

Google Cloud 与 **Mongoose OS** 构建 **Serverless** 的物联网气象台

Phodal Huang

October 24, 2017

目录

| | |
|--|----|
| 步骤 1: 设置 Google Cloud 项目和 Cloud IoT Core | 4 |
| 步骤 2: Mongoose OS 和 ESP32/ESP8266 | 6 |
| 步骤 3: 编程硬件并设置我们的后端 | 8 |
| 步骤 4: 在 BigQuery 存储数据 | 11 |
| 步骤 5: Firebase 数据库和 Cloud Functions 部署 | 11 |
| 步骤 6: Google Data Studio | 17 |
| 步骤 7: 进一步阅读 | 18 |

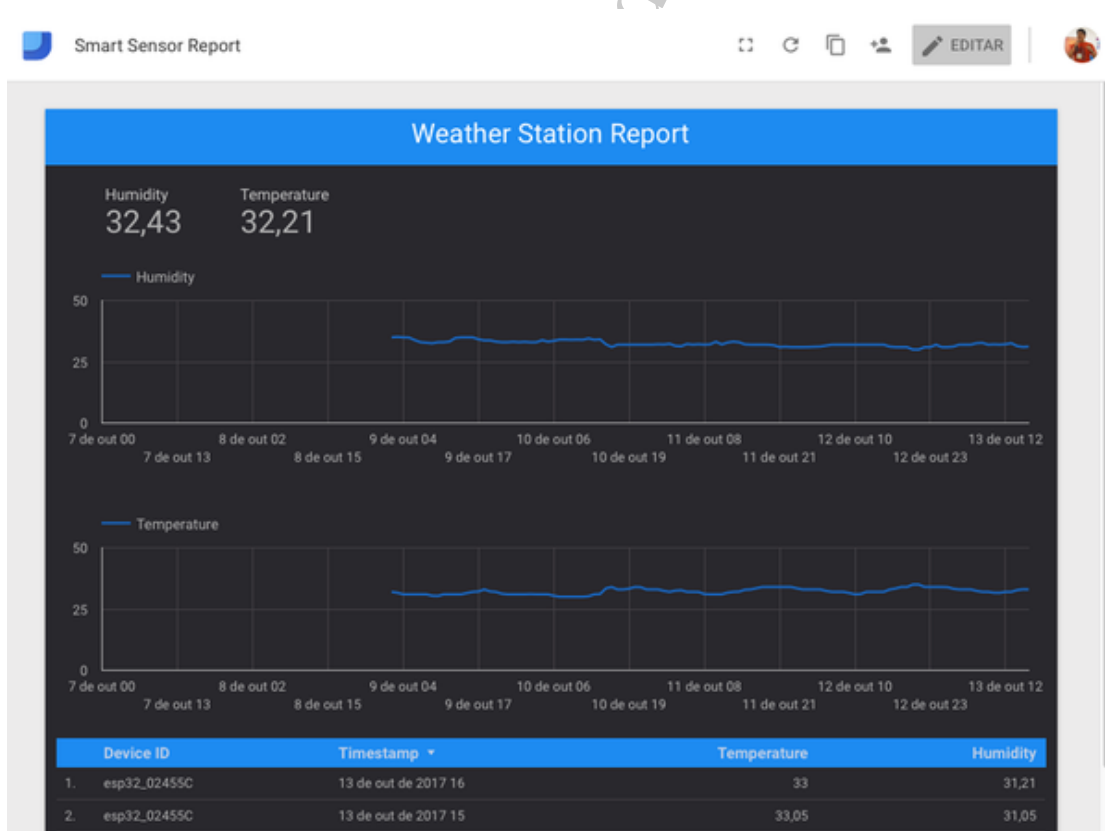
玩点什么: <https://www.wandianshenme.com>

原文链接:<https://www.wandianshenme.com/play/mongoose-os-esp32-google-cloud-iot-core-build>

收集大量数据, 使用托管及 Serverless 架构, 以便您不会在此过程中烧毁。

在本教程中, 我们将使用运行 MongooseOS 的 WiFi 微控制器 ESP8266 构建一个气象站。通过使用基于 Google Cloud IoT Core 之上的 MQTT 协议来安全地发送数据, 然后使用 Firebase Cloud Functions (基于事件的方式) 处理数据, 并将原始数据保存在 BigQuery, 同时在 Firebase 实时数据库中更新设备当前状态。然后可以通过 DataStudio 及 Firebase Hosting 上托管的简单 WebApp 访问数据。

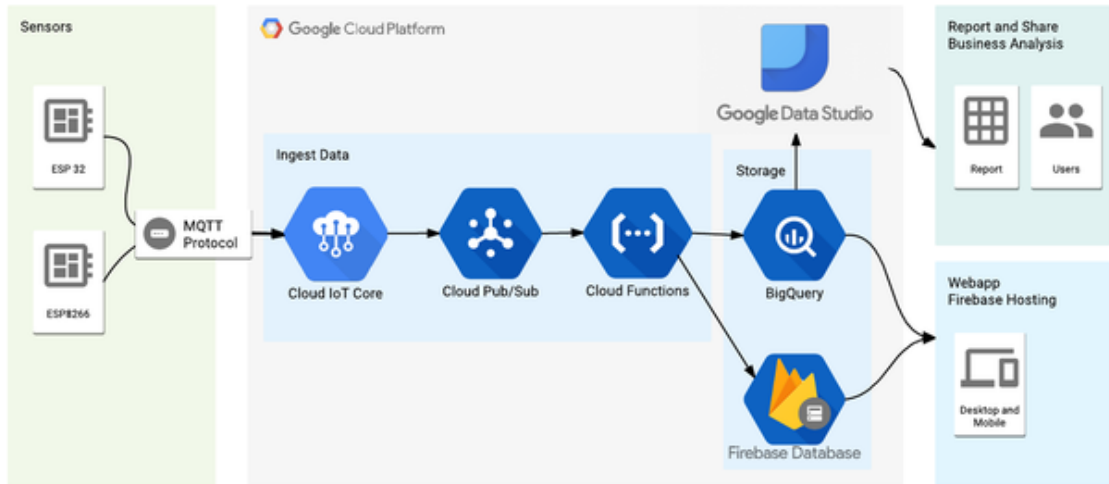
好的, 互联网上已经有很多关于『如何建立气象台』的玩法, 也有很多方法可以做到。这是一个简单的项目, 我将尝试专注于构建端到端解决方案, 从数据收集到数据分析。所有这些都使用托管的 Google Cloud 服务, 来概述如何构建完整的 IoT 解决方案。最后, 您可以构建出一个数据的报告, 并通过网络进行访问。如下图所示:



