4/22/2020

ESP8266 IoT Microcontroller Temperature Recorder

17091837

1 CONTENTS

2 Introduction

This document was created to show how to design an IoT system that enables data to be sent from a DHT22 temperature/humidity sensor and send this information to Google Cloud. All code or commands used within the Iot device and Google Cloud will be shown here along with an accompanying explanation of it. Any references or guides to external information used in this document will also be highlighted here too. The following website was predominantly used to conduct this design:

https://medium.com/google-cloud/build-a-weather-station-using-google-cloud-iot-core-and-mongooseos-7a78b69822c5

3 System Requirements

Here is a list of the functional requirements used for this design. A block diagram of the entire system displaying both the hardware and software used, alongside their connection to each other, can be found in Figure 1.

3.1 RQ1 - IoT Sensor Device:

The IoT device to use will need to be able to collect all readings of its surrounding climate, such as the temperature and humidity. For this design, the ESP8266 Wi-fi microchip armed with a full TCP/IP stack and microcontroller (Espressif Systems, 2019) was chosen. This will need to be used in combination with a DHT22, a low-cost humidity sensor and thermistor (https://learn.adafruit.com, 2012).

3.2 RQ2 - IoT Operating System

An operating system will need to be used on the IoT device which will be responsible for: collecting sensor data, sending data to Cloud IoT core, process of provisioning devices with certificates, WiFi manipulation and numerous other configurations. For this design, MongooseOS will be used, which is an internet of things firmware development framework which provides support for the ESP8266 microchip (Williams, 2017).

3.3 RQ3 - IoT Cloud Integration

All data collected by the IoT device will be sent to the cloud using the MQTT protocol. MQTT can be used with the gcloud suite to build a communication bridge between IoT devices and Cloud IoT Core (Google, 2020). Cloud IoT Core is a fully managed service that is designed to securely connect, manage and ingest data from dispersed devices (Google, 2020). From

Cloud IoT Core, we can then begin to process, store and analyse the data generated by our IoT device. Google's cloud functions will be used to compute the resources used within gcloud (Google, 2020).

3.4 RQ4 - Database

There will be two databases to store all the data collected by our IoT device. From these, we can run any functions or calculations from the data gathered. The first of which is Google's Big Query, a serverless high scalable cloud data warehouse that uses fast in-memory analysis to allow for the examination and querying of large and complex datasets (Google, 2020). The second is Firebase Realtime Database, which is used to store and sync data to a NoSQL cloud database (Google, 2020). This allows for data to be synced across all the components our system design will incorporate.

3.5 RQ5 - Message Subscription

We will need to integrate messaging in real-time between all devices and applications involved. For this, we will be using Google's Cloud Pub/Sub, which provides a global messaging send & receive service (Google, 2020).

3.6 RQ6 – Triggers

Data will need to be inserted into our database in real-time whenever any data has been recorded by our device. To do this, Firebase Cloud functions will be configured to activate a trigger or event when this occurs. These triggers will insert new data into a PubSub Topic and subsequently store the data collected into Google's BigQuery.

3.7 RQ8 – Data Analytics & Reporting

Reports can be generated from BigQuery where our database is stored using Data Studio. This is a platform that can provide a selection of interactive dashboards to work with and display any data or analytics that is provided to it (Google, 2020).

4 Instructions

4.1 Project Setup

First, we will need to ensure that we have access to Google Cloud services. This can be accessed via their website or installing the SDK on our machine. From here we can begin to write the commands necessary to authenticate ourselves and create a new project. Here we have named our project "myweatherstation7", but this can be named differently.

```
# Install beta components:
gcloud components install beta
# Authenticate with Google Cloud:
gcloud auth login
# Create cloud project — choose your unique project name:
gcloud projects create myweatherstation7
# Set current project
gcloud config set project myweatherstation7
```

(Viebrantz, 2017)

