Contents

[1 Configuration 1](#_Toc429037)

[1.1 Configuration of Google Cloud 1](#_Toc429038)

[1.1.1 Commands for creation 2](#_Toc429039)

[1.1.2 Commands compatible with Arduino Google Cloud library 4](#_Toc429040)

[1.2 Configuration of BigQuery 4](#_Toc429041)

[1.3 Configuration of Firebase 4](#_Toc429042)

[1.1.3 To visualize 6](#_Toc429043)

[1.4 Configuration of Data Studio 6](#_Toc429044)

[1.5 Adding new fields in database 6](#_Toc429045)

[1.1.4 Device configuration 6](#_Toc429046)

[1.1.5 Telemetry data 6](#_Toc429047)

[2 Daily use 7](#_Toc429048)

[3 Reply: 7](#_Toc429049)

[4 PCB 3D printing 9](#_Toc429050)

# Purpose & Description

Problem: I wanted to have an autonomous watering system for outdoor watering with the following main requirements:

* No wires or cabling
* A low-pressure thin water tube to each water station
* Low price per unit
* No maintenance or battery exchange
* Monitoring and control from app

# System overview



# Hardware & PCB

## Tools

KiCAD has been used.

## Components

## Schematics

## PCB layout

# Data flow & data structure

# Firebase Cloud

# Arduino software

# Android App

# 3D printed case

# Water system

# Daily use, installation, tuning

# Data structure

* JSON database

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table | Field name (in Arduino code) (lower is new name) | NEW JSON name (Firebase and Android) | Type | Notes / usage |
| Irrdevice |  |  |  | (on child per device) |
| metadata |  |  |  | (Semi-static configuration, can be updated runtime) |
|  | macAddr | mac | String | (not updatable, used as unique device key) |
|  | location | loc | String | Physical location, e.g. “ved gasbrænderen” |
|  | deviceID | device | String | Free to use |
|  | hardware | hw | String | Which hardware is used |
|  | softwareVersion | sw | String | Software name and compile time |
| state |  |  |  | (updated by device) |
|  | SSID | SSID | String | Current hotspot |
|  | runOnce | runOnce | boolean | Used for debugging and testing |
|  | timestamp | timestamp | long | ms since 1970. Assigned on Firebase server side |
|  | curSleepCycle  sleepCurCycle | slpCurCyc | int | Current sleep cycle. Between zero and sleepMaxCycles |
|  | secsToSleep  sleepSecondsToSleep | slpDura | Int | Seconds  See below |
|  | maxSlpCycles  sleepMaxCycles | slpMxCyc | Int | See below |
| settings |  |  |  | (can be set runtime by both device and user) |
|  | UserUpdate | Updated | boolean | Set to true by app or user if user updates any value.  Set to false by device when device reads the values. |
|  | deepSleepEnabled  sleepEnabled | slpEnabl | boolean |  |
|  | totalSecondsToSleep  sleepTotalDuration | totSlp | int | Seconds. Is converted to sleepSecondsToSleep and sleepMaxCycles as:  sleepMaxCycles = # of cycles sleeping at max\_sleep\_duration (hardware number)  sleepSecondsToSleep = remainder (last sleep cycles has this duration). |
|  | openDur  valveOpenDuration | vlvOpen | Int | Seconds |
|  | soakTime | vlvSoak | Int | Seconds |
|  | humLimit | humLim | Int | Pct humidity to trigger valve opening |
|  | mainLoopDelay | loopSec | Int | Seconds |
|  | mode  measurementMode | mode | String | “normal” “tuneSoilsensor”  “testhw” |
|  |  |  |  |  |
| telemetry\_ current |  |  |  | (last set of values received from device) |
|  | Vcc | Vcc | float | Measured battery voltage (provided measurement mode is “battery voltage”) |
|  | humidity | Hum | float | Measured humidity (provided measurement mode is “humidity”) |
|  | lastAnalogueReading | lastAnalog | int | Voltage Raw reading (0-1023) |
|  | lastOpenTimestamp | lastOpen | String |  |
|  | timestamp | timestamp | long | ms since 1970. Assigned on Firebase server side |
|  | valveState | vlvState | int |  |
|  | wifi | Wifi | int | Wifi signal strength |
| telemetry |  |  |  | all values received from device. One entry per timestamp |
|  | Equal to Telemetry\_Current |  |  |  |
| log |  |  |  |  |
|  | time | ts |  |  |
|  | text | txt |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| Telemetry\_current |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# Configuration

