

Couleur d'un jour Tempo EDF

J. Lemaire

Juillet 2025 (révision)

1 Contrat Tempo EDF

EDF propose maintenant à ses clients particuliers l'option **Tempo** qui fait varier le tarif de la consommation d'électricité en fonction de la couleur du jour (Bleu, Blanc, Rouge) et du type d'heure (Creuse ou Pleine)¹.

Dans une année, il y a :

- 300 jours Bleu
- 43 jours Blanc
- 33 jours Rouge

Cette couleur est déterminée chaque jour vers 10h30 pour le lendemain par RTE² et le calendrier des couleurs journalières peut être consulté^{3 4} ; voici par exemple ce qu'on obtient en France métropolitaine :

En ce moment



¹ <https://particulier.edf.fr/fr/accueil/gestion-contrat/options/tempo/details.html>

² Réseau de Transport d'Electricité

³ <https://particulier.edf.fr/fr/accueil/gestion-contrat/options/tempo.html#/>

⁴ <https://www.services-rte.com/fr/visualisez-les-donnees-publiees-par-rte/calendrier-des-offres-de-fourniture-de-type-tempo.html>

Pour l'option Tempo, les heures sont réparties chaque jour comme suit :

- De 0h à 6h : heure Creuse
- De 6h à 22h : heure Pleine
- De 22h à 0h : heure Creuse

La grille des tarifs en vigueur actuellement (depuis le 1^{er} février 2024)⁵ permet de comparer les 3 offres du Tarif Bleu :

- Option de **Base**
- Option **Heures Creuses**
- Option **Tempo**

Voici par exemple les tarifs actuels (le 20/07/2025) pour un abonnement de 9kVA :

Option	Abonnement mensuel TTC (€)	Prix du kWh TTC (€)					
		Heure Pleine			Heure Creuse		
		Jour Bleu	Jour Blanc	Jour Rouge	Jour Bleu	Jour Blanc	Jour Rouge
Base	17,27	20,16					
Heures creuses	18,01	21,46			16,96		
Tempo	17,45	15,52	17,92	65,86	12,88	14,47	15,18

On constatera⁶ ici que, si les tarifs de l'abonnement sont proches, ceux de la consommation sont très différents ; par exemple, en heure **Pleine**, entre l'option de **Base** et l'option **Tempo**, on note :

- -23% pour un jour **Bleu**
- -11% pour un jour **Blanc**
- +226% pour un jour **Rouge**

Il importera donc de bien connaître la couleur du jour et celle du lendemain si l'on souhaite profiter des avantages de l'option Tempo.

A noter également que la couleur d'un jour Tempo ne s'applique qu'à partir de 6h et ce jusqu'à 6h le lendemain ; chaque jour, c'est donc la couleur du jour précédent qui s'appliquera entre 0h et 6h ; par exemple, comme le calendrier de **Février 2024** indique :

- 11/02/2024 jour **Bleu**
- 12/02/2024 jour **Rouge**
- 13/02/2024 jour **Blanc**

les tarifs horaires à appliquer, décalés de 6h, seront les suivants :

	0h	6h	22h	0h	6h	22h	0h
Jour	12-févr				13-févr		
Tarifs	HC	HP	HC	HP	HP	HC	HC

Il faut donc bien faire la différence entre la couleur Tempo d'un jour donné et la tarification Tempo qui s'applique à un instant donné.

⁵ https://particulier.edf.fr/content/dam/2-Actifs/Documents/Offres/Grille_prix_Tarif_Bleu.pdf

⁶ Pour une analyse plus fine, cf. <https://www.fournisseurs-electricite.com/fournisseurs/edf/tarifs/tempo> par exemple.

2 API Tempo Like Supply Contract

Il y a quelques mois, on pouvait encore obtenir très simplement la couleur Tempo d'un jour, en appelant un service EDF avec l'URL suivante, où il suffisait de renseigner la date de ce jour sous la forme YYYY-MM-DD :

<https://particulier.edf.fr/services/rest/referentiel/searchTempoStore?dateRelevant=YYYY-MM-DD>

Malheureusement ce service ne fonctionne plus et il faut maintenant appeler une API plus complexe, publiée par RTE, **Tempo Like Supply Contract**, décrite dans le catalogue des API RTE dédiées à la consommation⁷ :



En cliquant sur **Découvrir l'API**, on affiche sa page Web⁸ :



En cliquant sur le lien **Guide utilisateur de l'API**, on peut faire afficher son manuel d'utilisation et le télécharger en format pdf⁹. Il décrit le moyen d'obtenir la couleur d'un ou plusieurs jours, en formulant une requête HTTP GET à laquelle le serveur répondra, dans un format JSON¹⁰ ; par

⁷ <https://data.rte-france.com/catalog/consumption>

⁸ <https://data.rte-france.com/catalog/-/api/consumption/Tempo-Like-Supply-Contract/v1.1>

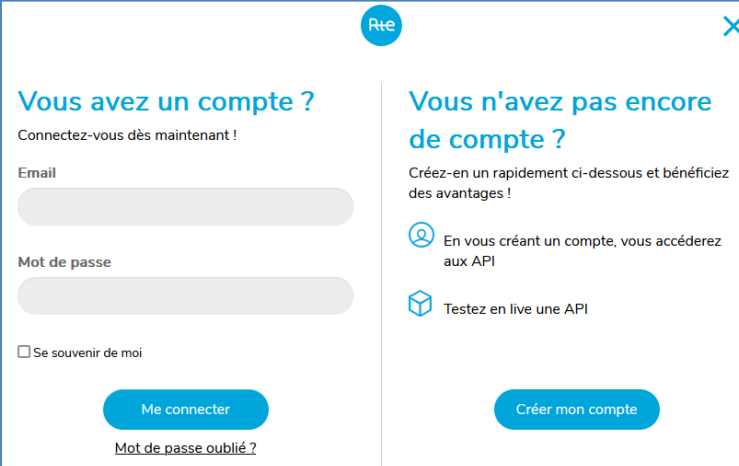
⁹ https://data.rte-france.com/documents/20182/224298/FR_GU_API_Tempo_Like_Supply_Contract_v01.02.pdf