4.2 PubSub & IoT Registry

Permissions will initially be required to authorize Cloud IoT Core to publish messages on PubSub. Once completed, there will need to be a Topic created from PubSub where Cloud IoT core can publish messages. A PubSub subscription will also need to be created to read messages from. As a final step, a registry named will need to be created where the IoT device can be registered to connect to Cloud IoT Core. In this case, we have called this “weather-station-registry“.

```
# Add permissions for IoT Core
gcloud projects add-iam-policy-binding myweatherstation7 --
member=serviceAccount:cloud-iot@system.gserviceaccount.com --
role=roles/pubsub.publisher
# Create PubSub topic for device data:
gcloud beta pubsub topics create telemetry-topic
# Create PubSub subscription for device data:
gcloud beta pubsub subscriptions create --topic telemetry-topic telemetry-
subscription
# Create device registry:
gcloud beta iot registries create weather-station-registry --region us-
central1 --state-pubsub-topic=telemetry-topic
```

(Viebrantz, 2017)

4.3 MongooseOS

Next, we will need to install MongooseOS. As previously mentioned, this will be the firmware running on the controller that will collect sensor data from the DHT22 and send to the Cloud IoT Core. The following link can be used to install this:

https://mongoose-os.com/docs/quickstart/setup.html

4.4 Setup Backend of Hardware

We will then download the following GitHub repository by following the installation instructions to build and deploy it to our device:

https://github.com/alvarowolfx/weather-station-gcp-mongoose-os

This repository contains:

- **Firmware**: The Mongoose OS to run on the microcontroller
- **Functions**: Cloud IoT functions to deploy on Firebase
- **Public**: A WebApp that shows the data collected from the device sensor

The firmware of which will contain:

1) **fs**: JavaScript code containing all logic to collect data and send through MQTT
2) **src**: Native C code initiating the Google Cloud library to configure our project with MQTT
3) **mos.yml and mos_esp8266.yml:** Project configuration declaring project dependencies

4.5 Program Hardware

The hardware can be configured using the following instructions (WiFi password should be adjusted accordingly) which will flash the firmware, organise the WiFi and provision the device on Cloud IoT Core:

```
# Build firmware of hardware
mos build --arch esp8266
# Flash firmware
mos flash
# Configure WiFi on device
mos wifi your_ssid your_pass
# Register device to Cloud IoT & generate secure keys for communication
mos gcp-iot-setup --gcp-project myweatherstation7 --gcp-region us-central1
--gcp-registry weather-station-registry
```

(Viebrantz, 2017)

4.6 BigQuery

All the data collected from our device will now be stored using Google's "BigQuery" where we can run queries and build reports from. Achieved with the following instructions:

1. Open the BigQuery Web User Interface

2. Name your dataset “weather_station_iot” (Can be named differently)
3. Create a new table called “raw_data” using the fields and types from Figure 2

4.7 Firebase

Firebase cloud functions will can now be used to insert data into BigQuery & Firebase Realtime Database. A trigger will be created for our PubSub topic which will listen to each piece of data recorded and insert that into our databases. The Firebase Command Line Tools will need Node.JS and npm to be installed to work, which can be done at https://nodejs.org/. After these are installed, we can use the following command to install Firebase CLI:

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

The project will now be configured to use with our project by using the next set of instructions.

```
# Authenticate with Google
firebase login
# Associate local project with Firebase project
firebase init
# Point variable to BigQuery dataset & table
firebase functions:config:set
bigquery.datasetname="weather_station_iot"
bigquery.tablename="raw_data”
# Deploy Functions and Webapp onto public flder
firebase deploy
```

(Viebrantz, 2017)

All resources and data sent by the device should now be displayed on the Firebase Console.

