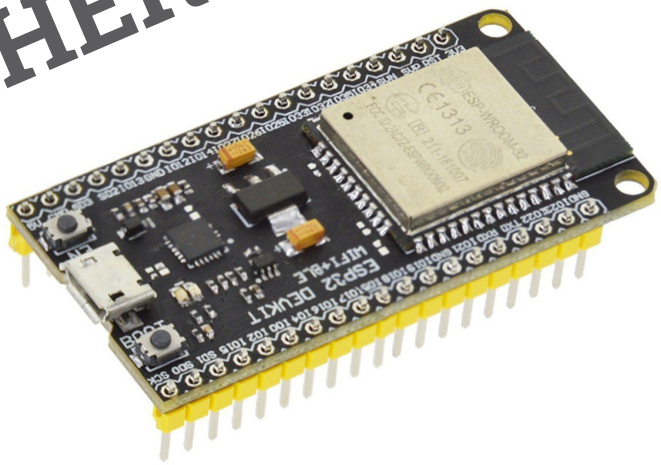
The Easiest Way to Get The Weather

OpenWeatherOneCall



Arduino & PlatformIO

V3.0.0

AN ESP32 LIBRARY

OpenWeatherOneCall

V3.0.0

http://www.github.com/jhershey69/OpenWeatherOneCall

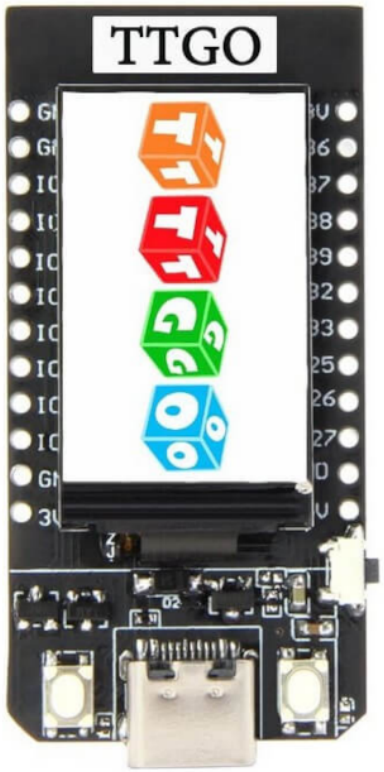


Table Of Contents

Thank you for your interest in OpenWeatherOneCall. This library gives the user the easiest method for gathering weather information for any location on the planet Earth. All care has been taken to ensure things work properly, but as you know someone will find a way to break this. The Software Testing Team is a rowdy bunch of hackers, but still they can't think of everything an end user may think to do.

If you do, in fact, find a bug please contact us with an error message (if possible) or at least what you were doing at the time, and maybe a screenshot. The development squad will get right on the issue. Anyway, let's get on with how this works. Thanks again for your interest. And a special shout out to Bill Dudley for putting up with my never ending inquisitive nature.

Jessica Hershey
Software Development

What OpenWeatherOneCall Can (and Cannot) Do

Using a **latitude and longitude** the program sends an API request to OpenWeatherMap and they return a JSON with all of the variables listed in the Variable Sheet. ***How you get the Latitude and Longitude is up to you.*** It is suggested you use a **GPS Module**, but if you are connected to a **non-Cellular Hotspot WiFi** your IP Address can be used to determine your location. This of course has limitations and may be a few meters from your actual location. **This would be a good time to mention if you are connected to a cell phone hotspot, this won't be very accurate at all.**

As an example of Hotspot inaccuracy, the software was initially tested in Toms River, NJ near the Jersey Shore. When using the Hotspot, the weather was returned for Newark, NJ, a scant 61 miles to the North. As you can see, we don't suggest using a Hotspot. Again, we recommend **non-Cellular Hotspot WiFi or a GPS Module**.

There is a third option available and that is CITY ID. CITY ID is a number assigned to a city (usually with more than 15,000 in population) and OpenWeatherMap claims this is the most accurate of all the ways to get the weather from them. However, in our testing we have discovered the list of CITY IDs they make available have many errors and throw a lot of 404 errors. If you know for sure your CITY ID is valid, then by all means use it.

OpenWeatherOneCall Installation

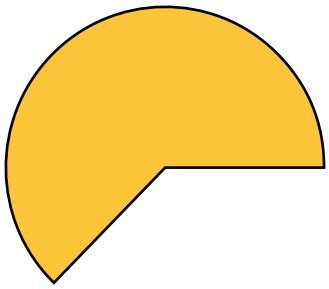
OpenWeatherOneCall installs like any other library in the Arduino Library System, if you do not use the library manager, **OR HAVE A NON-ARDUINO IDE**, then you have your own special way of installing libraries.