* 1. Links

<https://github.com/GoogleCloudPlatform/google-cloud-iot-arduino>

<https://console.cloud.google.com/home/dashboard?project=cohesive-photon-227011>

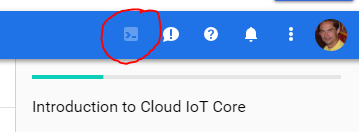
<https://console.cloud.google.com/iot/registries?project=cohesive-photon-227011&folder&organizationId&pli=1>

<https://console.cloud.google.com/iot/registries?project=autoirrigation&orgonly=true&supportedpurview=organizationId&walkthrough_tutorial_id=iot_core_quickstart>

## Configuration of Google Cloud

Follow the description on the right-side sidebar on the Google Cloud Platform.

Main steps are:

1. Create project
   1. Name: irrigation
   2. ID: irrigation-229017
   3. Number: 388781223339
2. Turn on Google Cloud APIs (through the tutorial, step 2)
3. Activate a Cloud Shell (cmd prompt)
   1. 
4. Create **topic**
   1. Name: **irr\_topic**
5. Clone the Cloud IoT Core Node.js sample files from GitHub
6. Grant permission to the Cloud IoT Core service account
7. Create a device **registry**
   1. Name: **irr-registry**
   2. Region: europe-west1
8. Generate keys
9. Create **device**
   1. Name: **irr-device**
   2. Registry: irr-registry
   3. Region: europe-west1
10. Connect your device and view telemetry data
11. Create a **subscription** to the device's topic
    1. Name: **projects/irrigation/subscriptions/my-subscription**

### Commands for creation

gcloud pubsub topics create irr-topic

cd ~

rm ~/nodejs-docs-samples/package.json

rm -rf nodejs-docs-samples

git clone https://github.com/GoogleCloudPlatform/nodejs-docs-samples.git

cd ~/nodejs-docs-samples/iot

npm --prefix ./scripts install

node scripts/iam.js irr-topic

gcloud iot registries create irr-registry \

--project=irrigation-229017 \

--region=us-central1 \

--event-notification-config=topic=projects/irrigation-229017/topics/irr-topic

cd ~/nodejs-docs-samples/iot

./scripts/generate\_keys.sh

gcloud iot devices create irr-device \

--project=irrigation-229017 \

--region=us-central1 \

--registry=irr-registry \

--public-key \

path=ec\_public.pem,type=es256

# To get the 95 char key to use in the Arduino code, ciotc\_config.h:

openssl ec -in ec\_private.pem -noout -text

cd ~/nodejs-docs-samples/iot/mqtt\_example/

npm install

node cloudiot\_mqtt\_example\_nodejs.js \

--projectId=irrigation-229017 \

--registryId=irr-registry \

--deviceId=irr-device \

--privateKeyFile=../ec\_private.pem \

--numMessages=5 \

--algorithm=ES256 \

--mqttBridgePort=443

gcloud pubsub subscriptions create \

projects/irrigation-229017/subscriptions/my-subscription \

--topic=irr-topic

# this command is used to pull one or more messages off the stack. Check –help for parameters

gcloud pubsub subscriptions pull --auto-ack \

projects/irrigation-229017/subscriptions/my-subscription \

--limit=50 --sort-by=MESSAGE\_ID

# To delete it all again

gcloud pubsub topics list

gcloud iot devices list --region=us-central1 --registry=irr-registry

gcloud iot registries list --region=us-central1

gcloud pubsub subscriptions delete projects/irrigation-229017/subscriptions/my-subscription --quiet

gcloud iot devices delete irr-device \

--region=us-central1 \

--registry=irr-registry --quiet

gcloud iot registries delete irr-registry \

--region=us-central1 --quiet

gcloud pubsub topics delete projects/irrigation-229017/topics/irr-topic --quiet

### Commands compatible with Arduino Google Cloud library

## Configuration of BigQuery

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

#Add permissions for IoT Core

gcloud projects add-iam-policy-binding irrigation-229017 --member=serviceAccount:cloud-iot@system.gserviceaccount.com --role=roles/pubsub.publisher