¹⁰ Java Script Object Notation.

exemple, voici ce qu'on obtient comme réponse quand on demande la couleur des 11, 12 et 13 février 2024 :

```
{
  "tempo_like_calendars": {
    "start_date": "2024-02-11T00:00:00+01:00",
    "end_date": "2024-02-14T00:00:00+01:00",
    "values": [
      {
        "start_date": "2024-02-13T00:00:00+01:00",
        "end_date": "2024-02-14T00:00:00+01:00",
        "value": "WHITE",
        "updated_date": "2024-02-12T10:20:00+01:00"
      },
      {
        "start_date": "2024-02-12T00:00:00+01:00",
        "end_date": "2024-02-13T00:00:00+01:00",
        "value": "RED",
        "updated_date": "2024-02-11T10:20:00+01:00"
      },
      {
        "start_date": "2024-02-11T00:00:00+01:00",
        "end_date": "2024-02-12T00:00:00+01:00",
        "value": "BLUE",
        "updated_date": "2024-02-10T10:20:00+01:00"
      }
    ]
  }
}
```

Pour formuler avec succès de telles requêtes, il faut commencer par s'abonner à l'API en cliquant sur **Abonnez-vous à l'API** ; la page qui s'affiche demande une connexion à un compte RTE, si ce n'est pas déjà fait :

The image shows a web interface for the RTE API. It has a light blue header with the RTE logo and a close button. The main content is divided into two columns. The left column, titled 'Vous avez un compte ?' (Do you have an account?), includes a sub-header 'Connectez-vous dès maintenant !' (Log in now!), input fields for 'Email' and 'Mot de passe' (Password), a checkbox for 'Se souvenir de moi' (Remember me), a blue 'Me connecter' (Log in) button, and a link for 'Mot de passe oublié ?' (Forgot password?). The right column, titled 'Vous n'avez pas encore de compte ?' (You don't have an account yet?), includes a sub-header 'Créez-en un rapidement ci-dessous et bénéficiez des avantages !' (Create one quickly below and benefit from the advantages!), two icons with text: a person icon for 'En vous créant un compte, vous accéderez aux API' (By creating an account, you will access the APIs) and a cube icon for 'Testez en live une API' (Test an API in live), and a blue 'Créer mon compte' (Create my account) button.

Quitte à re cliquer à nouveau sur **Abonnez-vous à l'API**, on est alors conduit à créer une **Application** :

CRÉER UNE APPLICATION

Nom

Tempo

Type

☒ Web / Serveur ☐ Mobile

Description

Couleur du jour Tempo EDF

Valider

Celle-ci apparaîtra alors dans la liste **Mes applications** :

The screenshot shows a web browser window with the URL <https://data.rte-france.com/group/guest/apps>. The page features the RTE logo and a user profile for Jacques LEMAIRE. A sidebar on the left contains navigation links: 'API, FAQ, applications', 'Mes applications' (selected), 'Toutes les catégories', 'Partenaires', and 'Production'. The main content area displays a list of applications, with 'Tempo' selected. The application details show the name 'Tempo', the description 'Couleur du jour Tempo EDF', the number of linked APIs as '1', and the type as 'WEB'.

En cliquant dessus, on constate que celle-ci est caractérisée par un **ID Client** et un **ID Secret** :

TEMPO

Statut : activé

ID Client 508

6c3 |

ID Secret c35:

ibc0

Copier en base 64

Réinitialiser

Couleur du jour Tempo EDF

Web / Serveur

Dans notre cas :

ID Client = 508-----6c3

ID Secret = c35-----bc0

Pour la suite, il sera utile de récupérer leur **codage en base 64**, en cliquant sur **Copier en base 64** :

NTA-----A==

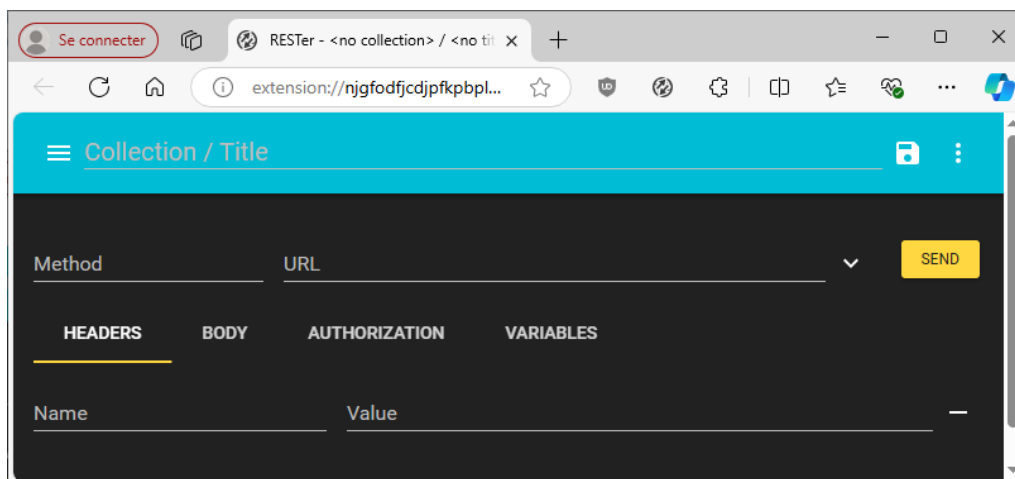
En fait, c'est la seule information permanente qu'il faudra connaître dans ce qui suit et il sera donc inutile de recommencer à nouveau les étapes précédentes à l'avenir.

