).setValues([row]);

// return json success results

return ContentService

.createTextOutput(JSON.stringify({"result":"success", "row": nextRow}))

.setMimeType(ContentService.MimeType.JSON);

} catch(e){

// if error return this

return ContentService

.createTextOutput(JSON.stringify({"result":"error", "error": e}))

.setMimeType(ContentService.MimeType.JSON);

} finally { //release lock

lock.releaseLock();

}

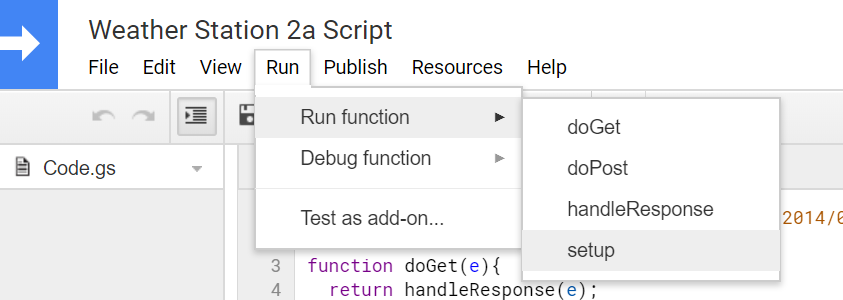
}

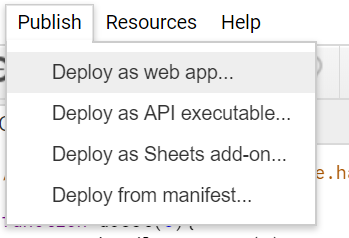
function setup() {

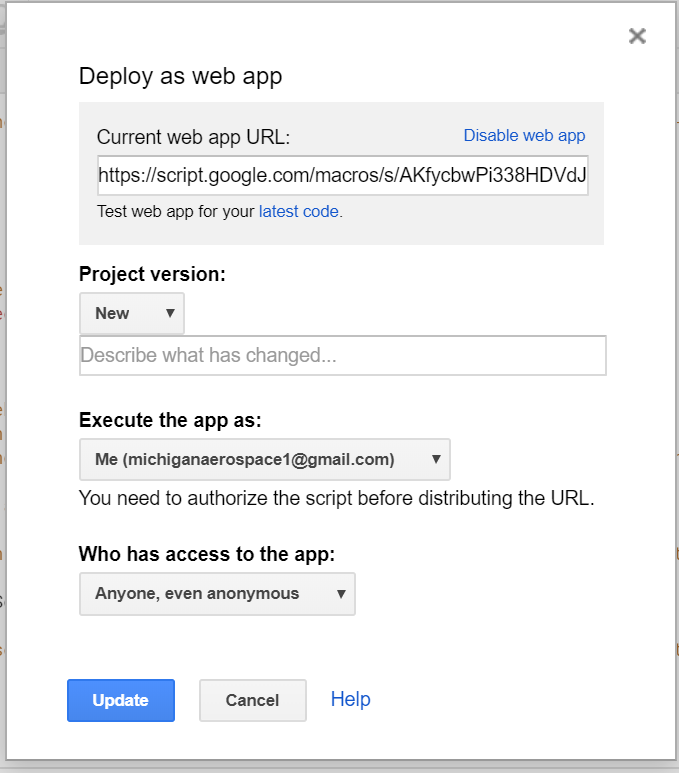
var doc = SpreadsheetApp.getActiveSpreadsheet();

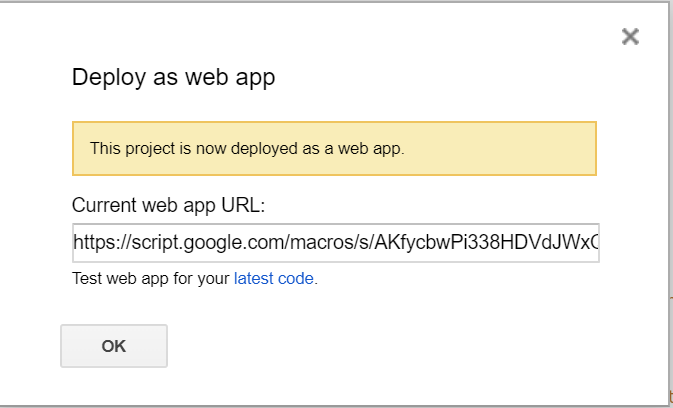
SCRIPT\_PROP.setProperty("key", doc.getId());

