Google Sheets Integration Documentation

The Google Sheet is used as the database for all device communication in the system. That is, all communication between the app and the hardware components goes through the Google Sheet. This document outlines the way you can use the Google Sheet to communicate with the device and all related information required to facilitate this. It is recommended that the Google Sheet used for this project be replicated on the user's own account by replicating the layout of the two sheets 'deviceToApp' and 'appToDevice' as well as generation of a unique API key for communication. Note that for the communication outlined in this document, the Google Sheet must also be public. The Google Sheet used as a prototype can be found here: https://docs.google.com/spreadsheets/d/1RcYlmxQAznj83vKuDPzCUz2ENzumkxtC3WUML4Wr3Mg/edit

API Use

Generating a unique API key for a Google Sheet:
https://support.google.com/googleapi/answer/6158862?hl=en

In order to retrieve information from the database, we require access to Google Sheets API. This access is granted via the use of an API key. The API key is used for two forms of communication within the system. They are:

- Retrieving information about soil moisture and tank level for the app.
    - This process is done in Android Studio and works by making an HTTP GET request to the Google Sheet using the API key. This request then returns a JSON object that can be parsed and used for displaying information.

```
String sheetsID = "1RcYlmxQAznj83vKuDPzCUz2ENzumkxtC3WUML4Wr3Mg";
String apiKEY = "API KEY GOES HERE";
String urls =
"https://sheets.googleapis.com/v4/spreadsheets/"+sheetsID+"/
          values/deviceToApp?key="+apiKEY;
JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, urls, null, new
Response.Listener<JSONObject>(){}
```

    - The code above will make a request for the sheet with the given sheetsID. It is important to note that the JSON object will contain all data for the sheet labeled 'deviceToApp'.
- Retrieving commands from the app in the Arduino.
    - This process is done on the Arduino and is similar to the process outlined above for the app.

```
String apiURL =
"/v4/spreadsheets/1RcYlmxQAznj83vKuDPzCUz2ENzumkxtC3WUML4Wr3Mg/values
/appToDevice?key=API KEY GOES HERE";
char serverGet[] = "sheets.googleapis.com";
```

```
httpGet.beginRequest();
httpGet.get(apiURL);
httpGet.endRequest();
String response = httpGet.responseBody();
const size_t CAPACITY = JSON_OBJECT_SIZE(3)
      + JSON_ARRAY_SIZE(2) + 8*JSON_ARRAY_SIZE(8) + 280;
StaticJsonDocument<CAPACITY> doc;
DeserializationError error = deserializeJson(doc, response);
JsonArray values = doc["values"].as<JsonArray>();
```

- The code above works similarly to the app code in that it receives a JSON object for the sheet labeled ‘appToDevice’. The object is then converted to a multidimensional array that can be used like a regular array in C.

POSTs and Google Script

Posting data to the Google Sheet does not require an API, however it does utilize Google’s App Scripts functionality. There are two instances where data is posted on the sheet, when the app sends a command, and when the Arduino posts soil data and tank level information. Below is documentation on how this is done in Android Studio and Arduino as well as information regarding the Google Script.

- Posting from the app

```
StringRequest stringRequest = new StringRequest(Request.Method.POST, "https://script.google.com/macros/s/AKfycbx6iq-tczFTbGb3ddgn-uy9ihIHFXTG21i4Wk9aJ61q7GazfrxSAx7MycWCfSI9sQahIw/exec",
    @Override
    public void onResponse(String response) {
        progressDialog.hide();
    }
}, new Response.ErrorListener() {
    public void onErrorResponse(VolleyError error) {

    }
}){
    @Override
    protected Map<String, String> getParams() {
        Map<String, String> params = new HashMap<>();
        params.put("client", "app");
        params.put("vWaterAmount", size);
        params.put("plant", value);

        return params;
    }
};
```

- This code works by sending a POST request with key parameter pairings similar to an HTML form which are then parsed in the Google Script. Note it is not necessary to indicate which sheet this is posting to as this is also handled in the Google Script.

- Posting from Arduino:

```
// Posting data to the Google Sheet
void postData(String data) {
  httpPost.beginRequest();
  httpPost.post(scriptURL, contentType, data);
  int statusCode = httpPost.responseStatusCode();
  String response = httpPost.responseBody();
  httpPost.endRequest();
}
```

```
postData("client=" + clientType + "&waterLevel=" + findHeight() + "&moistureLevel1=" + moisture[0] +
"&moistureLevel2=" + moisture[1] + "&moistureLevel3=" + moisture[2] + "&moistureLevel4=" + moisture[3]);
```

    - The Arduino code simply sends a POST request with key parameter pairings similar to the Android code. The parameter pairings should follow the format in the sample function call to ‘postData()’.
- Google Script
    - The Google Script is included in the GitHub repository for this project and should be readable to any user familiar with JavaScript. The main flow of the script works by parsing the POST to the sheet and determining which client it came from. It then calls the relevant function for updating the sheet based on the client request.
    - Security Note
        - One feature that could be implemented in the code is to include a security parameter of the user’s own choosing that must also be sent in order to update the sheet. This would be useful as the posting does not require an API key.