我们的最后 WebAPP 访问地址:<https://weather-station-iot-170004.firebaseio.com>

DataStudio 报告:<https://datastudio.google.com/reporting/oBow5dnm9bD8sdy1OR1ZQol4Vmc>

架构图如下所示:



为了易于开发，我将使用已经具有 Cloud IoT Core 连接器（connector）的 MongooseOS，并帮助配置证书、WiFi 配置和其他自定义配置的设备的过程。

我们将学习以下的内容：

- 创建一个 Cloud IoT Core 注册的设备。
- 创建 PubSub 主题以接收和发送数据。
- 安装 Mongoose OS 命令行工具 - mos。
- 使用 mos 编程 ESP32/ESP8266。
- 为设备提供证书和 WiFi 配置。
- 设置 BigQuery 和 Firebase 来接收数据。
- 部署 Firebase Cloud 函数来接收数据。
- 在 Firebase Hosting 中部署基本的 WebApp。
- 使用 Data Studio 在 BigQuery 中进行报告。

说了这么多内容，让我们开始□。

步骤 1: 设置 Google Cloud 项目和 Cloud IoT Core

Google 最近推出了公开的 beta 版本 Cloud IoT Core，这是一种托管服务，可以使用通用协议（MQTT 和 HTTP）与您的 IoT 设备进行安全通信，并以简单的方式管理这些设备。基本上，通过这项服务，您可以与许多其他 Google 服务连接，以便于处理、存储和分析设备生成的所有数据。在这里，我们可以看到使用 Cloud IoT Core 的推荐架构的例子。

Cloud IoT Core 有一个设备注册的（registry of devices）的概念，其中我们的项目我们将组合一系列类似的设备并与此注册表关联。要开始使用 Google Cloud，您可以在

Cloud Console Web 界面上完成所有操作，但是命令行工具是一个更强大的工具，它是我选择在此项目中使用的工具。

要使用 `gcloud` 命令行工具，请按照此处的说明进行下载并安装。

[Installing Cloud SDK | Cloud SDK Documentation | Google Cloud Platform](#)

安装完 **SDK** 后，您应该安装测试版工具来访问 **Cloud IoT Core** 命令。此外，您应该在本教程中验证和创建一个要使用的项目，使用您要为此项目的名称，用于替换 `YOUR_PROJECT_NAME`：

```
1 # Install beta components:
2 gcloud components install beta
3 # Authenticate with Google Cloud:
4 gcloud auth login
5 # Create cloud project — choose your unique project name:
6 gcloud projects create YOUR_PROJECT_NAME
7 # Set current project
8 gcloud config set project YOUR_PROJECT_NAME
```

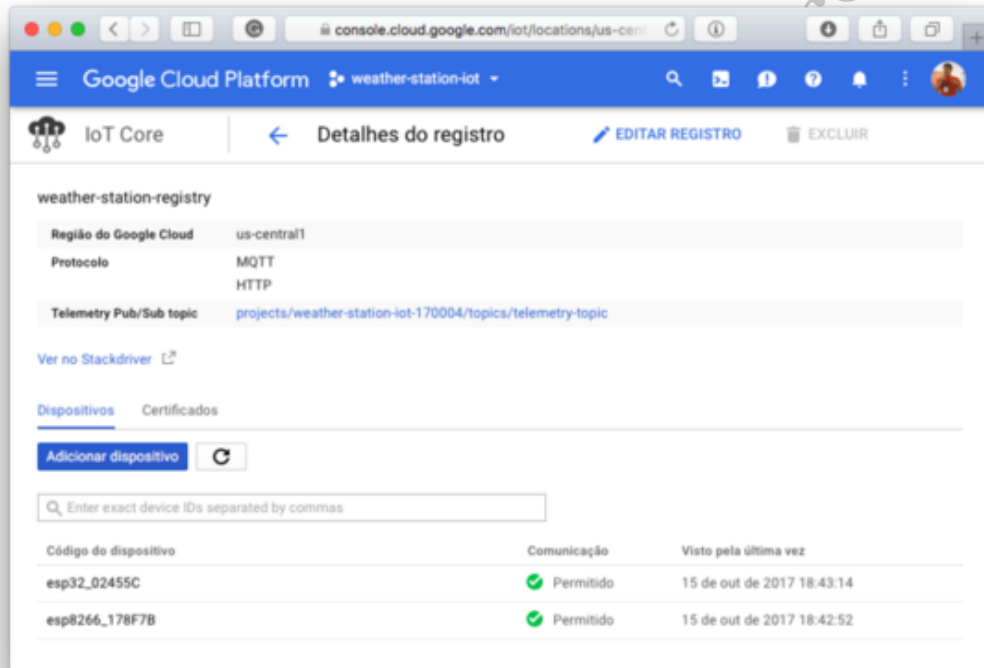
现在，在 **Cloud IoT Core** 方面，您首先应该配置一些与 **Cloud PubSub** 相关的组件，**Cloud PubSub** 是 **Cloud IoT Core** 使用的主要组件之一。在下面的命令中，您将执行以下操作：