The library is for use on the ESP32 and was built using the Arduino IDE. We have a team now working to make the Library compatible with PlatformIO and it should be as of this release.

Because this is an Open Source project, edits to improve functionality, speed, ease of use by other members of the Open Source Community is welcome and should be submitted on the github page. Bug reports should also be submitted through the GitHub Issues Reporting system.

Once the library is installed you use it like others with:

```
#include <OpenWeatherOneCall.h>
```



Then invoke the library as:

```
OpenWeatherOneCall OWOC;
```

Please note, you can call the new instance anything you want. We use **OWOC** in our examples but you can use whatever is easier for you or makes more sense for your project. We don't judge.

Now, we move on to using the library.

OpenWeatherOneCall Options

Excludes

If left to its own devices, the OpenWeatherMap OneCall API Key (required for this library) will return the following data sets of weather information:

CURRENT
DAILY (8 day future forecast)
HOURLY (48 hour future forecast)
MINUTELY (60 Minutes of precipitation forecast)
ALERTS (Current NWS Alerts for the chosen location)

According to the OpenWeatherMap API documentation, everything ***EXCEPT*** CURRENT has a limit of 1,000 calls per day for FREE. CURRENT has a limit of 1,000,000 per month for free. So as you can see it would be wise to create projects that call the DAILY and HOURLY once every 24 hours, MINUTELY once per hour, and ALERTS no more than once every 10 minutes.

OpenWeatherMap insists they only update their weather every 10 minutes, and even though they allow 1,000,000 calls a month, for a single project only 360 a day would be updated information.

With that said, to exclude any of those data sets you will use the method

```
OWOC.setExcl(args);
```

The args for setExcl() is an **int** but for ease of use you can access it as such:

```
EXCL_C
EXCL_D
EXCL_H
EXCL_M
EXCL_A
```

That is, in order of appearance, Current, Daily, Hourly, Minutely, and Alerts. Place the excludes you want in the method call like this:

```
OWOC.setExcl(EXCL_D+EXCL_M+EXCL_A+EXCL_H);
```

OpenWeatherOneCall Location

Latitude/Longitude

By using a GPS, WiFi Triangulation, or manually entering the Longitude and Latitude you will get weather information for your exact desired location. **WiFi Triangulation** requires an additional library called **WiFiTri** available from the same GitHub Repository owner as this library. Also required is an additional Google Developer Key. This key was free when this document was created and allowed 100,000 calls for reverse GEOCODING. Please check Google for updated information and any fees.

Location detection by IP ADDRESS is also done by this library, but please be advised using a **CELLULAR HOTSPOT** will give erroneous results. As an example, the library was testing in Toms River, NJ on a hotspot, and the weather returned for Newark, NJ a distance of 61 miles to the north.

How you get the Latitude and Longitude is up to you, but it gets inserted like this:

```
OWOC.setLatLon(float latitude,float longitude);
```

If you call `OWOC.setLatLon()` with no arguments, IP ADDRESS location will take over.

City ID

This is probably the least accurate of all calls for the weather, but OpenWeatherMap insists it is the most accurate. (It is not, we've tested it a lot) OpenWeatherMap has a zipped file of city identification numbers they use available, but it is out of sync, missing some cities, and others are just plain incorrect. But we put the calling method here anyway.

```
OWOC.setLatLon(CITY_ID);
```

OpenWeatherOneCall Weather

Finally. How to get the weather data. Please be sure you have set all the options previously described. To review, we've set the units of measure, the location, the data to include, and if we want historical data. So, here go...

Current/Historical

As of V3.0.0 to call the weather from OpenWeathermap you do this:

```
OWOC.parseWeather();
```

That's it. No more unearthly long string of arguments! We set everything up already, and you can now change anything on it's own so you don't have to inject that long list of arguments just to change between Metric and Imperial, for example.

****IF YOU ARE UPGRADING FROM V2.0.2 OR EARLIER****

There is a ***LEGACY*** calling method so you can continue to use the old method with all the arguments and will still benefit from the latest version using new memory and other improvements. For those of you who are starting with V3.0.0 and are wondering what I'm talking about, here is the monster:

parseWeather(DKEY,GKEY,SEEK_LATITUDE,SEEK_LONGITUDE,SET_UNITS,CITY_ID,API_EXCLUDES,GET_HISTORY)

Horrid. But it still works so you can still use it. See previous versions of the documentation to set it up.

Alerts

When you include ALERTS (remember there is a fee for over 1000 calls a day) you receive NWS Alerts for the Latitude and Longitude sent. You receive Sender Name, Sender Location, Alert Description, as well as Start and End times. For testing purposes you will need to search the NWS Alerts Webpage to find a location with an alert if your location doesn't have one.

