# Geolocating with ESP8266

Version 1.0

Copyright © 2017

# About This Guide

This documents introduces how to obtain a device location with ESP8266.

| Chapter | Title | Content |
|---|---|---|
| Chapter 1 | Overview | Overview of geolocating with ESP8266. |
| Chapter 2 | System Architecture | System architecture regarding geolocating with ESP8266. |
| Chapter 3 | Geolocating with ESP8266 | Procedures and commands used to geolocate a device with ESP8266. |
| Chapter 4 | Power Consumption | Power consumption of the ESP8266 when used to locate a connected device. |
| Chapter 5 | Conclusion | Conclusion. |

## Release Notes

| Date | Version | Release notes |
|---|---|---|
| 2017.09 | V1.0 | Updated version. |

## Documentation Change Notification

**Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe *here*.**

# Table of Contents

# 1. Overview

Locating a networking device is of critical importance in B2B applications and applications that interact with the end user directly. The device manufacturer may often need access to the location information of the device in order to provide a better user experience or implement some integral features of the device. There are various approaches towards geolocating a device (in terms of latitude and longitude) on the globe.

The first and rather conventional approach is to use a GPS module. This approach provides the best available accuracy for geolocating a device, and is extremely reliable. The second method for obtaining a fix on the device location is to use the mobile communication system (GSM, LTE, etc). However, not all devices support integrated cellular connectivity. The third approach is to use nearby internet-connected Wi-Fi access points and obtain an approximated geolocation for the target device.

The latter two approaches require internet connectivity, which is often available in most ESP8266-based applications.

# 2. System Architecture

This application note assumes that the user device connects to Wi-Fi networks, using an ESP8266 module (such as the ESP-WROOM-02 from Espressif). The ESP8266 module supports official Espressif AT command firmware based on non-OS SDK V2.1.0 (as of July, 2017).

The device is managed by a host controller that is responsible for all processing tasks. This host controller is connected to the ESP8266 module via UART.

This application note describes how the ESP8266 module may be used to scan for nearby Wi-Fi access points and, then, use their SSID, RSSI and MAC address to obtain a potential fix on the device's geolocation, using Google geolocation API. This application note also takes into consideration the availability of cellular network information.

Finally, it is demonstrated how the ESP8266 module may be used to connect securely to Google API and obtain a fix based on the above-mentioned data.

# 3. Geolocating with ESP8266

As mentioned above, ESP8266 can be used for geolocation by firstly obtaining nearby AP properties, and then using Google geolocation API to locate the user-device.

To be able to obtain a fix on the location of the device that integrates the ESP8266 chip, we assume that the host controller first could obtain data from nearby Wi-Fi networks or cellular sub-systems. The data is then consolidated into a data block that must be sent to an online geolocation API or service that will estimate the device location in terms of latitude, longitude and accuracy of the fix.

## 3.1. Obtaining AP Property

The ESP8266 is capable of actively scanning for nearby Wi-Fi access points operating in the 2.4GHz band. The scanning only takes a couple of seconds and the ESP8266 may then be put in a power-saving mode, thus saving power in battery-powered systems.

This is achieved by issuing a single AT command that lists all APs available in the area, along with their corresponding SSID, RSSI and MAC address:

1. Firstly, make sure that ESP8266 is powered up and functional by issuing command:

   ```
   AT
   ```

   The response must be:

   ```
   OK
   ```

> 📖 ***Note:***
>
> *All the **commands issued must be terminated with CR-LF** (\r\n combination).*

2. Then, the host controller can obtain a list of all available Wi-Fi APs by issuing command:

   ```
   AT+CWLAP
   ```

   In response to this command, the ESP8266 will return a list of APs available, and their corresponding SSID (the second parameter), RSSI value (the third parameter) and MAC address (the fourth parameter), which are important defining parameters that also are required for the geolocation.

