Prepare the Google Spreadsheet.

Create the Spreadsheet and Copy the URL

1. Open Google Drive
2. Click on: “New”. “Google Sheets”
3. On the new sheet, fill in column headers on first row; column A, B, …
4. Click on “Untitled spreadsheet” and change the name to whatever you want to call it (WSMSantaRosaData).
5. Copy the URL of the spreadsheet.
   Copy the url here:

   https://docs.google.com/spreadsheets/d/…./edit#gid=0

6. Note the name of the current sheet within the spreadsheet. The default name is: “Sheet 1”.
   Copy the sheet name here:
   …….

Create the Google Apps Script.

1. Open Google Drive
2. Click on “Google Apps Script”
3. Click on “Untitled project” and change the project name to your chosen project name (WriteWSMSheet)
4. Replace the default “function myFunction() {}” with the script:

```
function doGet(e) {

 var ss = SpreadsheetApp.openByUrl("https://docs.google.com/spreadsheets/d/ … <your spreadsheet url> ...");
 var sheet = ss.getSheetByName("Sheet1");

 addUser(e,sheet);
}

function doPost(e) {
 var ss = SpreadsheetApp.openByUrl("https://docs.google.com/spreadsheets/d/ … <your spreadsheet url> ...");
 var sheet = ss.getSheetByName("Sheet1");
```

```
  addUser(e, sheet);
}

function addUser(e, sheet) {

  var edata = e.parameter.data;
  var wsmData = JSON.parse(edata);
  var ev = e.parameter.event;
  var pub = e.parameter.published_at;

  var time = wsmData.etime ;
  var temp = wsmData.temp ;
  var rh = wsmData.rh ;
  var pp = wsmData.pp ;
  var wp = wsmData.wp ;
  var ptm = wsmData.ppon ;
  var wtm = wsmData.wpon ;
  var loctm = wsmData.loctime ;

  sheet.appendRow([time,temp,rh,pp,wp,ptm,wtm,ev,loctm]);
}
```

5. On the lines beginning with “var ss = “, replace the default url with the url of the spreadsheet. Note: do this twice, once in “function doGet(e)” and then again in “function doPost(e)”.
6. On the lines beginning with “var sheet = “, replace “Sheet 1” with the top sheet of the spreadsheet, unless you left it named the default “Sheet 1”. Note: do this twice, once in “function doGet(e)” and then again in “function doPost(e)”.
7. In the function “addUser(e, sheet){}”, make sure that the the variable names correspond to your API calling names and the these names are the parameters in the “function appendRows([...])”, in the order that you want these values on your spreadsheet.
8.

Publish the Script and Set Permissions

1. USE THE **OLD SCRIPT EDITOR**: click on “publish”, “Deploy as web app”
2. You can leave “Project version” as the default
3. Your can leave “”Execute the app as” as the default (your Google account)
4. Change “Who has access to the app” to “Anyone, even anonymous”
5. Click “Deploy”

6. You will get a popup “Authorization required” and click “Review permissions”
7. You will get a popup “Choose an Account”. Click on the Google account that you used to create the spreadsheet.
8. You will get a popup with the error “This app isn’t verified”. Click on “Advanced” at the bottom.
9. Click on “Go to <your script name> (unsafe)” at the bottom.
10. On the popup, click on “Allow”.
11. On the popup “Deploy as web app”, copy the script url (under “Current web app URL:”) here:

    https://script.google.com/macros/ … /exec

12. Then click on “OK”.
13. IF YOU NEED TO EDIT YOUR SCRIPT:
    a. Edit the script in the OLD script editor
    b. click on “publish”
    c. On the version pulldown list, select “new”
    d. Enter comment about the change
    e. Click on “Deploy”
    f. Permissions will remain as for the original version.