}

1. Save this code then Run setup. 
2. Next, click on the Publish tab and the Deploy as a Web App



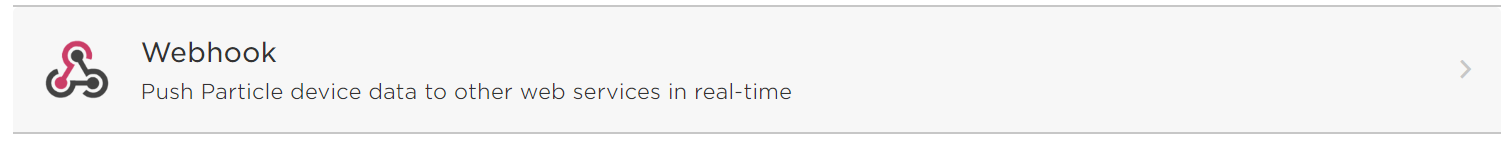
1. That will bring up this screen: 
2. Set the Project version to new, the execute app as to Me and the access to anyone even anonymous. Click Update.
3. That will bring up this screen:

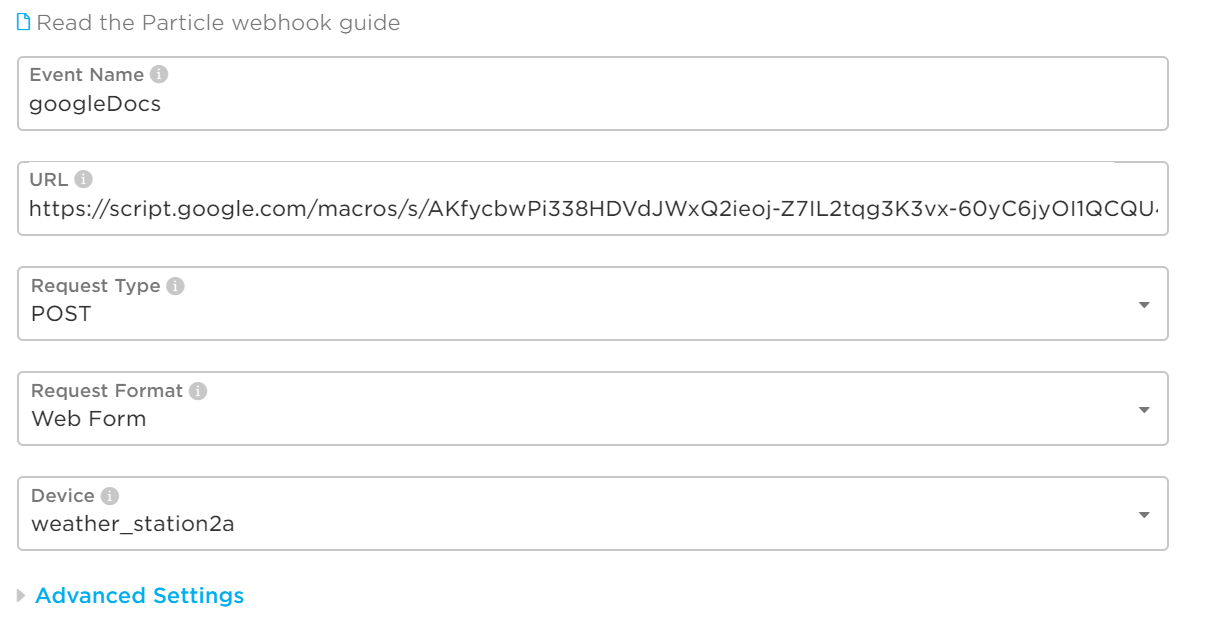