3 Accès à l'API avec un navigateur Internet

La documentation de l'API¹¹, décrit la démarche à utiliser pour récupérer la couleur Tempo d'un ou plusieurs jours :

- Demander un Jeton d'accès à durée limitée (2h)
- Utiliser ce jeton pour demander la couleur d'un ou plusieurs jours

Dans les 2 cas, il suffit de formuler une **requête HTTP GET**, avec une autorisation adéquate. Ceci peut se faire avec un navigateur Internet, doté d'une extension had hoc ; dans le cas de Microsoft **Edge**, on peut utiliser l'extension **RESTer**¹² :



3.1 Obtention d'un jeton d'accès

Pour demander un jeton d'accès à durée limitée, il suffit de créer une requête **GET** avec les paramètres suivants :




- URL : <https://digital.iservices.rte-france.com/token/oauth/>
- Header :

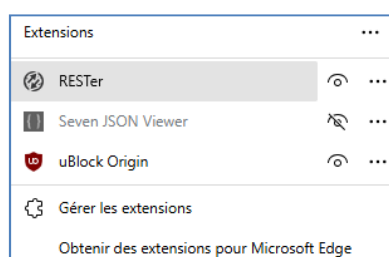
Accept=application/json

Authorization=Basic NTA-----A==

(codage en base 64 de l'autorisation précédemment obtenue dans l'**Application**)

¹¹ https://data.rte-france.com/documents/20182/224298/FR_GU_API_Tempo_Like_Supply_Contract_v01.02.pdf

¹² On peut l'ajouter avec la commande **.../Extensions** et, la lancer en cliquant sur son icône  après l'avoir ajoutée à la barre d'outils d'Edge. Pour l'ajouter à la barre d'outils, il suffit d'ajouter l'icône des Extensions  à cette barre (avec la commande **.../Paramètres/Apparence/Personnaliser la barre d'outils/Sélectionner les boutons à afficher dans la barre d'outils**) puis, en cliquant dessus, d'activer la visibilité de **RESTER** dans cette barre  :



Requests / Token

Method: GET URL: https://digital.iservices.rte-france.com/token/oauth/ [SEND]

HEADERS BODY AUTHORIZATION VARIABLES

Name	Value
Accept	application/json
Authorization	Basic NTA [redacted]

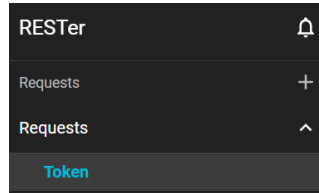
En cliquant sur **SEND**, on obtient la réponse suivante du serveur, en format JSON, contenant le jeton recherché :

Response **200 OK**

```
{
  "access_token": "dtYCdWtoLPhCG0fZLjA3PyQtsAXeOo62aX0ydJ1g84oySw2ame6fID",
  "token_type": "Bearer",
  "expires_in": 7200
}
```

Remarques

- Comme on pourra être conduit à reformuler cette requête quand le jeton obtenu sera obsolète, on aura intérêt à l'enregistrer dans **Requests/Token** :



- On aurait pu également utiliser directement les valeurs de **ID Client** et **ID Secret** obtenues lors de la création de l'application **Tempo** : il suffit de définir une **AUTHORIZATION** de type **Basic**, en passant ces valeurs dans les champs à renseigner :
 - User name**=508-----6c3
 - Password**= c35-----bc0

(ID Client et ID Secret de l'Application)

On crée ainsi un jeton d'accès permanent de type **Basic** que l'on sauvegarde ici sous le nom **Token** :

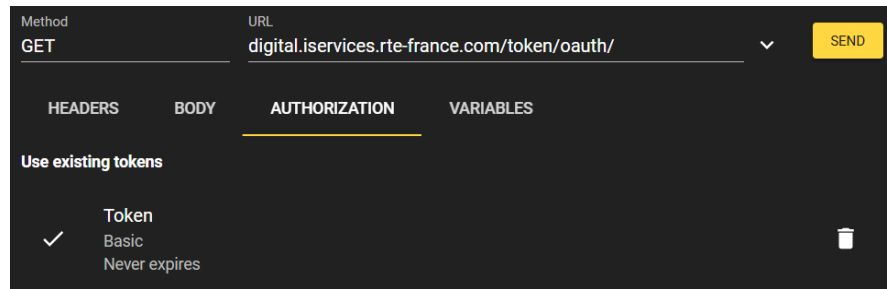
Title: Token

User name: 508

Password: c35

CANCEL SAVE

Pour l'utiliser dans la requête, il suffit de la sélectionner :



En fait cette technique conduit à la même requête que précédemment car, utiliser le jeton **Token**, revient à ajouter la ligne suivante dans les **HEADERS** :



3.2 Obtention de la couleur d'un ou plusieurs jours

D'après la documentation de l'API **Tempo Like Supply Contract**¹³, pour obtenir la couleur Tempo d'un ou plusieurs jours, il suffit de formuler une requête **GET**, avec les paramètres suivants :

- URL : https://digital.iservices.rte-france.com/open_api/tempo_like_supply_contract/v1/tempo_like_calendars?start_date=<Date début>&end_date=<Date fin>
- Header :
Accept=application/json
Authorization=Bearer <Jeton d'accès>

Les dates doivent être écrites en format ISO 8601¹⁴ :