1. 授权 **Cloud IoT Core** 在 **PubSub** 上发布消息。
2. 创建一个名为遥测主题 (**telemetry-topic**) 的主题，这些消息将在这被发布。
3. 创建一个名为遥测订阅 (**telemetry-subscription**) 的订阅，我们稍后将使用它们来读取主题中的一些消息。
4. 创建一个名为 **weather-station-registry** 的注册表，我们的设备将被注册为能够连接到 **Cloud IoT Core**。这里我们与创建的主题相关联。

```
1 # Add permissions for IoT Core
2 gcloud projects add-iam-policy-binding YOUR_PROJECT_NAME
   --member=serviceAccount:cloud-iot@system.gserviceaccount.com
   --role=roles/pubsub.publisher
3 # Create PubSub topic for device data:
4 gcloud beta pubsub topics create telemetry-topic
5 # Create PubSub subscription for device data:
6 gcloud beta pubsub subscriptions create --topic telemetry-topic
   telemetry-subscription
```

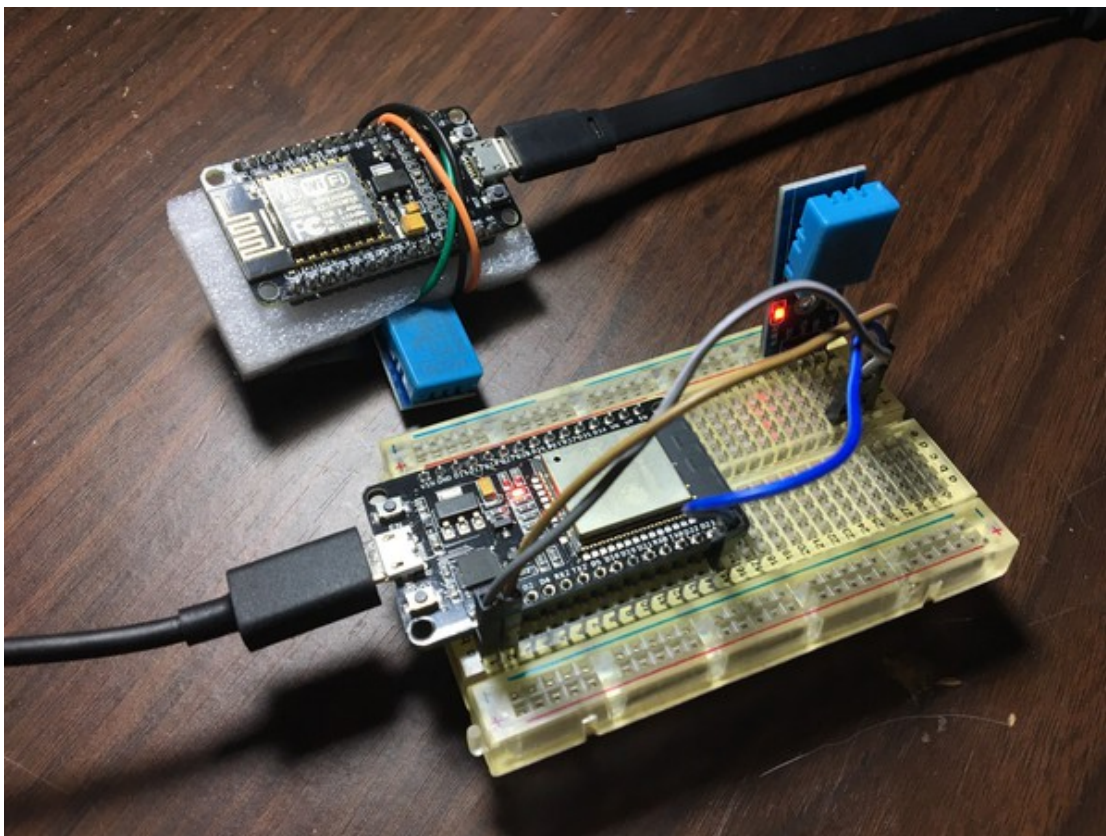
```
7 # Create device registry:
8 gcloud beta iot registries create weather-station-registry --region
   us-central1 --event-pubsub-topic=telemetry-topic
```