# Create PubSub topic for device data:

gcloud 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 --event-pubsub-topic=telemetry-topic

Overview: <https://console.cloud.google.com/iam-admin/iam/project?project=irrigation-229017&orgonly=true>

Database: irr\_data

Table: raw\_data

Link: <https://console.cloud.google.com/bigquery?orgonly=true&project=autoirrigation&p=autoirrigation&d=irr_data&page=dataset>

## Configuration of Firebase

In Google Cloud command shell:

cd ~

npm install -g firebase-tools

functions stop

npm uninstall -g @google-cloud/functions-emulator

npm install -g @google-cloud/functions-emulator

firebase login --no-localhost

# create link to project using the console

<https://console.firebase.google.com/u/0/?pli=1> (see screenshot below)

# hereafter add some more features:

firebase init # select all 5 options and use default answers

firebase functions:config:set bigquery.datasetname="irr\_data" bigquery.tablename="raw\_data"

firebase use –clear

firebase use --add irrigation-229017

firebase use --add

*? Which project do you want to add? irrigation-229017*

*? What alias do you want to use for this project? (e.g. staging) irr\_firebase*

Created alias irr\_firebase for irrigation-229017.

Now using alias irr\_firebase (irrigation-229017)

firebase deploy

*=== Deploying to 'irrigation-229017'...*

*i deploying database, storage, firestore, functions, hosting*

*i database: checking rules syntax...*

*✔ database: rules syntax for database irrigation-229017 is valid*

*i storage: checking storage.rules for compilation errors...*

*✔ storage: rules file storage.rules compiled successfully*

*i firestore: checking firestore.rules for compilation errors...*

*i firestore: reading indexes from firestore.indexes.json...*

*✔ firestore: rules file firestore.rules compiled successfully*

*i functions: ensuring necessary APIs are enabled...*

*✔ functions: all necessary APIs are enabled*

*i storage: uploading rules storage.rules...*

*i firestore: uploading rules firestore.rules...*

*Error: HTTP Error: 400, Project 'irrigation-229017' is not a Cloud Firestore enabled project.*

firebase deploy --only functions

firebase deploy --only database

firebase deploy --only hosting

firebase deploy --only storage

Link to BigQuery (Firebase project console settings => integrations)

Create database (Firebase project console => Database => create database)

firebase deploy

*=== Deploying to 'irrigation-229017'...*

*i deploying database, storage, firestore, functions, hosting*

*i database: checking rules syntax...*

*✔ database: rules syntax for database irrigation-229017 is valid*

*i storage: checking storage.rules for compilation errors...*

*✔ storage: rules file storage.rules compiled successfully*

*i firestore: checking firestore.rules for compilation errors...*

*i firestore: reading indexes from firestore.indexes.json...*

*✔ firestore: rules file firestore.rules compiled successfully*

*i functions: ensuring necessary APIs are enabled...*

*✔ functions: all necessary APIs are enabled*

*i storage: uploading rules storage.rules...*

*i firestore: uploading rules firestore.rules...*

*✔ firestore: deployed indexes in firestore.indexes.json successfully*

*i functions: preparing functions directory for uploading...*

*i hosting[irrigation-229017]: beginning deploy...*

*i hosting[irrigation-229017]: found 2 files in public*

*✔ hosting[irrigation-229017]: file upload complete*

*i database: releasing rules...*

*✔ database: rules for database irrigation-229017 released successfully*

*✔ storage: released rules storage.rules to firebase.storage/irrigation-229017.appspot.com*