   An example of the output data is:

   ```
   +CWLAP:(2,"HotelFlower",-76,"c8:3a:35:b3:16:48",11,0,0)
   +CWLAP:(3,"IoTBits",-93,"08:bd:43:66:6a:86",6,-27,0)
   OK
   ```

> 📖 ***Note:***
>
> *For more information on the response data, please consult the AT instruction set documentation, corresponding to your AT firmware version, available on our website: http://espressif.com/sites/default/files/documentation/4a-esp8266_at_instruction_set_en.pdf*

## 3.2. Obtaining Geolocation

To prepare the ESP8266 for establishing a connection with the online geolocation API or service, the model must be connected to a router with internet access. This can be done by setting the ESP8266 to station mode (so that it can be connected to a router).

### 3.2.1. Establishing a secure connection

1. Set ESP8266 to station mode:

   ```
   AT+CWMODE=1
   ```

   An access point can now be connected to ESP8266 by using:

   ```
   AT+CWJAP="SSID","password"
   ```

   If the connection is successful and the AP assigns the ESP8266 with a valid local IP address, the command will return:

   - `WIFI CONNECTED`
   - `WIFI GOT IP`

   The ESP8266 can now be made to establish connection with a geolocation API or service over a secure connection (https).

2. Initialize the SSL buffer as follows:

   ```
   AT+CIPSSLSIZE=2048
   ```

3. Connect securely to *Google API* by issuing command:

   ```
   AT+CIPSTART="SSL","www.googleapis.com",443
   ```

   If everything works as expected, this command will return:

   - `CONNECT`
   - `OK`

   Otherwise, the ESP8266 will respond with ERROR.

> 📖 Note:
>
> *SSL is memory-intensive. Make sure that, even in the worst-case scenario, no outgoing and incoming transaction sizes exceed the size of SSL buffer.*

### 3.2.2. Obtaining a location fix by using Google API

The Google geolocation API requires the user to send in defining parameters via a POST request, formatted as a JSON string, to obtain a location fix. The URL must contain the user API key for quota management purposes by Google.

The detailed steps for obtaining a location fix by using Google API are as follows:

1. Send a standard HTTP POST header with your API key
2. Send POST data with JSON-formatted geolocating parameters

The host controller should initiate a request at this stage. To send data to the Google API server, issue a send command followed by the data length to be sent (336 for the sample request used here):

```
AT+CIPSEND=336
```

When the ESP8266 responds with "**>**", the actual request should be sent in as follows:

```
POST /geolocation/v1/geolocate?key=AIzaSyCNStbdlDS_L HTTP/1.1
Host: www.googleapis.com
Content-Type: application/json
Content-Length: 166

{"homeMobileCountryCode": 310, "homeMobileNetworkCode": 410, "radioType":
"gsm", "carrier": "Vodafone", "considerIp": "true", "cellTowers": [],
"wifiAccessPoints":[]}
```

📖 Note:

- *The JSON-formatted request body must comply with the structure outlined in the Google geolocation API documentation: https://developers.google.com/maps/documentation/geolocation/intro.*
- *The wifiAccessPoints field is not populated in the above example to simplify the request content.*

# 4. Power Consumption

## 4.1. Testing Environment

For evaluating power consumption, the demonstration in this application note uses the standard AT command firmware for ESP8266 (based on non-OS SDK V2.0.0) and the following hardware:

- PC as the host controller
- ESP-Launcher as the ESP8266 device
- NetGear 2.4 GHz router with internet access
- 26 MHz crystal oscillator
- 40 MHz SPI flash (QIO mode)

Some methods of reducing power consumption:

- Disable RF calibration on waking up from deep sleep;
- Use low power flash variants running at 26 MHz, if acceptable;
- Use non-FOTA firmware to slightly reduce wake-up time;
- Use low transmit power if possible.

> 📖 Note:
> - *The current consumption data shown in this chapter includes the flash chip power consumption as well.*
> - *Please consult the relevant AT instruction set documentation for further information.*

## 4.2. A Brief Note on Power Consumption

The approximate current consumption in some low-power modes is as follows:

- Modem sleep mode: ~15 mA
- Light sleep mode: ~3 mA
- Deep sleep mode: ~20 uA (please account for flash standby mode current)

Assuming that the sleep mode is set to Modem-sleep (i.e. `AT+SLEEP=2`) and the DTIM interval is ~100ms, when connected to an AP, the power consumption trends will be similar to what is outlined below:

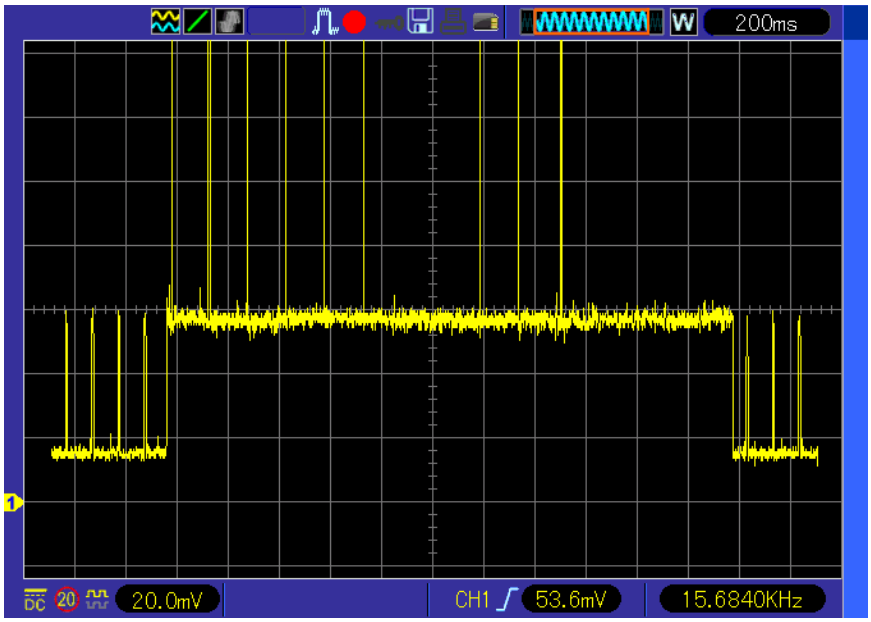


**Figure 5-1. Active SSID Scanning with Modem-sleep Mode Enabled (X: 200 ms/div, Y: 20 mA/div)**

- For applications that stay connected to APs:

  Assuming that the ESP8266 is connected to an AP with light sleep between beacons activated, and DTIM beacons every 100 ms, the average current consumption when maintaining connection with an AP is less than 20 mA.

- For applications that simply scan for APs:

  The typical scan time for active SSID scanning can range from 2 sec to 2.5 sec. Note that passive SSID scanning may be carried out using the Wi-Fi sniffer APIs provided in the ESP8266 SDK. Passive scanning is not available via AT command firmware. The average current consumption is about 75 mA when actively scanning for APs.

Besides, the device may be put to deep sleep when scanning is not required (using **AT+GSLP=<time>**). The power consumption for the ESP8266 in deep-sleep mode is only in the order of micro amperes and removing the power supply from the ESP8266 is not required. In the test case above, the number of APs detected by the ESP8266 was over ten in about 2.2 sec.

> 📖 Note:
>
> *The power consumption data for transmitting the geolocation parameters over the Wi-Fi network has not been listed in this application note, because it depends heavily on the network setup, latency of network and interference from nearby devices. We recommend that the end user evaluates this under the conditions in which their product is supposed to operate.*

# 5. Conclusion

It can thus be concluded that the ESP8266 may be used for geolocation in multiple ways: by obtaining AP properties and then using Google geolocation API to locate a user-device, or by locating the end-user's geolocation directly, when the ESP8266 has access to the internet.

Availability of the cellular network parameters is not a necessity for implementing geolocation with ESP8266. However, the accuracy of results may vary depending on how many access points are available and connected to the internet.

The power consumption of the ESP8266 module, when used as a geolocating device, may be kept extremely low with Deep-sleep mode enabled between scans. For mains powered applications where power consumption is not critical, the ESP8266 may be put in Light-sleep or Modem-sleep mode between scans for acceptable power consumption.

Espressif IoT Team

*www.espressif.com*

**Disclaimer and Copyright Notice**

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE.

All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The Wi-Fi Alliance Member logo is a trademark of the Wi-Fi Alliance. The Bluetooth logo is a registered trademark of Bluetooth SIG.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

**Copyright © 2017 Espressif Inc. All rights reserved.**