如果您访问 **Google Cloud Console**，您可以验证它是否已创建和配置。



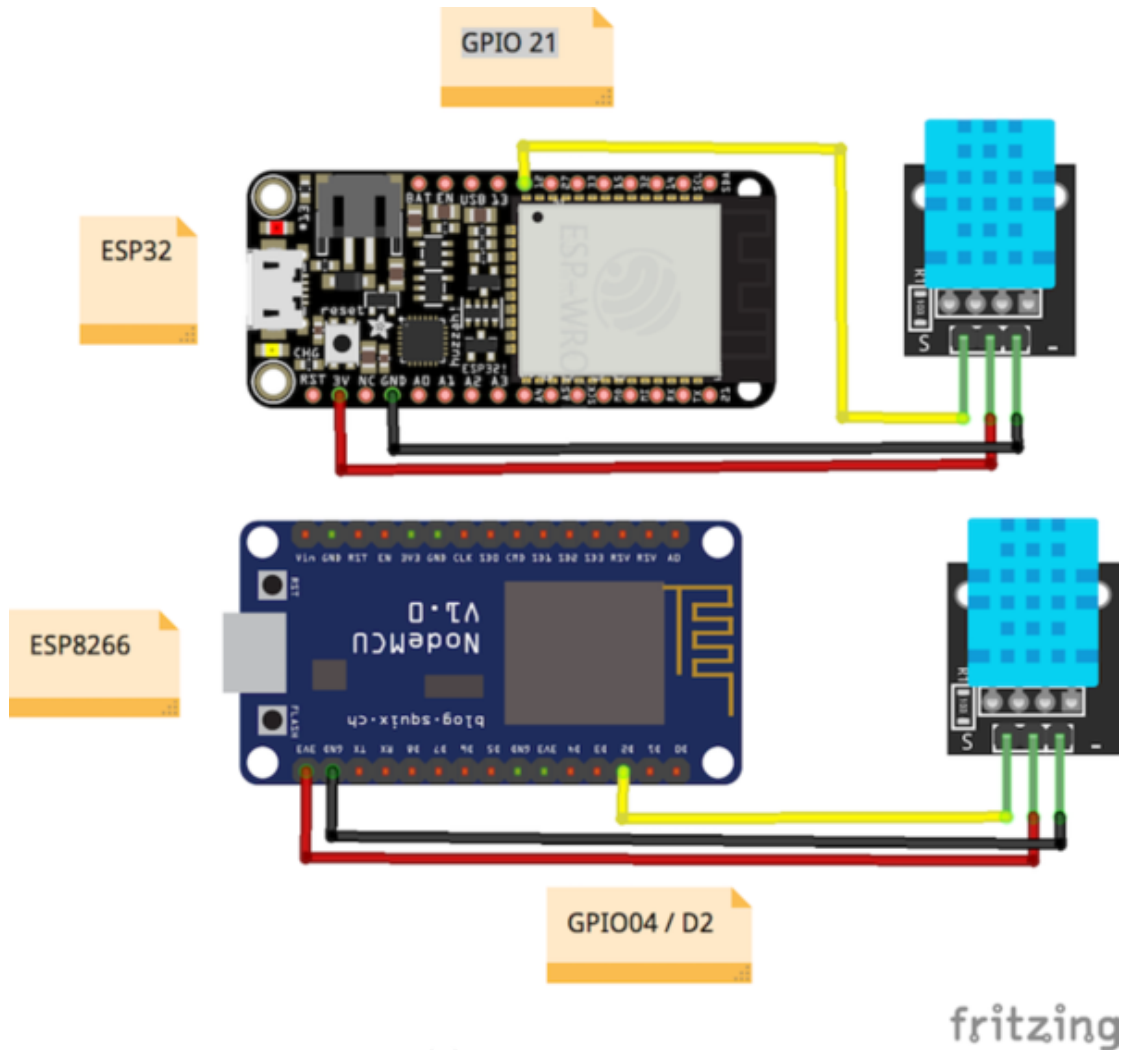
步骤 2: **Mongoose OS** 和 **ESP32/ESP8266**

对于这个项目，我将使用最新的 **ESP32 WiFi** 微控制器。对于那些不了解它的人来说，它是来自 **ExpressIf** 的相关成功的 **ESP8266 WiFi** 微控制器的升级版。但它拥有更多功能，如内置蓝牙低功耗、双核心处理器时钟频率为 **240MHz**、触摸传感器和支持闪存加密，所以没有人可以访问您的代码。一个地狱式的升级。



该项目也适用于 **ESP8266**，因此这里提供的代码和原理图可以在两个微控制器上运行。该项目的电路非常简单，只需将温度 (**DHT**) 传感器连接到 **ESP32/ESP8266**，如下图所示：

玩点什么: <https://www.wm.com>



为了对开发板编程，我们将使用 **MongooseOS**，这是一个操作系统，具有许多很棒的功能，并且可以用于商业胜任。它支持一些微控制器，如 **CC3200**，**ESP32** 和 **ESP8266**。其中一个很酷的功能是使用 **Javascript** 快速原型化嵌入式应用程序的可能性，并且它有一个名为 **mos** 的工具，使得编程、配置和配置在支持的开发板上变得非常简单。

要使用它，我们需要从官方网站下载并安装。按照 <https://mongoose-os.com/docs/quickstart/setup.html> 上的安装说明进行操作。

步骤 3: 编程硬件并设置我们的后端

安装工具后，下载在此链接的 **Github** 仓库上的项目代码：[Build a Weather station using Google Cloud IoT Core and Mongoose OS](#)，以便您可以在设备上构建和部署它。

这个仓库包含了三个子项目：

- **firmware**: 在微控制器上运行的 **MongooseOS** 项目，收集传感器数据并通过 **Cloud**

IoT Core 发送数据

- **functions**: 将部署到 **Firebase** 上的云功能。这里我们有一个函数, 对 **PubSub** 上的新数据做出反应, 然后发送给 **BigQuery** 和 **Firebase** 实时数据库。还有一个功能, 它基本上可以说是一个 **HTTP** 端点, 可以在最近 7 天的数据中查询 **BigQuery**, 以供我们的 **WebApp** 使用。
- **public**: 一个将部署在 **Firebase** 托管上的简单 **WebApp**, 它会查询我们的数据库以显示我们的传感器数据。

这里有一些 **firmware** 项目的描述:

- **fs**: 这里我们有我们的 **Javascript** 代码, 其中包含收集数据的所有逻辑, 并在固定的时间间隔内通过 **MQTT** 发送。
- **src**: 我们的原生 **C** 代码, 基本用于启动 **Google Cloud** 库, 因此它会自动将我们的项目配置为与 **Google MQTT** 服务器连接。
- **mos.yml** 和 **mos_esp8266.yml**: 我们的项目配置, 这里我们声明我们的项目依赖关系, 在这种情况下, **GCP** 库, **DHT** 传感器库和 **mJS** 库, 最后一个添加了对 **Javascript** 嵌入式的支持。这里我们声明一个名为 **app.dht** 的自定义配置变量, 这样我们可以通过更改这个配置来改变 **DHT** 引脚, 这个配置可以在这个文件上修改或通过 **mos** 工具。此外, 这种配置在适用于微控制器之间的变化, 增加了对相同代码的两个微控制器的支持。

要对硬件编程, 请进入 **firmware** 文件夹, 并运行以下说明来烧录固件, 在 **Cloud IoT Core** 上配置 **WiFi** 并配置设备:

- 根据所选的硬件, 运行 `mos build --arch esp32` 或者 `mos build --arch esp8266`。此命令来构建我们硬件的固件。
- 通过 **USB** 连接的硬件运行 `mos flash` 来刷新固件。
- 运行 `mos wifi.your_ssid your_pass` 在您的设备上配置 **WiFi**。
- 运行以下命令在 **Cloud IoT Core** 上注册此设备。该命令生成用于通信的公钥和私钥, 将私钥放在设备上, 将公钥发送到 **Cloud IoT Core** 并注册设备, 从 **ESP** 获取 **deviceId**。感谢 **MongooseOS**。

```
1 mos gcp-iot-setup --gcp-project YOUR_PROJECT_NAME --gcp-region us-central1
  --gcp-registry YOUR_REGISTRY
```

就这样, 您的设备将开始向 **Cloud IoT Core** 发送数据。这些项目配置为每分钟发送数据, 您可以在 `fs/init.js` 文件中进行更改, 也可以创建自定义配置变量来更改时

间。我会把这作为一个 **homework**。您可以使用 `mos console` 工具查看设备上发生的情况。您会看到它尝试与 `mqtt.googleapis.com` 连接。

```

1 $ mos console
2 Using port /dev/cu.SLAB_USBtoUART
3 [Oct 15 18:17:47.230] pm open,type:2 0
4 [Oct 15 18:17:47.234] mgos_snmp_ev          SNMP reply from 192.99.2.8: time
   1508102268.124028, local 15.317275, delta 1508102252.806753
5 [Oct 15 18:17:47.448] mgos_mqtt_ev          MQTT CONNACK 4
6 [Oct 15 18:17:47.455] mgos_mqtt_ev          MQTT Disconnect
7 [Oct 15 18:17:47.463] mqtt_global_reconnec MQTT connecting after 2017 ms
8 [Oct 15 18:17:48.167] Info:
   {"hum":34,"temp":30,"free_ram":35.593750,"total_ram":51.218750}
9 [Oct 15 18:17:49.487] mgos_mqtt_global_con MQTT connecting to
   mqtt.googleapis.com:8883

```

要查看 **PubSub** 上的数据，您可以使用 `gcloud` 命令来查询我们创建的订阅：

```

1 $ gcloud beta pubsub subscriptions pull --auto-ack telemetry-subscription
2 ┌
3 │                                DATA                                │ MESSAGE_ID
4 │                                │                                │
5 │                                │                                │
6 │                                │                                │
7 │                                │                                │
8 │                                │                                │
9 │                                │                                │
10 │                                │                                │
11 │                                │                                │
12 │                                │                                │
13 │                                │                                │
14 │                                │                                │
15 │                                │                                │
16 │                                │                                │
17 │                                │                                │
18 │                                │                                │
19 │                                │                                │
20 │                                │                                │
21 │                                │                                │
22 │                                │                                │
23 │                                │                                │
24 │                                │                                │
25 │                                │                                │
26 │                                │                                │
27 │                                │                                │
28 │                                │                                │
29 │                                │                                │
30 │                                │                                │
31 │                                │                                │
32 │                                │                                │
33 │                                │                                │
34 │                                │                                │
35 │                                │                                │
36 │                                │                                │
37 │                                │                                │
38 │                                │                                │
39 │                                │                                │
40 │                                │                                │
41 │                                │                                │
42 │                                │                                │
43 │                                │                                │
44 │                                │                                │
45 │                                │                                │
46 │                                │                                │
47 │                                │                                │
48 │                                │                                │
49 │                                │                                │
50 │                                │                                │
51 │                                │                                │
52 │                                │                                │
53 │                                │                                │
54 │                                │                                │
55 │                                │                                │
56 │                                │                                │
57 │                                │                                │
58 │                                │                                │
59 │                                │                                │
60 │                                │                                │
61 │                                │                                │
62 │                                │                                │
63 │                                │                                │
64 │                                │                                │
65 │                                │                                │
66 │                                │                                │
67 │                                │                                │
68 │                                │                                │
69 │                                │                                │
70 │                                │                                │
71 │                                │                                │
72 │                                │                                │
73 │                                │                                │
74 │                                │                                │
75 │                                │                                │
76 │                                │                                │
77 │                                │                                │
78 │                                │                                │
79 │                                │                                │
80 │                                │                                │
81 │                                │                                │
82 │                                │                                │
83 │                                │                                │
84 │                                │                                │
85 │                                │                                │
86 │                                │                                │
87 │                                │                                │
88 │                                │                                │
89 │                                │                                │
90 │                                │                                │
91 │                                │                                │
92 │                                │                                │
93 │                                │                                │
94 │                                │                                │
95 │                                │                                │
96 │                                │                                │
97 │                                │                                │
98 │                                │                                │
99 │                                │                                │
100 │                                │                                │