4.8 Server Setup

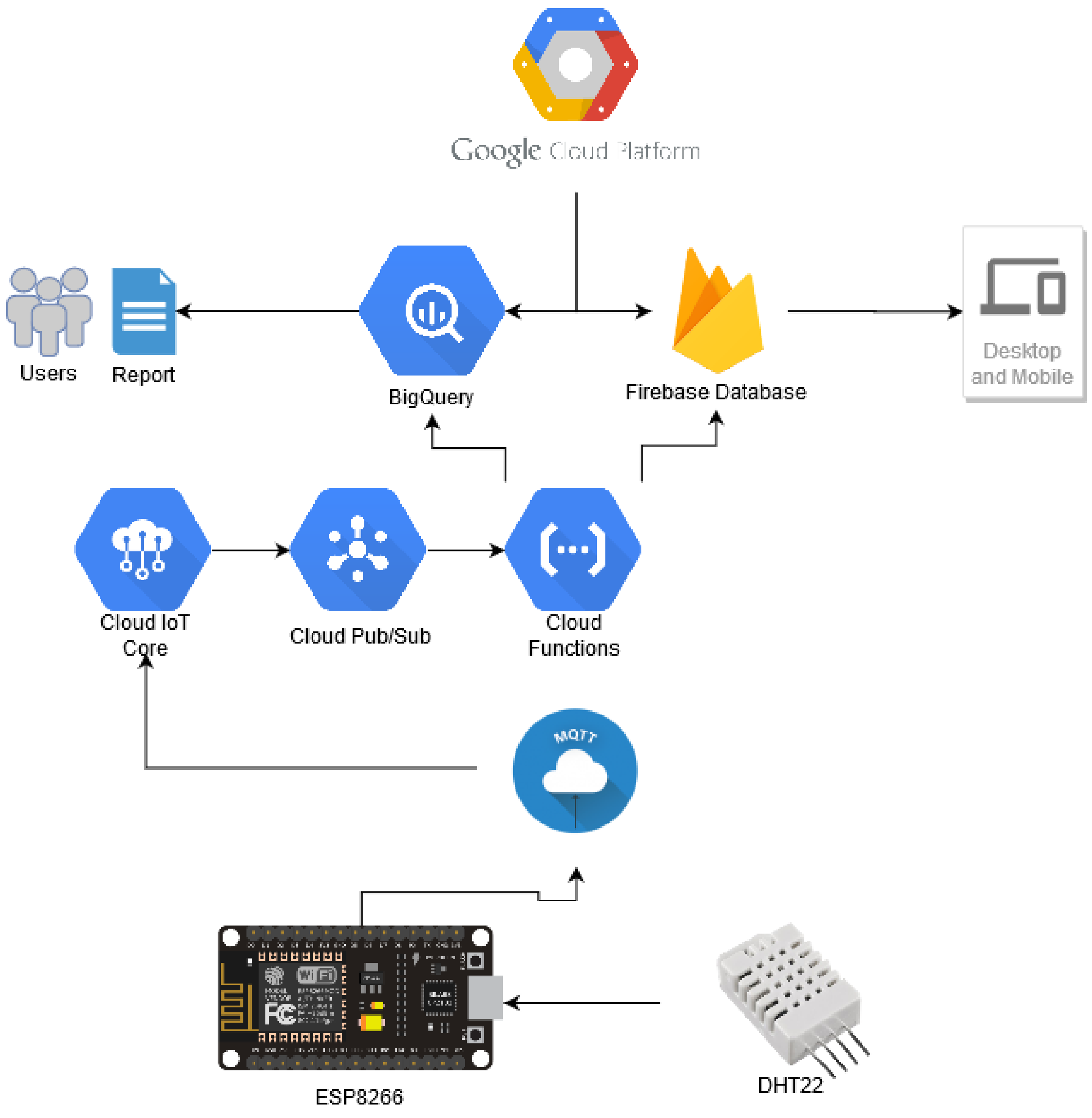
A server can be started on the project by running “firebase serve” as shown in Figure 3. This will start a web server on port 5000 by default. The web app is displayed on the public directory, with the logic being displayed at public/app.js and the frontend at public/index.html.