*✔ firestore: released rules firestore.rules to cloud.firestore*

*i hosting[irrigation-229017]: finalizing version...*

*✔ hosting[irrigation-229017]: version finalized*

*i hosting[irrigation-229017]: releasing new version...*

*✔ hosting[irrigation-229017]: release complete*

*✔ Deploy complete!*

*… but did not show up in console, so did this:*

firebase init functions #overwriting everything

npm install --save @google-cloud/bigquery

npm install cors

edit ~/functions/index.js

*# insert the source code from below “You can see the code for the functions above”*

*# Edits:*

*# delete last () in line 3 (line with bigquery)*

*# modify topic name to “irr-topic”*

*# modify database table fields to the ones used in this project*

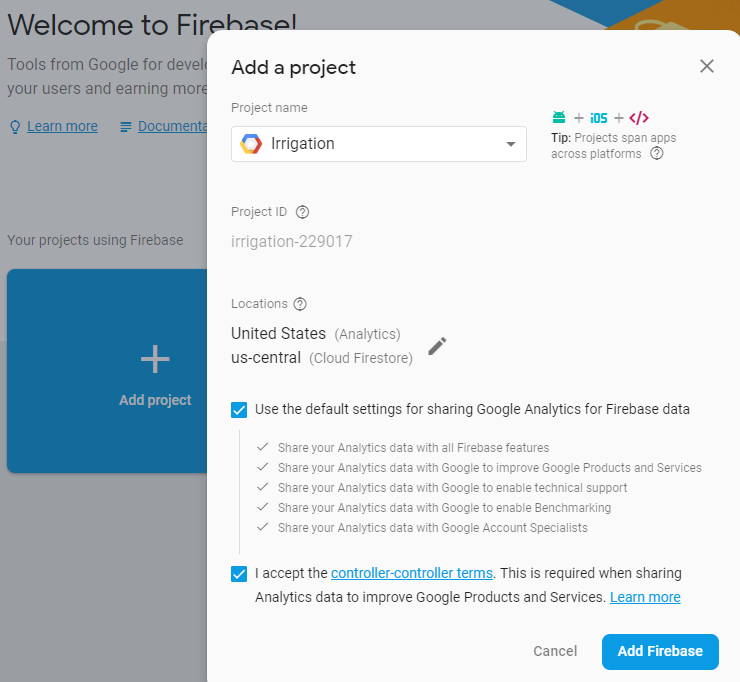
firebase deploy --only functions

Changes to source code example in …/functions/index.js:

Replace line 3 by these two lines (I think the library has been updated since this examnple was made):

const {BigQuery} = require('@google-cloud/bigquery')

const bigquery = new BigQuery()



Project overview: <https://console.firebase.google.com/u/0/project/autoirrigation/overview>

### Auth tokens (Database secrets)

Go to Project overview -> Settings -> Service accounts

Copy the “secret” and insert into Arduino code (typically called FIREBASE\_AUTH)

## Configuration of Data Studio

<https://datastudio.google.com/u/0/navigation/datasources>

## Adding new fields in database

### Device configuration

Data structure is defined in source code files:

* Globals.h
* Persistentmemory.cpp

### Telemetry data

Defined in

* AutoIrrigation.ino
  + Function CreateTelemetryJson() defines the data transmitted.
* WaterValve.h
* WaterSensor.h
* In Firebase Cloud shell (<https://console.cloud.google.com/cloudshell/editor?project=irrigation-229017&shellonly=true&fromcloudshell=true> )

edit ~/functions/index.js

* + - change “const data = {“
* In BigQuery platform (<https://console.cloud.google.com/bigquery?orgonly=true&project=irrigation-229017&p=irrigation-229017&d=irr_data&t=raw_data&page=table> )
* In DataStudio (<https://datastudio.google.com/u/0/navigation/datasources> )

## 

# Daily use

Using google cloud

Pub/Sub overview: <https://console.cloud.google.com/cloudpubsub/topics/irr-topic?project=autoirrigation&orgonly=true>

# Reply:

Hi Alvaro,

Thanks for a very good and elaborate example!!!

I made your example work, but only after a few adjustments. The Google + Firebase libraries and tools have probably been updated since this examples was made.