```

如果您在控制台上看到数据，您可以开始庆祝，我们走在正确的路上□□

步骤 4: 在 **BigQuery** 存储数据

直接从官方网站了解 **BigQuery** 的定义:

BigQuery 是 **Google** 的低成本、完全可管理的 **PB** 级可扩展数据存储服务。**BigQuery** 是独立的, 没有要管理的基础设施, 并且随着数据扩展, 您不需要数据库管理员。

在这里, 我们将使用它来存储我们收集的所有传感器数据, 以运行一些查询, 并在以后使用 **Data Studio** 构建报告。我们将创建一个数据集和一个表存储我们的数据。要执行此操作, 请打开 **BigQuery Web UI**, 然后按照以下说明进行操作:

- 单击向下的箭头图标, 然后单击“Create new dataset”。
- 将数据集命名为“weather_station_iot”。
- 使用以下字段和类型创建一个表“raw_data”表:

Create Table

Source Data Create from source Create empty table

Destination Table

Table name weather_station_iot . raw_data ?

Table type Native table ?

Schema

| Name | Type | Mode |
|-----------|-----------|----------|
| deviceId | STRING | REQUIRED |
| timestamp | TIMESTAMP | REQUIRED |
| temp | FLOAT | REQUIRED |
| humidity | FLOAT | REQUIRED |

Add Field [Edit as Text](#)

Options

Partitioning Day

Create Table

步骤 5: **Firestore** 数据库和 **Cloud Functions** 部署

现在为了在 **BigQuery** 中插入数据, 我们将使用 **Firestore Cloud Functions**, 可以配置为基于许多不同的触发器和事件执行。其中一个触发器是在 **PubSub** 主题中插入的新

数据，因此我们将监听与我们的设备注册表关联的主题，并收集到达的每个数据，我们执行一个在 **BigQuery** 中存储数据的功能，并在 **Firebase** 实时数据库里维护最后的设备数据。

Firebase 实时数据库是一种非常有用的维护实时数据的技术，可在所有连接的客户端之间实现自由和自动同步。即使 **Google** 也推荐它来保持物联网设备的实时状态，见：<https://cloud.google.com/solutions/iot-overview>，就像我们在这里看到的一样。

我们的函数的代码可以在上面看到，它基本上对 **PubSub** 事件做出反应并插入 **BigQuery**，同时进行更新。

Firebase 命令行工具需要 **Node.JS** 和 **npm**，您可以按照 <https://nodejs.org/> 上的说明进行安装，安装 **Node.js** 也将安装 **npm**。

一旦安装了 **Node** 和 **NPM**，运行以下命令来安装 **Firebase CLI**。

```
1 npm install -g firebase-tools
```

现在要使用我们的项目配置 **firebase** 并部署函数，在项目根文件夹中，按照以上说明进行操作：

- 运行 `firebase login`，与 **Google** 进行身份验证并设置命令行工具
- 运行 `firebase init` 将本地项目与您的 **Firebase** 项目关联。
- 运行上面的代码设置一些环境变量，指向我们的 **BigQuery** 数据集和表。

```
1 firebase functions:config:set
2 bigquery.datasetname="weather_station_iot"
3 bigquery.tablename="raw_data"
```

- 最后运行 `firebase deploy` 以在 **public** 文件夹中部署“函数”和“Webapp”。

```
alvaroviebrantz@MacBook-Pro: ~/Documents/Desenvolvimento/IoT/WeatherSta...
# alvaroviebrantz @ MacBook-Pro in ~/Documents/Desenvolvimento/IoT/WeatherStation on git:master x [20:01:15]
$ firebase deploy

=== Deploying to 'weather-station-iot-170004'...

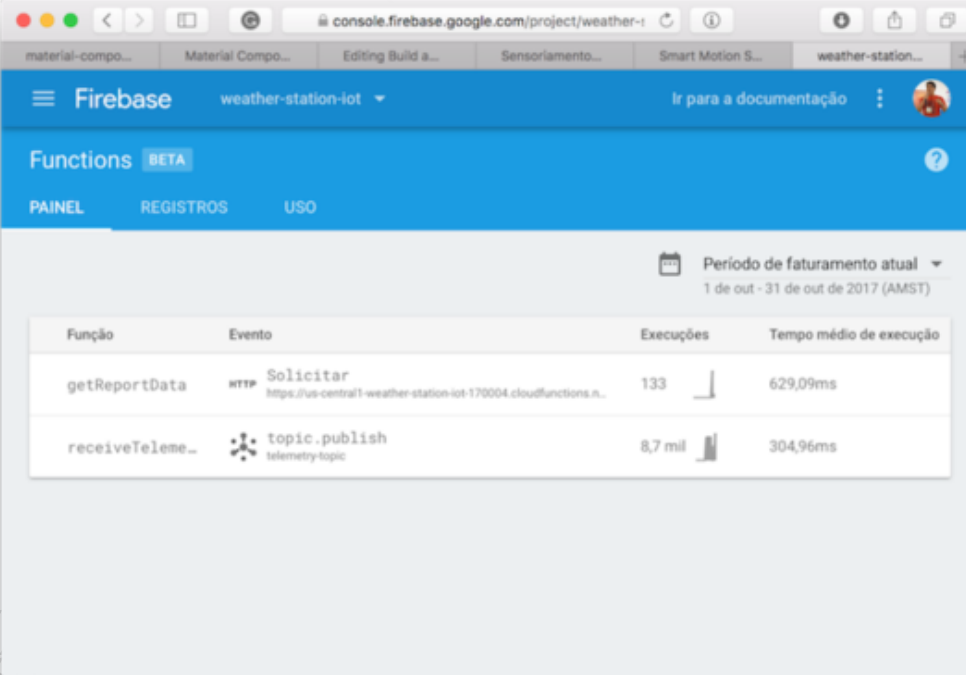
i deploying database, functions, hosting
✓ database: rules ready to deploy.
i functions: ensuring necessary APIs are enabled...
i runtimeconfig: ensuring necessary APIs are enabled...
✓ runtimeconfig: all necessary APIs are enabled
✓ functions: all necessary APIs are enabled
i functions: preparing functions directory for uploading...
i functions: packaged functions (21.16 KB) for uploading
✓ functions: functions folder uploaded successfully
i hosting: preparing public directory for upload...
✓ hosting: 2 files uploaded successfully
i starting release process (may take several minutes)...
i functions: updating function receiveTelemetry...
i functions: updating function getReportData...
✓ functions[receiveTelemetry]: Successful update operation.
✓ functions[getReportData]: Successful update operation.
✓ functions: all functions deployed successfully!

✓ Deploy complete!

Project Console: https://console.firebase.google.com/project/weather-station-iot-170004/overview
Hosting URL: https://weather-station-iot-170004.firebaseio.com
Function URL (getReportData): https://us-central1-weather-station-iot-170004.cloudfunctions.net/getReportData

# alvaroviebrantz @ MacBook-Pro in ~/Documents/Desenvolvimento/IoT/WeatherStation on git:master x [21:52:18]
$
```

