SNTP Time Synchronization

Displaying Local Time on the Web Page

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 SNTP

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.

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
  - Time zones→ 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:

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

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

```c
struct tm
{
  int   tm_sec;
  int   tm_min;
  int   tm_hour;
  int   tm_mday;
  int   tm_mon;
  int   tm_year;
  int   tm_wday;
  int   tm_yday;
  int   tm_isdst;
#ifdef __TM_GMTOFF
  long  __TM_GMTOFF;
#endif
#ifdef __TM_ZONE
  const char *__TM_ZONE;
#endif
};
```

    - We'll then check the information from the tm struct to see if the time was set yet.

```c
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();
}
```

Let’s get started!