YYYY-MM-DDTHH:MM:SS±hh:mm

où **HH:MM:SS** est l'**heure locale** et **±hh:mm** précise son **décalage**, par rapport à l'**heure UTC**, dû au fuseau horaire et à l'heure d'été ; par exemple, dans le cas de la zone de Paris, on aura :

- **±hh:mm** = +02 :00 en heure d'été
- **±hh:mm** = +01 :00 en heure d'hiver

Quand on a bien compris cette complication qui paraît inutile¹⁵ car les dates à fournir sont toujours de la forme :

YYYY-MM-DDT00:00 00±hh:mm

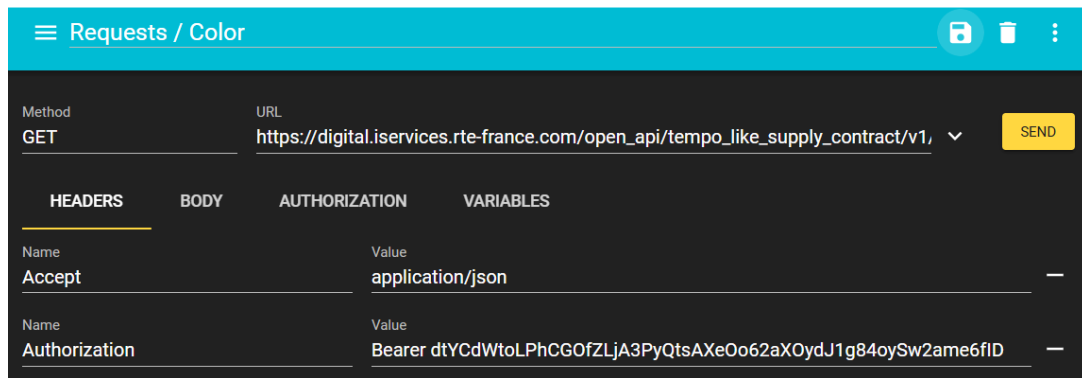
on peut créer la requête HTTP permettant d'obtenir les couleur Tempo des jours J-1, J et J+1 le 12 février 2024 à Nice (fuseau horaire de Paris et heure d'hiver donc), en utilisant le jeton d'accès précédemment obtenu et l'URL suivante :

https://digital.iservices.rte-france.com/open_api/tempo_like_supply_contract/v1/tempo_like_calendars?start_date=2024-02-11T00:00:00+01:00&end_date=2024-02-14T00:00:00+01:00

¹³ https://data.rte-france.com/documents/20182/224298/FR_GU_API_Tempo_Like_Supply_Contract_v01.02.pdf

¹⁴ https://fr.wikipedia.org/wiki/ISO_8601

¹⁵ Les temps qui interviennent ici ne sont jamais ambigus si on ne précise que leur date. Cf. <https://github.com/jlemaire06/Times-and-ESP32> pour une justification.



qui conduit à la réponse du §3.1 :

Response **200 OK**

```
{
  "tempo_like_calendars": [
    {
      "start_date": "2024-02-11T00:00:00+01:00",
      "end_date": "2024-02-14T00:00:00+01:00",
      "values": [
        {
          "start_date": "2024-02-13T00:00:00+01:00",
          "end_date": "2024-02-14T00:00:00+01:00",
          "value": "WHITE",
          "updated_date": "2024-02-12T10:20:00+01:00"
        },
        {
          "start_date": "2024-02-12T00:00:00+01:00",
          "end_date": "2024-02-13T00:00:00+01:00",
          "value": "RED",
          "updated_date": "2024-02-11T10:20:00+01:00"
        },
        {
          "start_date": "2024-02-11T00:00:00+01:00",
          "end_date": "2024-02-12T00:00:00+01:00",
          "value": "BLUE",
          "updated_date": "2024-02-10T10:20:00+01:00"
        }
      ]
    }
  ]
}
```

Remarques

- Si on demande la couleur Tempo d'un jour, non encore déterminée par RTE, l'API retournera une erreur. Par exemple, voici ce qu'on obtient quand on demande la couleur Tempo du 20 octobre 2024, le 19 octobre 2024 à 9h du matin en France métropolitaine, en utilisant l'URL suivante :

https://digital.iservices.rte-france.com/open_api/tempo_like_supply_contract/v1/tempo_like_calendars?start_date=2024-10-20T00:00:00+02:00&end_date=2024-10-21T00:00:00+01:00

Response **400 Bad Request**

```
{
  "error": "TMPLIKSUPCON_TMPLIKCAL_F04",
  "error_description": "The value of \"end_date\" field is incorrect. It is not possible to recover data to this term.",
  "error_uri": "",
  "error_details": {
    "transaction_id": "Id-1166136759b284c0a978d95b"
  }
}
```

On détectera cette situation à l'aide du code HTTP retourné (400 ici).

- On peut s'interroger sur la nécessité de l'indication du temps dans les 2 dates à renseigner. Par exemple, si on demande la couleur Tempo du 12 février 2024, entre 0h et 5h59m59s, avec l'URL suivante :

https://digital.iservices.rte-france.com/open_api/tempo_like_supply_contract/v1/tempo_like_calendars?start_date=2024-02-12T00:00:00+01:00&end_date=2024-02-12T05:59:59+01:00

on obtient là encore une erreur :

```
Response 400 Bad Request
...
{
  "error": "TMPLIKSUPCON_TMPLIKCAL_F05",
  "error_description": "The period filled by fields \"start_date\" and \"end_date\" is too short to return values. Please check the user guide to verify the minimum period for this API.",
  "error_uri": "",
  "error_details": {
    "transaction_id": "Id-d1281267753578fb061c1bb0"
  }
}
```

Comme le code HTTP retourné est identique au précédent, il faudra veiller à bien écrire les dates, comme dans l'exemple précité.

