



# **SNTP Time Synchronization**

---

Displaying Local Time on the Web Page

The background features a complex network of thin, light-colored lines and small circles, forming a web-like structure. Overlaid on this are several translucent triangles of various sizes and orientations. The overall aesthetic is clean, modern, and technical, suggesting a focus on geometry or data visualization.

# **Our Implementation**

---

# SNTP Time Synchronization Implementation

- About the Implementation

- Once the ESP32 has an internet connection (connected to an AP/Router), the SNTP task start function will be called.
- The task start function will set off the FreeRTOS time synchronization task, which will call a function to obtain the updated time → In this implementation, the task will keep synchronizing/checking that the time is up-to-date.
- The obtain time function will initialize the SNTP service to query an SNTP server for the universal time – the obtain time function will reinitialize in the case the time is not up-to-date.
- The local time zone will be set after SNTP service is initialized.
- The web server will respond with the updated time once time service is initialized.



# **Additional Information About SNTF**

Brief Background Information

# SNTP

- About

- SNTP (Simple Network Time Protocol) is a protocol designed to synchronize the clock of devices connected to the internet.
- *Basic operation is as follows:*
  - The client device connects to the server using the UDP protocol on port 123.
  - The client transmits a request packet to the server.
  - The server responds with a time stamp packet.
  - The client can then parse out the current date and time values.
- If the ESP32 is connected to the internet, it can get the date and time using SNTP.

*Note: The Simple Network Time Protocol (SNTP) in ESP-IDF is supported by [LWIP functions](#).*

The background features a complex network of thin, light-colored lines connecting small circular nodes, primarily concentrated on the left side. Scattered across the entire background are various geometric shapes, including triangles and polygons, some of which are outlined in a light yellow or orange color. The overall aesthetic is technical and modern.

# **ESP-IDF APIs Used for SNTP Time Synchronization**

---

# Utilizing ESP-IDF for SNTP Time Synchronization

- Suggested Reading
    - SNTP Time Synchronization → [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/system\\_time.html#sntp-time-synchronization](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/system_time.html#sntp-time-synchronization)
    - Time zones → [https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/system\\_time.html#timezones](https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/system_time.html#timezones).
-

---

# Utilizing ESP-IDF for SNTP Time Synchronization

- Configuration Steps and Notible APIs Used
    - After the FreeRTOS task kicks off and the obtain time function is called, the initialize SNTP function is called, where the first SNTP function used, configures the client in *poll* mode to query the server every n seconds → `sntp_setoperatingmode(SNTP_OPMODE_POLL)`.
    - Also, within the initialize SNTP function, we will tell the client which server to use. A common choice is a cluster of servers from `pool.ntp.org` → `sntp_setservername(0, "pool.ntp.org")`.
    - Then we'll initialize service → Using `sntp_init()`.
    - Set the TZ variable and initialize the time zone conversion → set the time zone e.g., `setenv("TZ", "CET-1", 1)`; and `tzset()`; to initialize the timezone conversion routine.
-

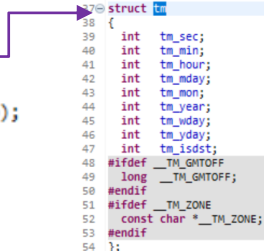


# Utilizing ESP-IDF for SNTP Time Synchronization...

- After SNTP service is initialized, we'll need to check if the system clock has the updated time...
  - To get the actual time from the system clock, we use the `time()` which updates the `time_t` variable.
    - To split the variable into different time values (year, month, day...), the `localtime_r` function is used which updates the `tm` struct, as shown here:

```
time_t now = 0;
struct tm time_info = {0};

time(&now);
localtime_r(&now, &time_info);
```



A diagram with a purple arrow pointing from the `time_info` variable in the first code block to the `struct tm` definition in the second code block.

```
38 struct tm
39 {
40     int    tm_sec;
41     int    tm_min;
42     int    tm_hour;
43     int    tm_mday;
44     int    tm_mon;
45     int    tm_year;
46     int    tm_wday;
47     int    tm_isdst;
48     #ifdef __TM_GMTOFF
49     long    __TM_GMTOFF;
50     #endif
51     #ifdef __TM_ZONE
52     const char *__TM_ZONE;
53     #endif
54 };
```

- We'll then check the information from the `tm` struct to see if the time was set yet.
- ```
{
    if (time_info.tm_year < (2016 - 1900))
    {
        sntp_time_sync_init_sntp();
        // Set the local time zone
        setenv("TZ", "CET-1CEST,M3.5.0,M10.5.0/3", 1);
        tzset();
    }
}
```

The background features a complex network of thin, light-colored lines and small circles, forming a web-like structure. Overlaid on this are several triangles of various sizes and orientations, some of which are filled with a light blue or yellow color. The overall aesthetic is clean and modern, with a focus on geometric shapes.

**Let's get started!**