通过部署函数，您可以设置所有设置来摄取设备发送的遥测数据，并存储在两个存储解决方案中。您可以在 **Firebase** 控制台上看到所有部署的资源。



| Função | Evento | Execuções | Tempo médio de execução |
|------------------|--|-----------|-------------------------|
| getReportData | HTTP Solicitar https://us-central1-weather-station-iot-170004.cloudfunctions.n... | 133 | 629,09ms |
| receiveTeleme... | topic.publish telemetry-topic | 8,7 mil | 304,96ms |

如下是上述功能的代码:

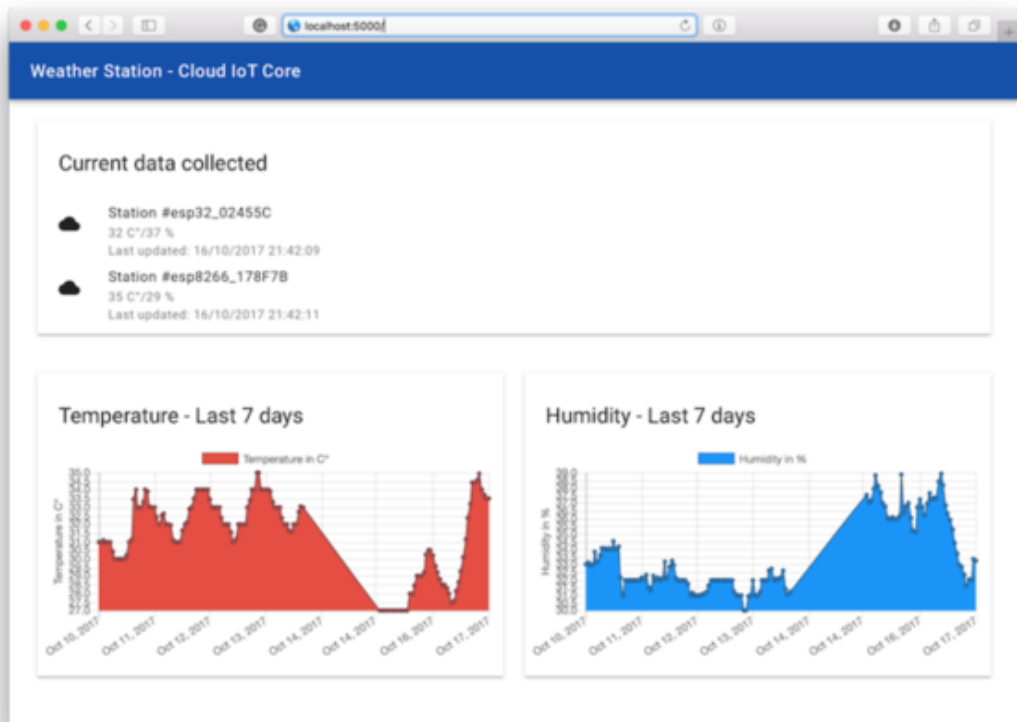
```
1 const functions = require('firebase-functions');
2 const admin = require('firebase-admin');
3 const bigquery = require('@google-cloud/bigquery')();
4 const cors = require('cors')({ origin: true });
5
6 admin.initializeApp(functions.config().firebase);
7
8 const db = admin.database();
9
10 /**
11  * Receive data from pubsub, then
12  * Write telemetry raw data to bigquery
13  * Maintain last data on firebase realtime database
14  */
15 exports.receiveTelemetry = functions.pubsub
16   .topic('telemetry-topic')
17   .onPublish(event => {
18     const attributes = event.data.attributes;
19     const message = event.data.json;
20
21     const deviceId = attributes['deviceId'];
22
23     const data = {
24       humidity: message.hum,
25       temp: message.temp,
26       deviceId: deviceId,
27       timestamp: event.timestamp
28     };
29
30     if (
31       message.hum < 0 ||
32       message.hum > 100 ||
33       message.temp > 100 ||
34       message.temp < -50
35     ) {
```

```
36     // Validate and do nothing
37 return;    }
38
39     return Promise.all([
40         insertIntoBigquery(data),
41         updateCurrentDataFirebase(data)
42     ]);
43 });
44
45 /**
46  * Maintain last status in firebase
47 */
48 function updateCurrentDataFirebase(data) {
49     return db.ref(`/devices/${data.deviceId}`).set({
50         humidity: data.humidity,
51         temp: data.temp,
52         lastTimestamp: data.timestamp
53     });
54 }
55
56 /**
57  * Store all the raw data in bigquery
58 */
59 function insertIntoBigquery(data) {
60     // TODO: Make sure you set the `bigquery.datasetname` Google Cloud
        environment variable.
61     const dataset = bigquery.dataset(functions.config().bigquery.datasetname);
62     // TODO: Make sure you set the `bigquery.tablename` Google Cloud
        environment variable.
63     const table = dataset.table(functions.config().bigquery.tablename);
64
65     return table.insert(data);
66 }
67
68 /**
69  * Query bigquery with the last 7 days of data
```

```
70 * HTTPS endpoint to be used by the webapp
71 */exports.getReportData = functions.https.onRequest((req, res) => {
72   const table = `weather-station-iot-170004.weather_station_iot.raw_data`;
73
74   const query = `
75     SELECT
76       TIMESTAMP_TRUNC(data.timestamp, HOUR, 'America/Cuiaba') data_hora,
77       avg(data.temp) as avg_temp,
78       avg(data.humidity) as avg_hum,
79       min(data.temp) as min_temp,
80       max(data.temp) as max_temp,
81       min(data.humidity) as min_hum,
82       max(data.humidity) as max_hum,
83       count(*) as data_points
84   FROM ${table} data
85   WHERE data.timestamp between timestamp_sub(current_timestamp, INTERVAL
86     7 DAY) and current_timestamp()
87   group by data_hora
88   order by data_hora
89 `;
90   return bigquery
91     .query({
92       query: query,
93       useLegacySql: false
94     })
95     .then(result => {
96       const rows = result[0];
97
98       cors(req, res, () => {
99         res.json(rows);
100       });
101     });
102 });
```