5 Conclusion

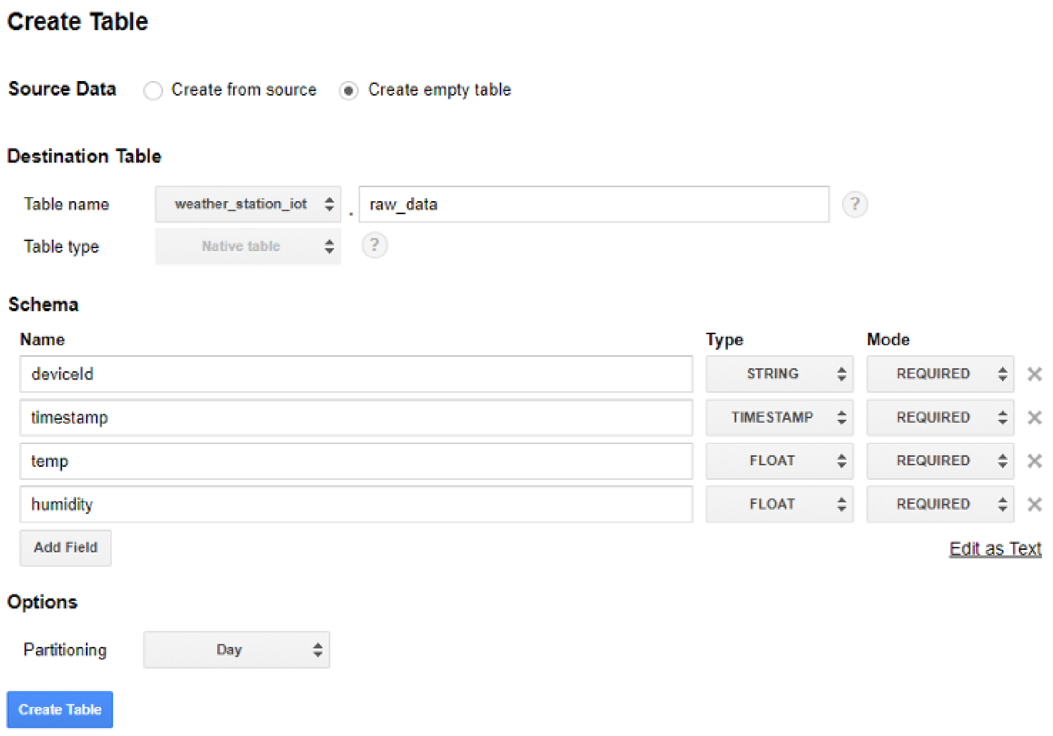
Following the intrusions laid out in this report, we now have a fully working IoT weather recording device that sends weather data to Google Cloud from an ESP28266 microchip, ready for the data to be analysed, build reports and collect data from.