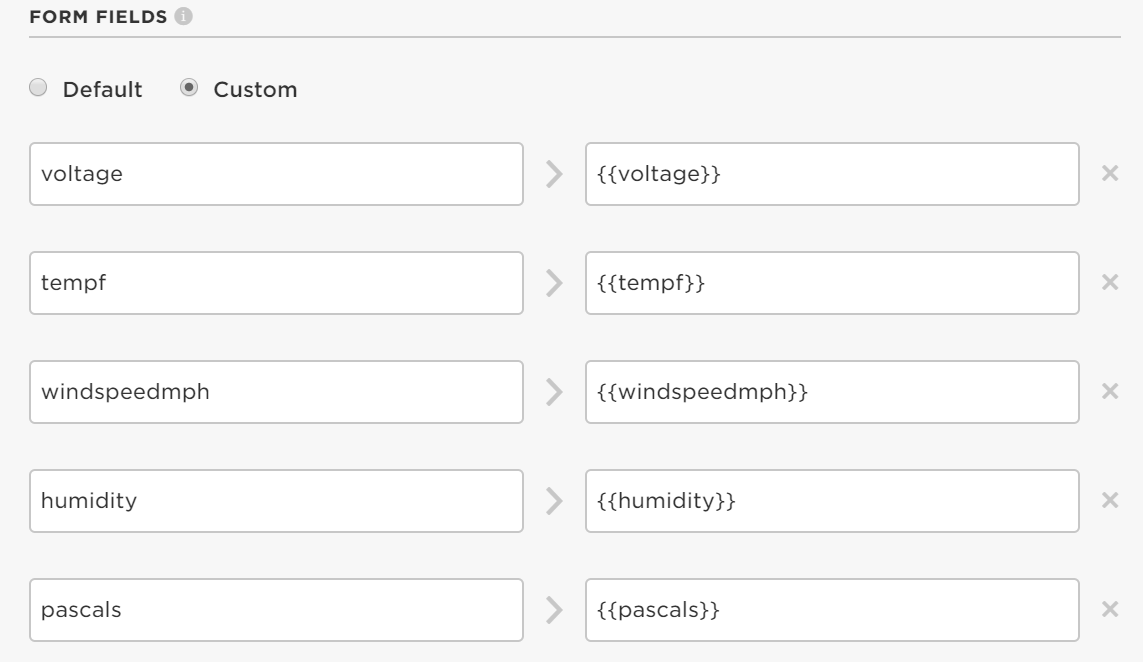
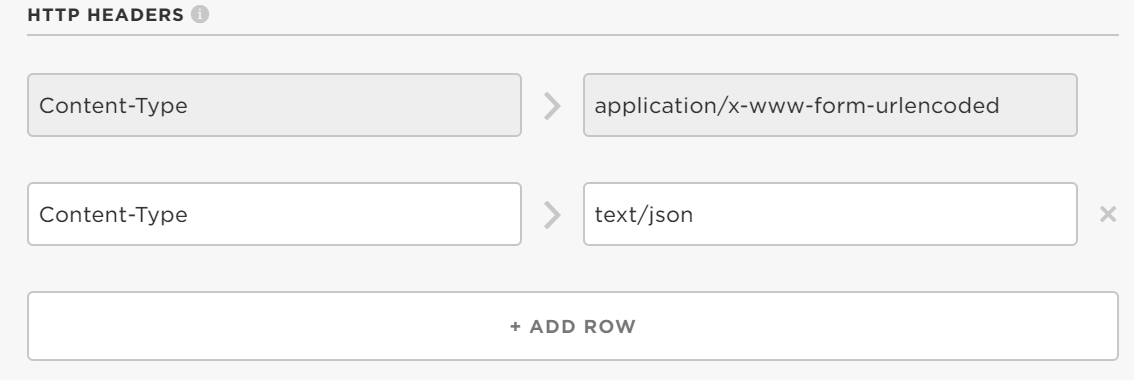
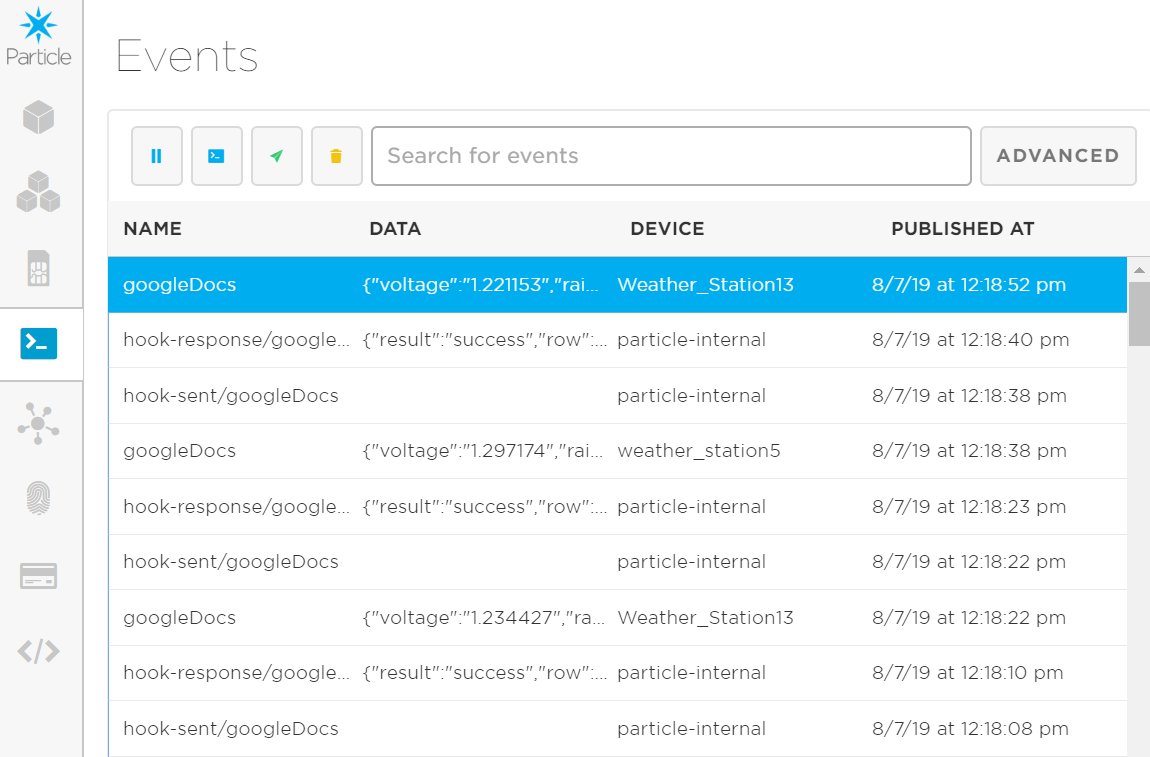
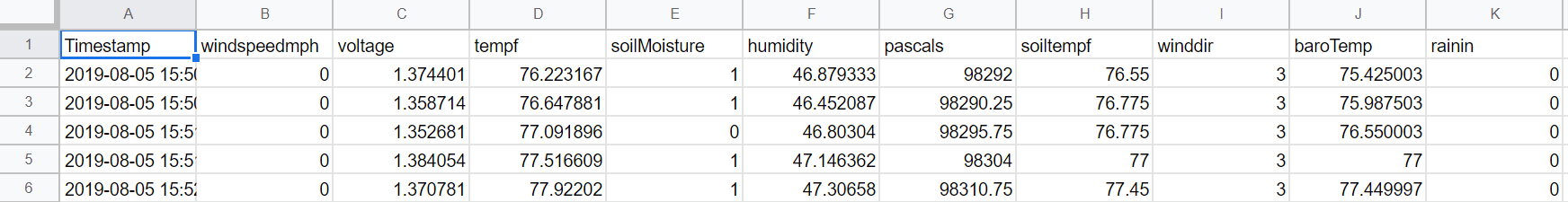