4 Accès à l'API avec un microcontrôleur

4.1 Carte ESP32 DevkitC V4¹⁶

Basée ici sur le module **Espressif ESP32-WROOM-32UE**, on peut y connecter une antenne externe 2,4GHz pour améliorer la qualité de la transmission Wifi :



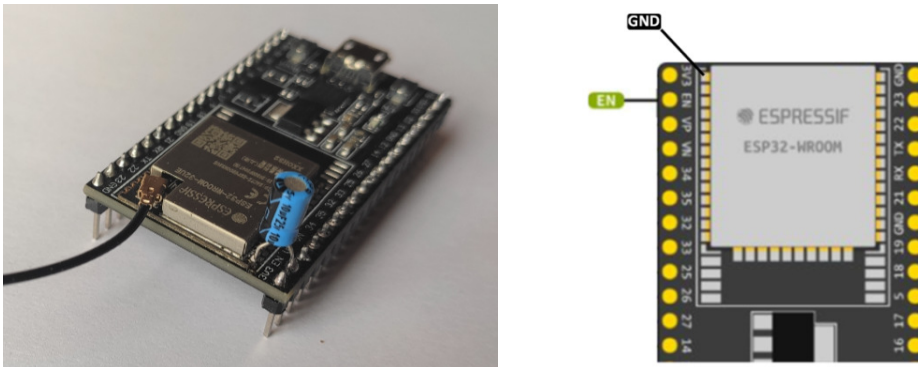
Elle a été programmée avec un PC sous Windows 10, en utilisant l'extension **PlatformIO** de **Visual Studio Code**¹⁷.

Pour y transférer un programme, il suffit de la connecter au PC via son port micro-USB, qui permet également de l'alimenter¹⁸.

¹⁶ <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html#get-started-esp32-devkitc-board-front>

¹⁷ <https://docs.platformio.org/en/stable/integration/ide/vscode.html>

Par contre, ce transfert n'est possible que si la carte est en mode « bootloader » et il faut a priori maintenir enfoncé le bouton « Boot » pendant tout le transfert pour installer ce mode. On peut éviter cela en soudant un condensateur électrolytique de 10µF entre la broche EN de la carte et la patte GND de son module Espressif¹⁹ :



4.2 Programme en C++

Dans **PlatformIO**, on commence par créer un nouveau projet, **Tempo**, adapté au microcontrôleur précédent et incluant la bibliothèque **ArduinoJson**²⁰ pour décoder les réponses du serveur en format JSON ; ceci crée le fichier **platformio.ini**, où on a précisé le port série à utiliser :

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200
upload_port = COM3
lib_deps = bblanchon/ArduinoJson@^7.4.2
```

Ensuite, on définit comme suit le fichier **main.cpp** où il faudra préciser les constantes **SSID**, **PWD** et **BASIC_AUTH** :

```
// Tempo - main.cpp (for PlatformIO)
/*****

Objective
    Obtain Tempo color for a day, using the RTE API Tempo Like Supply Contract.

Steps
1. Connect to a local network using a Wifi Access Point.
2. Send an HTTP GET Request to the RTE API, to obtain an access Token.
3. Init the time system with a time zone string, to handle local times.
4. If the current local time is required, call an NTP server to init the RTC clock.
5. Send HTTP GET requests to the RTE API, to obtain the Tempo colors.

NB
- In steps 2 and 5, decode the JSON data sent by the RTE API.
- The results are displayed on the serial monitor.
- Use robust Wifi connection :
    . restart the ESP32 when the connections don't succeed ;
    . automatic Wifi reconnection when accidentally lost.
- If required, use a robust connection to the NTP server :
    . restart the ESP32 when the connections don't succeed.

References
- ESP32-DevKitC V4 and ESP32-WROOM-32UE :
```

¹⁸ Outre l'utilisation du port USB, on peut aussi l'alimenter en utilisant ses broches 5V et GND ou bien 3V3 et GND, mais à condition de ne choisir qu'une seule de ces 3 possibilités.

¹⁹ <https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/>

²⁰ <https://arduinojson.org/v7/>

```

    . https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html
    . https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf
    . https://randomnerdtutorials.com/solved-failed-to-connect-to-esp32-timed-out-waiting-for-packet-header/
- Wifi :
    . https://randomnerdtutorials.com/esp32-useful-wi-fi-functions-arduino/
    . https://randomnerdtutorials.com/solved-reconnect-esp32-to-wifi/
- Current time :
    . https://randomnerdtutorials.com/esp32-date-time-ntp-client-server-arduino/
    . https://randomnerdtutorials.com/esp32-ntp-timezones-daylight-saving/
    . https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv
    . https://sourceware.org/newlib/libc.html#Timefn
    . https://cplusplus.com/reference/ctime/
    . https://github.com/espressif/arduino-esp32/blob/master/cores/esp32/esp32-hal-time.c#L47
- HTTPClient :
    . https://github.com/espressif/arduino-esp32/tree/master/libraries/HTTPClient/src
    . https://randomnerdtutorials.com/esp32-http-get-post-arduino/
- ArduinoJSON :
    . https://github.com/bblanchon/ArduinoJson
    . https://arduinojson.org/v7/doc/
    . https://arduinojson.org/v7/assistant/
    . https://arduinojson.org/v7/how-to/use-arduinojson-with-httpclient/
- Watch dog timer (WDT)
    . https://iotassistant.io/esp32/enable-hardware-watchdog-timer-esp32-arduino-ide/

/*****
Libraries and types
*****/

#include <WiFi.h>
#include <time.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <esp_task_wdt.h>