Test with cURL

1. Open “cmd” on Windows
2. Use cURL to write data to the spreadsheet, e.g.:

NOTE the construction of the json format data string! Need double quotes around the tags and need to use the escape character to delineate these:

curl -d
“data=’{\“etime\”:1567893,\”temp\”:77.5,\”rh\”:15.7,\”pp\”:1,\”wp\”:1,\”loctime\”:\”2021-06-15 16-03-20\”}’&event=WSMeventPPchange&published_at=2021-06-04-12-22” -X POST
https://script.google.com/macros/s/AKfycbzicwJ-AC-IAZLc8lFod0nDelHyrQbK59obgaiL1bGuvBVINR86/exec

curl -d
“data=’{\“etime\”:1234952,\”temp\”:77.0,\”rh\”:12.4,\”pp\”:0,\”ppon\”:1.23,\”wp\”:0,\”wpon\”:35.2,\”loctime\”:\”2021-06-15 16-27-20\”}’&event=WSMeventPPchange” -X POST
https://script.google.com/macros/s/AKfycbzicwJ-AC-IAZLc8lFod0nDelHyrQbK59obgaiL1bGuvBVINR86/exec

Configure the Particle Webhook

Event Name: “wsmEvent”

Full URL: the url of the script
Request Type: “POST”
Request Format: “Web Form”
Headers: “{`"Content-Type": "application/x-www-form-urlencoded"}`
Enforce SSL: “Yes”

Call the Webhook from Firmware

The firmware must create a JSON string with the data to be published to the Google App Script. The test firmware contains functions to create the JSON strings and perform the publication for new events of: new temp/rh reading, pressure pump change, well pump change. Here are these functions:

```
//  publish new temperature and humidity values
void publishTRH(float temp, float rh) {
  String eData = "";

  // build the data string with time, temp and rh values
  eData += "{\"etime\":";
  eData += String(Time.now());
  eData += ",\"temp\":";
  eData += String(temp);
  eData += ",\"rh\":";
  eData += String(rh);
  eData += ",\"loctime\":\"";
  eData += String(Time.format("%F %T"));
  eData += "\"}";

  // publish to the webhook
  Particle.publish("wsmEventTRH", eData, PRIVATE);

  return;
} // end of publishTRH()

//  publish pressure pump status change
void publishPPchange(int newPPstatus) {
  static unsigned long ppumpOnTimestamp;
  String eData = "";
  float pumpTime;

  // build the data string with time, pp value
  eData += "{\"etime\":";
```

```
  eData += String(Time.now());
  eData += ",\"pp\":";
  eData += String(newPPstatus);

  // computation of PP on time
  if(newPPstatus == 1) {  // the pump has come on
    ppumpOnTimestamp = millis();
    eData += ",\"loctime\":\"";
    eData += String(Time.format("%F %T"));
    eData += "\"}";
  }
  else {    // the pump has turned off
    eData += ",\"ppon\":";
    pumpTime = (float)(millis() - ppumpOnTimestamp)/60000.0;
    eData += String(pumpTime);
    eData += ",\"loctime\":\"";
    eData += String(Time.format("%F %T"));
    eData += "\"}";
  }

  // publish to the webhook
  Particle.publish("wsmEventPPstatus", eData, PRIVATE);

  return;
} // end of publishPPchange()

//  publish well pump status change
void publishWPchange(int newWPstatus) {
  static unsigned long wpumpOnTimestamp;
  String eData = "";
  float pumpTime;

  // build the data string with time, pp value
  eData += "{\"etime\":";
  eData += String(Time.now());
  eData += ",\"wp\":";
  eData += String(newWPstatus);

// computation of WP on time
  if(newWPstatus == 1) {  // the pump has come on
    wpumpOnTimestamp = millis();
    eData += ",\"loctime\":\"";
    eData += String(Time.format("%F %T"));
    eData += "\"}";
```

```
  }
  else {    // the pump has turned off
    eData += ",\"wpon\":";
    pumpTime = (float)(millis() - wpumpOnTimestamp)/60000;
    eData += String(pumpTime);
    eData += ",\"loctime\":\"";
    eData += String(Time.format("%F %T"));
    eData += "\"}";
  }

  // publish to the webhook
  Particle.publish("wsmEventWPstatus", eData, PRIVATE);

  return;
} // end of publishWPchange()
```