My configuration:

* Instead of your MongooseOS IoT device, I use an Arduino LOLIN (Wemos) D1 R2 mini using the example Esp8266-http.ino from <https://github.com/GoogleCloudPlatform/google-cloud-iot-arduino>
* BigQuery config: dataset=”irr\_data”, table=”raw\_data”
* ProjectID= irrigation-229017
* Topic: irr\_topic

During configuration of Firebase, I encountered some problems. My complete setup command sequence was this (including a few error messages).

cd ~

npm install -g firebase-tools

functions stop

npm uninstall -g @google-cloud/functions-emulator

npm install -g @google-cloud/functions-emulator

firebase login --no-localhost

# create link to project using the console

<https://console.firebase.google.com/u/0/?pli=1> (see screenshot below)

# hereafter add some more features:

firebase init # select all 5 options and use default answers

firebase functions:config:set bigquery.datasetname="irr\_data" bigquery.tablename="raw\_data"

firebase use –clear

firebase use --add irrigation-229017

firebase use --add

*? Which project do you want to add? irrigation-229017*

*? What alias do you want to use for this project? (e.g. staging) irr\_firebase*

Created alias irr\_firebase for irrigation-229017.

Now using alias irr\_firebase (irrigation-229017)

firebase deploy

*=== Deploying to 'irrigation-229017'...*

*i deploying database, storage, firestore, functions, hosting*

*i database: checking rules syntax...*

*✔ database: rules syntax for database irrigation-229017 is valid*

*i storage: checking storage.rules for compilation errors...*

*✔ storage: rules file storage.rules compiled successfully*

*i firestore: checking firestore.rules for compilation errors...*

*i firestore: reading indexes from firestore.indexes.json...*

*✔ firestore: rules file firestore.rules compiled successfully*

*i functions: ensuring necessary APIs are enabled...*

*✔ functions: all necessary APIs are enabled*

*i storage: uploading rules storage.rules...*

*i firestore: uploading rules firestore.rules...*

*Error: HTTP Error: 400, Project 'irrigation-229017' is not a Cloud Firestore enabled project.*

firebase deploy --only functions

firebase deploy --only database

firebase deploy --only hosting

firebase deploy --only storage

Link to BigQuery (Firebase project console settings => integrations)

Create database (Firebase project console => Database => create database)

firebase deploy

*=== Deploying to 'irrigation-229017'...*

*i deploying database, storage, firestore, functions, hosting*

*i database: checking rules syntax...*

*✔ database: rules syntax for database irrigation-229017 is valid*

*i storage: checking storage.rules for compilation errors...*

*✔ storage: rules file storage.rules compiled successfully*

*i firestore: checking firestore.rules for compilation errors...*

*i firestore: reading indexes from firestore.indexes.json...*

*✔ firestore: rules file firestore.rules compiled successfully*

*i functions: ensuring necessary APIs are enabled...*

*✔ functions: all necessary APIs are enabled*

*i storage: uploading rules storage.rules...*

*i firestore: uploading rules firestore.rules...*

*✔ firestore: deployed indexes in firestore.indexes.json successfully*

*i functions: preparing functions directory for uploading...*

*i hosting[irrigation-229017]: beginning deploy...*

*i hosting[irrigation-229017]: found 2 files in public*

*✔ hosting[irrigation-229017]: file upload complete*

*i database: releasing rules...*

*✔ database: rules for database irrigation-229017 released successfully*

*✔ storage: released rules storage.rules to firebase.storage/irrigation-229017.appspot.com*

*✔ firestore: released rules firestore.rules to cloud.firestore*

*i hosting[irrigation-229017]: finalizing version...*

*✔ hosting[irrigation-229017]: version finalized*

*i hosting[irrigation-229017]: releasing new version...*

*✔ hosting[irrigation-229017]: release complete*

*✔ Deploy complete!*

*… but did not show up in console, so did this:*

firebase init functions #overwriting everything

# \*\*\*\*\* IMPORTANT \*\*\*\*\*

npm install --save @google-cloud/bigquery

npm install cors

# \*\*\*\*\* IMPORTANT \*\*\*\*\*

# Finally I had to do a little, but crucial change to the source code in …/functions/index.js:

edit ~/functions/index.js

*# insert the source code from below “You can see the code for the functions above”*