/*****
Constants
*****/

// Local network access point
const char *WIFI_SSID = ".....";
const char *WIFI_PASSWORD = ".....";
const uint32_t WIFI_TIMEOUT = 20; // s

// NTP server (=>UTC time) and Time zone
const char* NTP_SERVER = "pool.ntp.org"; // Server address (or "ntp.obspm.fr", "ntp.unice.fr", ...)
const char* TIME_ZONE = "CET-1CEST,M3.5.0,M10.5.0/3"; // Europe/Paris time zone
const uint32_t NTP_TIMEOUT = 20; // s

// HTTP requests timeout
const uint16_t HTTP_TIMEOUT = 10000; // ms

// RTE API Tempo Like Supply Contract
const char *TOKEN_URL = "https://digital.iservices.rte-france.com/token/oauth/";
const char *BASIC_AUTH = "Basic .....";
const char *TEMPO_URL = "https://digital.iservices.rte-france.com/open_api/tempo_like_supply_contract/v1/tempo_like_calendars?start_date=YYYY-MM-DDThh:mm:sszzzz&end_date=YYYY-MM-DDThh:mm:sszzzz";
const char *UNDEFINED = "UNDEFINED"; // Undefined color

// Print flag
// #define PRINT_FLAG

/*****
Tool functions
*****/

void initWifi(const char *ssid, const char *password, const uint32_t timeOut)
{
    esp_task_wdt_init(timeOut, true); // Enable panic so ESP32 restarts
    esp_task_wdt_add(NULL); // Add current thread to WDT watch
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) continue;
    esp_task_wdt_delete(NULL); // Delete the WDT for the current thread
#ifdef PRINT_FLAG
    Serial.printf("Wifi connected : IP=%s RSSI=%d\n", WiFi.localIP().toString(), WiFi.RSSI());

```

```

#endif
}

void WiFiDisconnected(WiFiEvent_t event, WiFiEventInfo_t info)
{
#ifdef PRINT_FLAG
    Serial.printf("Wifi disconnected : Reason=%d\n", info.wifi_sta_disconnected.reason);
#endif
    if (info.wifi_sta_disconnected.reason != 8) // Not a call to Wifi.disconnect()
    {
        initWiFi(WIFI_SSID, WIFI_PASSWORD, WIFI_TIMEOUT);
    }
}

void setTimeZone(const char *timeZone)
{
    // To work with Local time (custom and RTC)
    setenv("TZ", timeZone, 1);
    tzset();
}

void initRTC(const char *NTPServer, const char *timeZone, const uint32_t timeOut)
{
    // Set RTC with Local time, using an NTP server
    esp_task_wdt_init(timeOut, true); // Enable panic so ESP32 restarts
    esp_task_wdt_add(NULL);           // Add current thread to WDT watch
    configTime(0, 0, NTPServer);       // To get UTC time
    tm time;
    while (!getLocalTime(&time)) continue; // UTC time
    setTimeZone(timeZone);               // Transform to Local time
    esp_task_wdt_delete(NULL);           // Delete the WDT for the current thread
#ifdef PRINT_FLAG
    Serial.println("RTC clock initialized with Local time, using an NTP server");
#endif
}

bool getCustomTime(int year, int month, int day, int hour, int minute, int second, tm *timePtr)
{
    // Set a time (date) without DST indication
    *timePtr = {0};
    timePtr->tm_year = year - 1900;
    timePtr->tm_mon = month - 1;
    timePtr->tm_mday = day;
    timePtr->tm_hour = hour;
    timePtr->tm_min = minute;
    timePtr->tm_sec = second;
    time_t t = mktime(timePtr);
    timePtr->tm_hour--;
    time_t t1 = mktime(timePtr);
    memcpy(timePtr, localtime(&t), sizeof(tm));
    if (localtime(&t1)->tm_isdst==1)
    {
        if (timePtr->tm_isdst==0) return false; // Ambiguous
        else timePtr->tm_hour--;
    }
    return true;
}

bool getJsonDocumentFromHTTPRequest(const char *url, const char *auth, JsonDocument &doc)
{
    bool okJson = false;
    HTTPClient http;
    http.useHTTP10(true); // To prevent chunked transfer encoding
#ifdef PRINT_FLAG
    Serial.printf("Url : %s\n", url);
#endif
    http.setTimeout(HTTP_TIMEOUT);
    http.begin(url); // http request
    http.addHeader("Accept", "application/json");
    if (auth != NULL)
    {
        http.addHeader("Authorization", auth);
    }
#ifdef PRINT_FLAG
    Serial.printf("Authorization : %s\n", auth);
#endif
    int httpCode = http.GET();
    if (httpCode > 0)
    {

```

```

    if (httpCode == HTTP_CODE_OK)
    {
        DeserializationError errorJson = deserializeJson(doc, http.getStream());
        if (!errorJson) okJson = true;
        else
        {
#ifdef PRINT_FLAG
            Serial.printf("Json error : %s\n", errorJson.c_str());
#endif
        }
    }
    else
    {
#ifdef PRINT_FLAG
        Serial.printf("HTTP code : %d\n", httpCode);
#endif
    }
    else
    {
#ifdef PRINT_FLAG
        Serial.printf("HTTP GET failed, error: %s\n", http.errorToString(httpCode).c_str());
#endif
    }
    http.end();
    return okJson;
}

const char* getTempoDayColor(const int year, const int month, const int day, const char *auth)
{
    // Copy dates in URL with ISO 8601 format
    tm timeStart; getCustomTime(year, month, day, 0, 0, 0, &timeStart);
    tm timeEnd; getCustomTime(year, month, day+1, 0, 0, 0, &timeEnd);
    char url[200];
    strcpy(url, TEMPO_URL);
    strftime(url+112, 23, "%FT%T%Z", &timeStart);
    strftime(url+147, 23, "%FT%T%Z", &timeEnd);
    strcpy(url+134, ":00");
    strcpy(url+169, ":00");
    url[137] = '&';
    JsonDocument doc;
    if (getJsonDocumentFromHttpRequest(url, auth, doc))
    {
        return (doc["tempo_like_calendars"]["values"][0]["value"] | UNDEFINED);
    }
    return UNDEFINED;
}