`firebase-tools` 还有一个内置服务器，您可以在刚刚运行 `firebase` 服务的项目

文件夹上启动它，默认情况下将启动端口 5000 上的 Web 服务器。

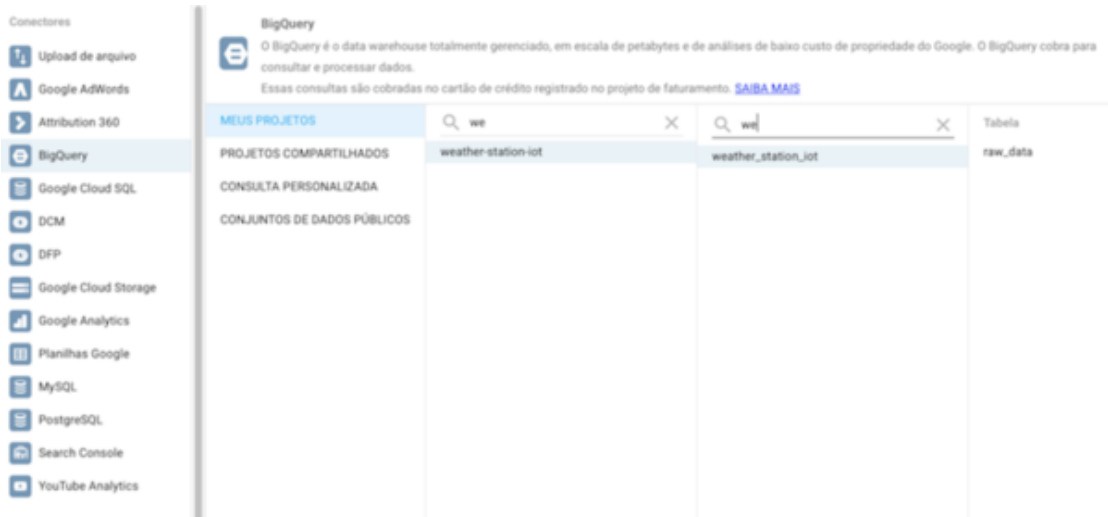


webapp 可以直接在公共目录中看到，逻辑在 public/app.js，前端在 public/index.html。这是非常基本的，只有 Javascript，Web Material Component 和 Chart.JS 的图表。

如果所有这些都正确设置，那么您可以再次庆祝，因为您开发了 IoT 的端到端解决方案，而无需触摸高级服务器设置。

步骤 6: Google Data Studio

Data Studio 是一个非常直观的工具，我不会在这里进行探索，所以这个教程变得不那么广泛，但是让你知道，Data Studio 有一个 BigQuery 连接器，所以只需导入你的表格，并使用这个真棒工具提供的不同的可视化。转到 datastudio.google.com 体验。



步骤 7: 进一步阅读

这是本教程的所有内容，希望对您 Google Cloud IoT Core 感兴趣，这是一个非常棒的服务，您可以使用它来实现强大的功能。该帖子比我玩法的要长一些，但我相信 Google Cloud Platform 上有很多工具。

该项目的代码可以在我的 Github 上找到: <https://github.com/alvarowolfx/weather-station-gcp-mongoose-os>，以下是一些有趣的链接及内容：

- <https://cloudplatform.googleblog.com/2017/09/announcing-Cloud-IoT-Core-public-beta.html>
- <http://mongoose-os.com/gcp>
- <https://cloud.google.com/iot/docs/quickstart>
- https://mongoose-os.com/docs/libraries/cloud_integrations/gcp.html
- <https://www.adafruit.com/product/3606>
- <https://github.com/alvarowolfx/weather-station-gcp-mongoose-os>

原文链接: <https://medium.com/google-cloud/build-a-weather-station-using-google-cloud-iot-core->

原文链接: <https://www.wandianshenme.com/play/mongoose-os-esp32-google-cloud-iot-core-build->