6 Appendix

6.1 Figure 1



6.2 Figure 2



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

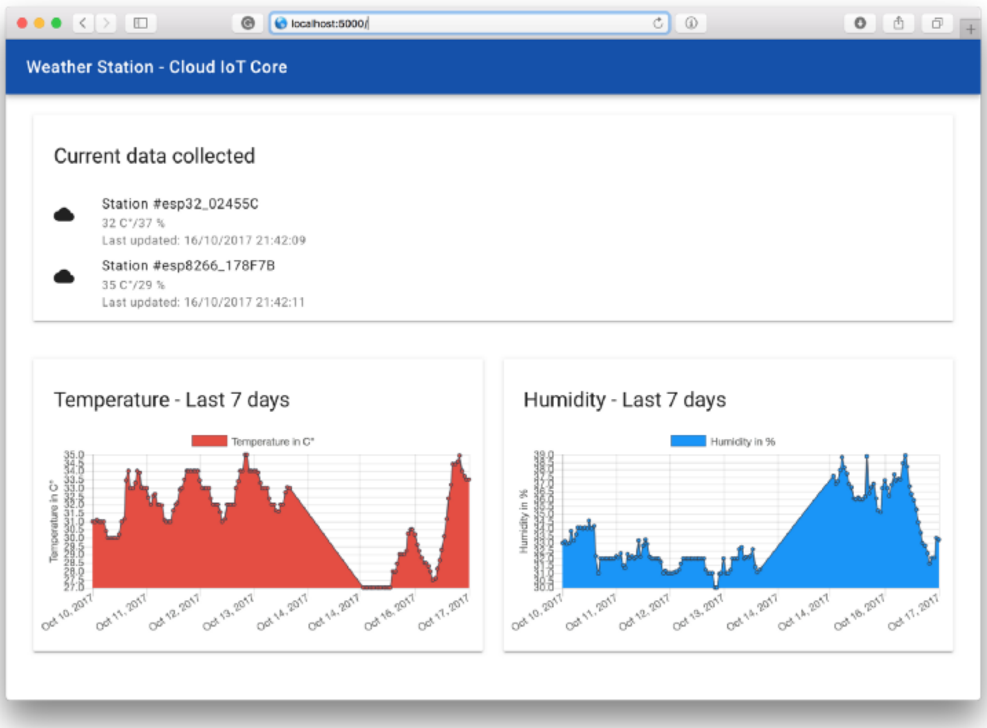
Options

Partitioning: Day

Create Table

(Viebrantz, 2017)

6.3 Figure 3



localhost:5000/

Weather Station - Cloud IoT Core

Current data collected

Station #esp32_02455C
32 C°/37 %
Last updated: 16/10/2017 21:42:09

Station #esp8266_178F7B
35 C°/29 %
Last updated: 16/10/2017 21:42:11

Temperature - Last 7 days

Humidity - Last 7 days

(Viebrantz, 2017)

7 References

alvarowolfx, 2017. *https://gist.github.com.* [Online]
Available at: https://gist.github.com/alvarowolfx/3faa2079bfa6c30beefcdb2db0b670ce#file-init-js
[Accessed 12 April 2020].

Espressif Systems, 2019. *https://www.espressif.com.* [Online]
Available at: https://www.espressif.com/en/products/hardware/esp8266ex/overview
[Accessed 18 April 2020].

Google, 2020. *https://cloud.google.com.* [Online]
Available at: https://cloud.google.com/iot/docs/how-tos/mqtt-bridge
[Accessed 16 April 2020].

Google, 2020. *https://cloud.google.com.* [Online]
Available at: https://cloud.google.com/iot-core/
[Accessed 18 April 2020].

Google, 2020. *https://cloud.google.com.* [Online]
Available at: https://cloud.google.com/pubsub/docs
[Accessed 18 April 2020].

Google, 2020. *https://cloud.google.com.* [Online]
Available at: https://cloud.google.com/functions/
[Accessed 18 April 2020].

Google, 2020. *https://cloud.google.com/bigquery/.* [Online]
Available at: https://cloud.google.com
[Accessed 20 April 2020].

Google, 2020. *https://firebase.google.com.* [Online]
Available at: https://firebase.google.com/docs/database/
[Accessed 20 April 2020].

Google, 2020. *https://marketingplatform.google.com.* [Online]
Available at: https://marketingplatform.google.com/about/data-studio/
[Accessed 21 April 2020].

https://learn.adafruit.com, 2012. *https://learn.adafruit.com.* [Online]
Available at: https://learn.adafruit.com/dht
[Accessed 18 April 2020].

Viebrantz, A., 2017. *https://medium.com.* [Online]
Available at: https://medium.com/google-cloud/build-a-weather-station-using-google-cloud-iot-core-and-mongooseos-7a78b69822c5
[Accessed 12 04 2020].

Williams, A., 2017. *https://hackaday.com.* [Online]
Available at: https://hackaday.com/2017/03/08/point-and-click-to-an-iot-button/
[Accessed 18 April 2020].