/*****
    setup and loop functions
*****/

void setup()
{
    // Open serial port
    Serial.begin(115200);
    while (!Serial) continue;
    delay(200);

    // Connect to the Wifi access point
    WiFi.onEvent(WiFiDisconnected, WiFiEvent_t::ARDUINO_EVENT_WIFI_STA_DISCONNECTED);
    initWifi(WIFI_SSID, WIFI_PASSWORD, WIFI_TIMEOUT);
    Serial.println("\nACCESS TO THE RTE API \"TEMPO LIKE SUPPLY CONTRACT\"");

    // Access token
    Serial.println("\nGET ACCESS TOKEN");
    bool okToken = false;
    char auth[100];
    {
        JsonDocument doc;
        if (getJsonDocumentFromHttpRequest(TOKEN_URL, BASIC_AUTH, doc))
        {
            Serial.printf("Token : %s\n", (const char *)doc["access_token"]);
            strcpy(auth, "Bearer ");
            strcpy(auth+7, doc["access_token"]);
            okToken = true;
        }
    }
}

```

```

// Days color
if (okToken)
{
    // Set time zone (not necessary after InitRTC)
    setTimeZone(TIME_ZONE);

    // Custom Tempo days color
    Serial.println("\nCUSTOM DAY TEMPO COLOR");
    Serial.println("12/2/2024 :");
    Serial.printf("Day J-1 Tempo color : %s\n", getTempoDayColor(2024, 2, 11, auth));
    Serial.printf("Day J Tempo color : %s\n", getTempoDayColor(2024, 2, 12, auth));
    Serial.printf("Day J+1 Tempo color : %s\n", getTempoDayColor(2024, 2, 13, auth));

    // Current Tempo day color
    Serial.println("\nCURRENT DAY TEMPO COLOR");
    initRTC(NTP_SERVER, TIME_ZONE, NTP_TIMEOUT); // Init the RTC with Local time, using an NTP server
    tm t;
    getLocalTime(&t);
    Serial.printf("%d/%d/%d : \n", t.tm_mday, t.tm_mon+1, t.tm_year+1900);
    Serial.printf("Day J Tempo color : %s\n", getTempoDayColor(t.tm_year+1900, t.tm_mon+1, t.tm_mday ,
auth));
    Serial.printf("Day J+2 Tempo color : %s\n", getTempoDayColor(t.tm_year+1900, t.tm_mon+1,
t.tm_mday+2 , auth));
}
else Serial.println("Error : cannot obtain access token");
}

void loop()
{
}

```

Sortie obtenue :

```

ACCESS TO THE RTE API "TEMPO LIKE SUPPLY CONTRACT"

GET ACCESS TOKEN
Token : aL9GHxd8o403rcYChEYHWH4cr28IMm59j1Lw8pu7vVGNEWDcGmh04J

GET CUSTOM DAY TEMPO COLOR
12/2/2024 :
Day J-1 Tempo color : BLUE
Day J Tempo color : RED
Day J+1 Tempo color : WHITE

GET CURRENT DAY TEMPO COLOR
23/7/2025 :
Day J Tempo color : BLUE
Day J+2 Tempo color : UNDEFINED

```

Explications

- Classiquement, l'objet **Wifi** est utilisé pour établir une connexion en WIFI, avec une box permettant d'accéder à Internet.
- Le jeton d'accès pour l'API RTE **Tempo Like Supply Contract** est obtenu avec une requête **GET**, en utilisant un objet **http**, instance de **HTTPClient**, défini dans la fonction **getJSONDocumentFromHTTPRequest**.
- Pour pouvoir manipuler des temps locaux, avec indication automatique de leurs décalages par rapport aux temps UTC, il suffit de préciser les paramètres de la **zone de temps** - celle de **Europe/Paris** dans notre cas - qui sont codés dans la chaîne de caractères²¹ **CET-1CEST,M3.5.0,M10.5.0/3** ; ceci est fait dans la fonction **setTimeZone**. On pourra dès lors utiliser la fonction **getCustomTime** qui remplit une structure de type **tm**, en renseignant le champ **tm_isdst**²². On le met à profit dans la fonction **getTempoDayColor** qui ne requiert que

²¹ https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv

²² <https://github.com/jlemaire06/Times-and-ESP32>

l'année, le mois et le jour comme paramètres caractérisant la date du jour dont on recherche la couleur Tempo.

- Enfin, pour obtenir la couleur Tempo du jour courant, on utilise l'horloge interne RTC de l'ESP32 qui est initialisée dans la fonction `initRTC`, à partir du temps UTC fourni par un serveur NTP (**pool.ntp.org** ici), transformé en temps local avec mention du décalage horaire, là-encore grâce au mécanisme mis en place par `setTimeZone`. Il suffit dès lors de faire appel à `getLocalTime` (sans appeler cette fois un serveur NTP), pour obtenir le temps local courant, là encore dans une instance de la structure `tm`. En extrayant l'année, le mois et le jour, on pourra, comme précédemment, faire appel à `getTempoDayColor`.
- Dans tous les cas la fonction clé est `getTempoDayColor` qui a été volontairement limitée, pour simplifier, à ne fournir que la couleur Tempo d'un seul jour. Mais vu les possibilités de l'API **Tempo Like Supply Contract**, on aurait pu évidemment l'étendre pour fournir les couleurs Tempo d'une **succession de jours**, en retournant cette fois un tableau de chaînes de caractères.
- La version de 7 de la bibliothèque ArduinoJson est ici utilisée. Elle simplifie considérablement le décodage de la réponse d'un serveur http : pour chaque requête à ce serveur, il suffit de définir au préalable une instance²³ `doc` de la classe `JsonDocument`, puis de décoder le flux réponse du serveur avec une instruction telle que :