1. Copy the Current web app URL to be used in a Particle Webhook and click OK

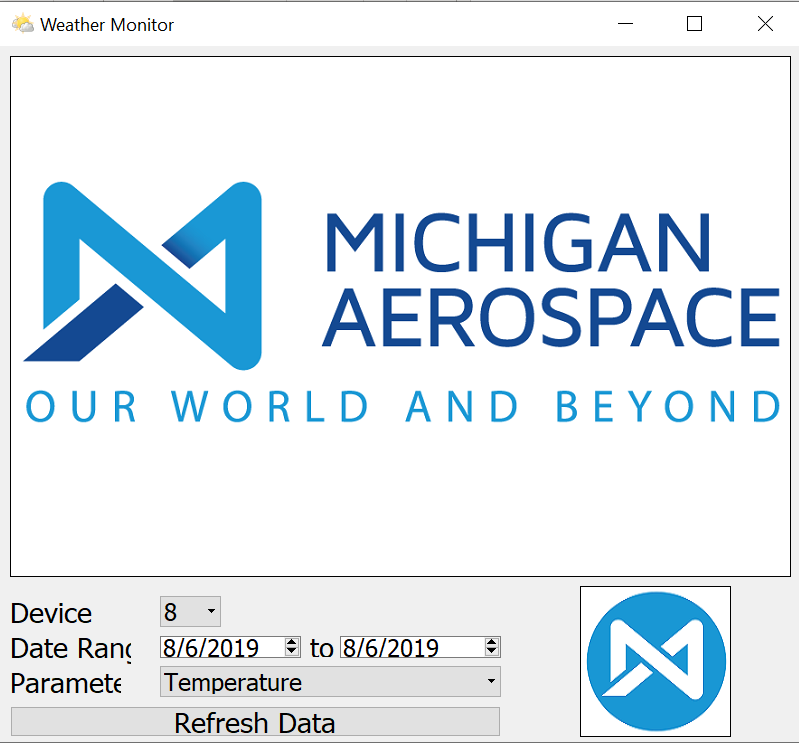
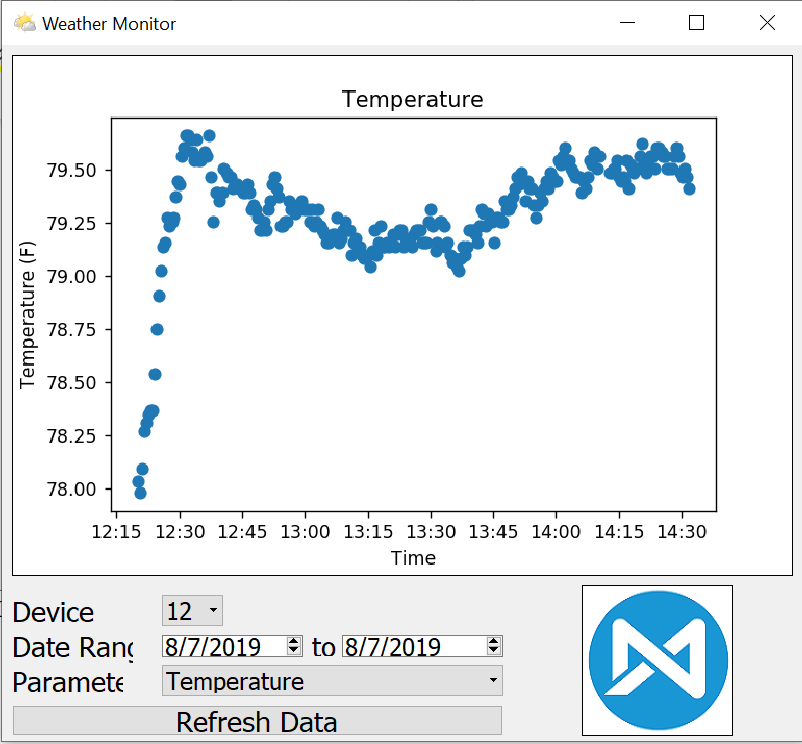
Setting Up a Particle Webhook

1. Open the Particle.io console and click on integrations 
2. Click on new integration 
3. Click on Webhook



1. Enter the event name googleDocs. Paste the URL you copied from the google script. Select your device.
2. Click on Advanced Settings. Enter each field as seen below
3. Under Http Headers enter as seen below 
4. At the bottom of the page click save 
5. You should now be able to go to the events tab on the Particle.io console and see hooks being sent to Google Sheets
6. Data should now be streaming in your Google Sheet

Using the App to Visualize Data

1. Open the python program on your laptop (in this case Spyder)
2. From the weather.ui folder open app.py
3. Run app.py, a separate window will pop up
4. From this window you can select your device, the date range you wish to look at and the parameter you are interested in. 
5. Click on Refresh Data to see the graph
6. This app downloads the data from google sheets and uses it to create these graphs. When you refresh data, it will take the newest points in the data range from google sheets if it doesn’t already have them.