*# Edits:*

Replace line 3 by these two lines (I think the library has been updated since this examnple was made):

const {BigQuery} = require('@google-cloud/bigquery')

const bigquery = new BigQuery()

*# modify topic name to “irr-topic”*

*# modify database table fields to the ones used by your IOT device*

firebase deploy --only functions

# Wrapping

Print from PCBNEW:

Front:

* Blue2 copper layer
* Edge cut
* F. crtyrd
* F. fab

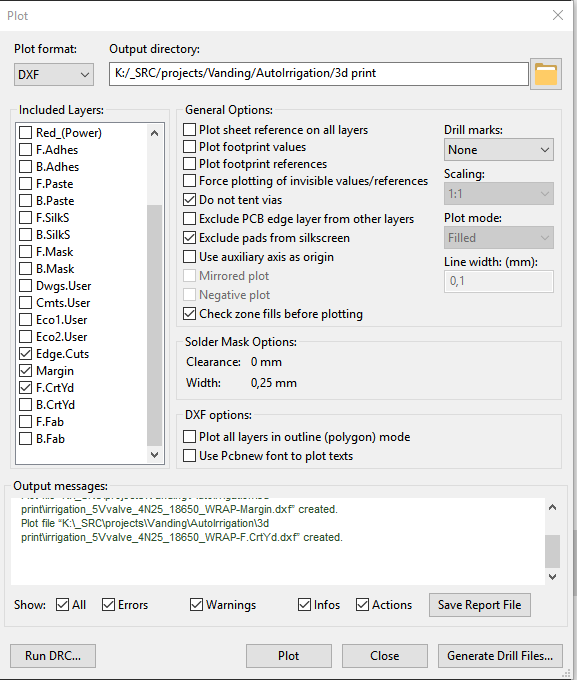
Back:

* All copper layers
* Edge cut
* MIRROR

# PCB integration in 3D print

## Export PCB (wrap) layout from kiCad to Fusion360

1. In KiCAD: File => plot => DXF format. Choose only the courtyard layer



1. In Fusion360:
   1. Open the DXF file just created (open into a new drawing)
   2. Copy all elements from this drawing
   3. Edit the sketch for the PCB board
   4. Paste into the sketch

# 3D printed case

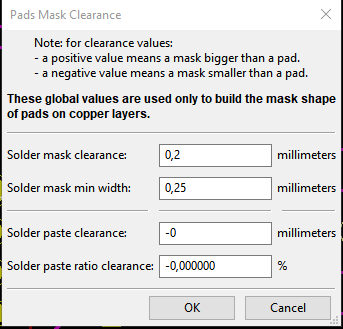
Order of design

1. Model each hardware component. Placed in separate proejcts.
2. Import needed
3. Place them physically as wished
4. Create base floor. Project spefici markers from the needed components – nothing else.