```
deserializeJson(doc, http.getStream())
```

à condition toutefois de pouvoir utiliser la version 1.0 du protocole http, pour que ce flux ne soit pas segmenté ; c'est le cas ici et on le met en œuvre dans la fonction suivante :

```
bool getJsonDocumentFromHttpRequest(const char *url, const char *auth, JsonDocument &doc)
```

où le document est passé en référence.

5 Conclusion

L'option Tempo, proposée aux clients d'EDF, permet de faire des économies substantielles dans la consommation d'électricité, mais à la condition de bien connaître la tarification en cours, qui dépend de la couleur Tempo du jour et de l'heure (creuse ou pleine).

Bien qu'un peu rébarbative a priori, l'API RTE Tempo Like Supply Contract peut fournir la couleur Tempo de n'importe quel jour, dès lors qu'on fait l'effort de comprendre son mécanisme à 2 niveaux (jeton d'accès, couleur du jour) et surtout d'apprendre à manipuler les heures locales en C++, en prenant en compte le décalage horaire (fuseau horaire + heure d'été).

Tout ceci peut être alors utilisé dans le système de routage d'une installation photovoltaïque par exemple.

6 Addendum

Lors d'une récente discussion sur le **Forum Photovoltaïque**²⁴, dans la rubrique **Avis option TEMPO de EDF**, **pjeantaud** m'a aimablement indiqué une autre possibilité²⁵ pour connaître la couleur des jours Tempo d'une « saison » EDF : appeler la page Web suivante, apparemment non documentée sur le site des services de RTE :

²³ Implémentée sur le tas.

²⁴ <https://forum-photovoltaique.fr>

²⁵ <https://forum-photovoltaique.fr/viewtopic.php?p=797045#p797045>

https://www.services-rte.com/cms/open_data/v1/tempo?season=YYYY-YYYY

Par exemple, l'URL :

https://www.services-rte.com/cms/open_data/v1/tempo?season=2023-2024

fournit la couleur Tempo de tous les jours de l'année 2023 et 2024 jusqu'en août, là encore dans un format JSON :

```
{
  "values": {
    ...
    "2024-02-11": "BLUE",
    ...
    "2024-02-12": "RED",
    ...
    "2024-02-13": "WHITE",
    ...
  }
}
```

Et , le 23/07/2025 à 15h20, pour obtenir la couleur Tempo du jour précédent, du jour courant et du lendemain, il suffit d'appeler dans un navigateur :

https://www.services-rte.com/cms/open_data/v1/tempo?season=2024-2025

Elle répond :

```
{
  "values": {
    ...
    "2025-07-22": "BLUE",
    ...
    "2025-07-23": "BLUE",
    ...
    "2025-07-24": "BLUE",
    ...
  }
}
```

A noter ici qu'il n'y a pas ici d'enregistrement avec la date "2024-07-25", ce qui revient à dire que la couleur du 25/07/2025 est "UNDEFINED".

C'est évidemment beaucoup plus simple qu'en utilisant l'API RTE **Tempo Like Supply Contract**, mais son absence de documentation sur le site des services de API RTE peut laisser supposer un doute sur sa pérennité.

Quoiqu'il en soit, la démarche sera la même, pour réaliser un appel : requête GET http **sans autorisation nécessaire**, décodage JSON de la réponse, etc.

Il existe également maintenant une autre possibilité d'accès à la couleur d'un Tempo EDF, en utilisant cette fois une API privée. Nous l'avons découvert, suite à un blocage de notre compte sur <https://data.rte-france.com/> résultant d'une fausse manip de notre part, réparé depuis par la hotline²⁶ qui répond très rapidement ! Cette API est décrite dans la page <https://www.api-couleur-tempo.fr/api> et on peut, par exemple, l'appeler avec une requête GET sur l'URL suivante :

<https://www.api-couleur-tempo.fr/api/jourTempo/YYYY-DD-JJ>

²⁶ rte-hotline@rte-france.com

Par exemple, avec **RESTer** :

Method	URL	
GET	https://www.api-couleur-tempo.fr/api/jourTempo/2025-07-23	SEND

conduit à la réponse suivante²⁷ :

```
{
  "dateJour": "2025-07-23",
  "codeJour": 1,
  "periode": "2024-2025"
}
```

Le code de cette API est fourni sans explication sur Github²⁸, mais l'indication de la « période » incite à penser que la méthode précédente a été ici utilisée.

Bien que très simple, cette solution reste néanmoins non officielle et, là encore, on peut s'interroger sur sa pérennité.

Savoir utiliser l'API RTE Tempo Like Supply Contract n'est donc pas inutile et c'est bien ce que l'on observe sur d'autres forums, par exemple sur celui-ci concernant Home Assistant²⁹.

²⁷ 0 => UNDEFINED, 1 => BLUE, 2 => WHITE, 3 => RED.

²⁸ <https://github.com/jbromain/rte-tempo>

²⁹ <https://axellefa.fr/2025/02/14/edf-tempo-afficher-les-informations/>