[Active Alerts (weather.gov)]

An example:

Message: NOAA-NWS-ALERTS-CA125F76320144.HighSurfAdvisory.125F763EF2F0CA.MTRCFWMTR.9609375b886768091642456b51fe6d48 from w-nws.webmaster@noaa.gov
Sent: 10:17 PST on 12-14-2020
Effective: 10:17 PST on 12-14-2020
Expires: 19:00 PST on 12-14-2020

Event: **High Surf Advisory**

Alert:

...LARGE, LONG PERIOD NORTHWEST SWELL WILL BRING HAZARDOUS CONDITIONS ALONG THE COAST THROUGH MONDAY...

...KING TIDES WILL CAUSE MINOR COASTAL OVERFLOW AND FLOODING THROUGH TUESDAY MORNING...

A large, long period NW swell will peak in the waters today, bringing large breaking waves of 15 to 20 feet, locally up to 25 feet, at favored breakpoints. Additionally, King Tides have returned to the region and will ebb and flood through Tuesday. Large swell long period swell and highest high tides of the year will overlap and allow the intrusion of seawater into low lying areas, generating minor coastal flooding. The two main time periods of concern occur during the highest of the high tides this morning and Tuesday morning. Thus, the surf zone/area beaches will be hazardous into Tuesday afternoon.

...COASTAL FLOOD ADVISORY REMAINS IN EFFECT UNTIL 1 PM PST TUESDAY...

OpenWeatherOneCall Misc

Display/Printing

Please be advised, you can't get HISTORICAL and CURRENT weather at the same time from OpenWeatherMap. Just can't. They don't do that. It's two different calls. You can do them one after the other by setting your HISTORY to a number to get HISTORICAL, and then setting your HISTORY to 0 or NULL to get CURRENT. That's just how it works over there at OpenWeatherMap.

Also, please check for the existence of a variable before trying to use it. You'll crash. I swear you will. We really went out of our way, used many man hours, and woman hours too, to make sure everything was a NULL or ZERO value, but check it anyway. You can check each section's struct for existence by doing this:

```
if(OWOC.current)
if(OWOC.hour)
if(OWOC.minute)
if(OWOC.alert)
if(OWOC.forecast)
```

The best rule of thumb is, **"IF YOU DIDN'T INCLUDE IT DON'T TRY TO USE IT!"** The program releases all unused memory now and you really need to check to be sure it is there. Memory on the ESP32 is larger than most Arduinos but still as programmers we'd be remiss if we didn't try to use as little as possible. Right?

Additional Libraries

Previous versions of this library listed HTTPClient as a dependency. While it is not required to be installed, it is required to be #included. There are dozens of HTTPClient libraries all with different forms of Camel Case names, and the one used in the ESP32 is built in, but still must be included. Other libraries are also required for NTPTime gathering to create proper HISTORICAL URL forming. Please consult all documentation and sketches to be sure you are using the proper additional libraries.

OpenWeatherOneCall Variables

Please note most variables are accessed by "`section[x].variable`" in dot format, while the CURRENT and ALERTS data sections are "`section->variable`" in pointer format.
Please download the VARIABLES PDF and keep it handy for reference...

OpenWeatherOneCall Revisions

1.2.0 - Added CITY ID option
1.3.0 - Added exclude values for API call
2.0.0 - Added HISTORICAL WEATHER, uses UNIX EPOCH TIMESTAMP see DOCs
2.0.1 - Fixed Historical Data bug for EPOCH calculation
2.0.2 - Added units (Kelvin/Metric/Imperial) to Historical
2.1.0 - Added TIMEZONE and OFFSET, removed unused vars left over from Dark Sky
3.0.0 -

- Changed calling method to less confusing.
- One Call Key is now set as an individual function call
- parseWeather() no longer takes **ANY** arguments!
- Added EXCLUDES and memory constructor for controlling memory use.
- Added ALERTS and fixed SUMMARY to include the entire message body.
- Added error tracing. (See script examples)
- Removed extra dead code and variables. (Transparent to user)
- Separated all functions into their own methods. (Transparent to user)
- Removed all references to Google
- WiFi Triangulation is a separate library to avoid needing more than one API Key.
- UNITS defaults to Imperial.
- Added IP Address location tracing. (DOES NOT WORK ON MOBILE HOTSPOTS)
- Variables for “Daily” have been fixed. temperatureHigh and temperatureLow are now loading proper amounts. They used to load Day and Eve values.
- All arrays allocate memory at run time as needed. For single layer arrays like: current.temperatureHigh please use current->temperatureHigh or current[0].temperatureHigh
- HTTPClient is not a “depends on” for the ESP32. It is built in. But still needs an #include
- Legacy Mode so no changes need to previous code using v2.0.2 or earlier

http://www.github.com/jhershey69/OpenWeatherOneCall

[THIS PAGE LEFT INTENTIONALLY BLANK]