# 3D printing of PCB board

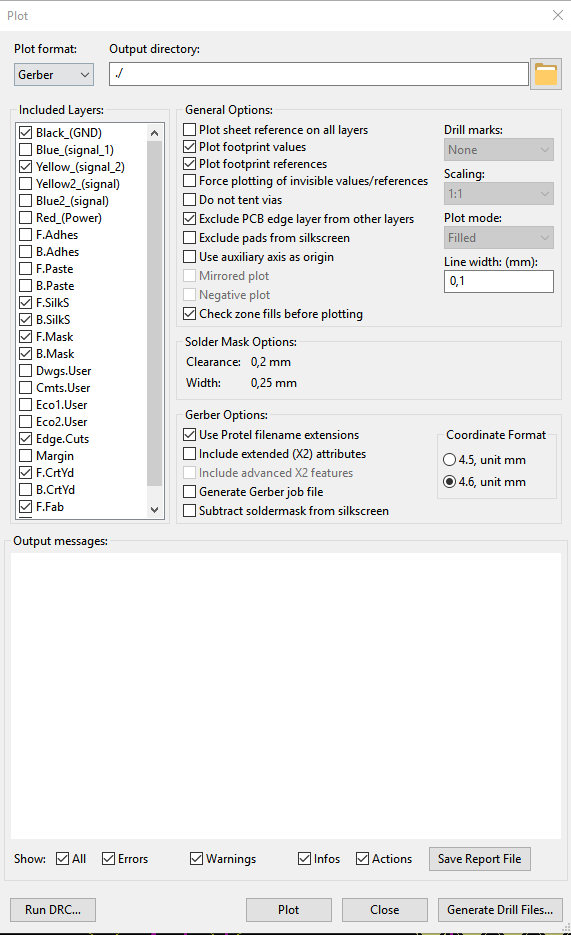
## Settings

Setup -> Pad to mask clearance:

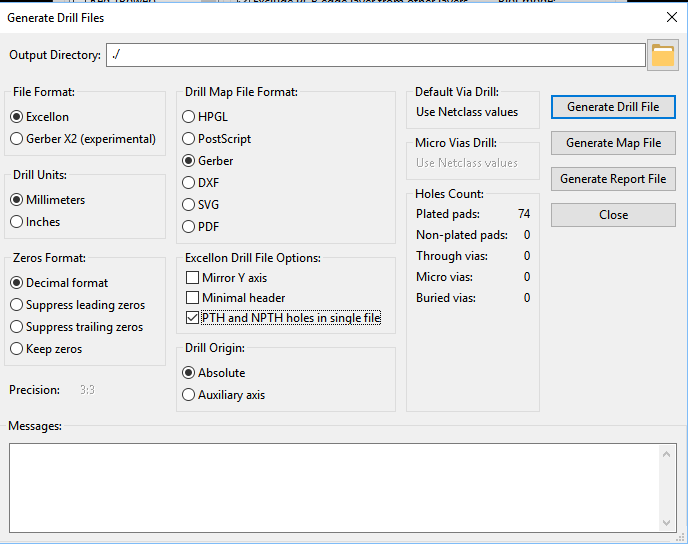


## Export to Gerber files

File -> Plot

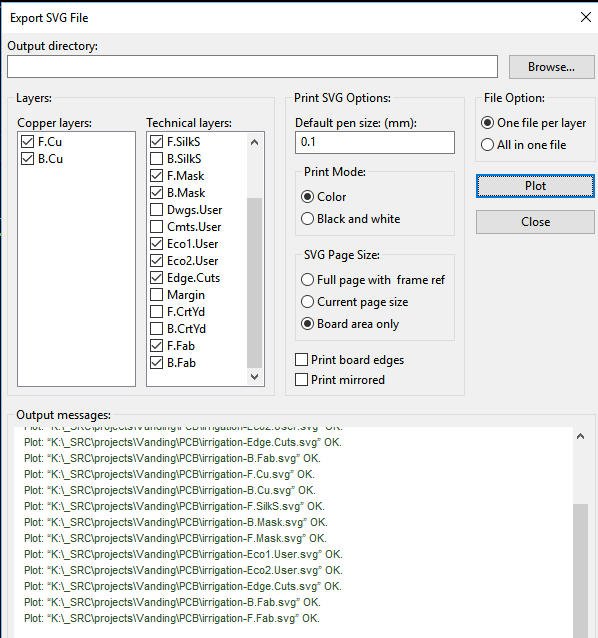


Drill holes:



Export as SVG

Import with 0.096 as scaling factor (don’t know why)



# Excel interface

<https://bigquery-connector.appspot.com/>

Use this unique key to run queries against BigQuery:

MCY6GsxRirTgTZNhLWcm2ILOlGUxHZgG8wdlNB9LPcUS7MZkOw6uyxIE8vzAO+NHog9L97z/qE17wa9hjj06dVXI+oOMZgSTonmZw/lDouQpiHMVmjMiFU60jQwGn8GmcQYiV99pVsXNgaunSq6YDpTubG+TE7D7fAs4IefKQf/9YM1U9Tp1/FjhVxZ29+fGabpn1Tm2/YmEViEkIt0yaw==

This key expires on Thursday, February 7, 2019 2:49:02 AM PST.