Cloc 2.0 : le réveil-matin de vos rêves

Yves Bourdon (France)

Avec le récepteur radio, l'horloge est probablement l'un des thèmes favoris des *elektorniciens* amateurs. Je m'attends à ce que tout lecteur de cet article ait construit au moins une horloge. Personnellement, j'en ai construit beaucoup. La plupart faisaient aussi réveil. Je sais donc bien ce que j'en attends. Mon réveil idéal n'étant pas commercialisé, je l'ai construit moi-même. Voici donc le *Cloc 2.0*.

Réalisée en 1971, ma première horloge utilisait des tubes *Nixie* récupérés sur un ordinateur *IBM* mis au rebut. Le circuit imprimé accueillait un nombre considérable de composants. La précision était d'environ dix secondes par jour. Par la suite, j'ai conçu et réalisé de nombreuses horloges dont les plus célèbres d'Elektor.

Mon horloge Nixie faisait aussi réveil, fonction que j'ai beaucoup développée au fil des années en simplifiant au mieux l'interface homme-machine. Animé par un PIC32, mon dernier réveil était doté de deux écrans (**figure 1**). Un module *ESP8266* exploitait un serveur de temps en ligne via *NTP* pour régler l'heure automatiquement. Le PIC32 « pressait » les boutons d'une télécommande ***authentique*** pour allumer la radio et diffuser ma station préférée ou de la musique. Un encodeur rotatif et deux poussoirs servaient à régler le réveil. Décrit ci-dessous, le projet Cloc 2.0 est basé sur celui-ci.

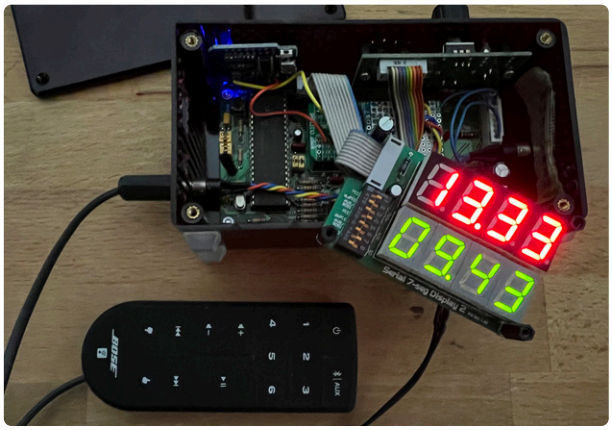


Figure 1. Cloc 1.0, la version précédant Cloc 2.0. Basé sur un PIC32, elle utilisait une authentique télécommande comme sortie IR.

Figure 2. Outre l'heure et la date en cours, la page principale affiche, une vue d'ensemble des paramètres importants.

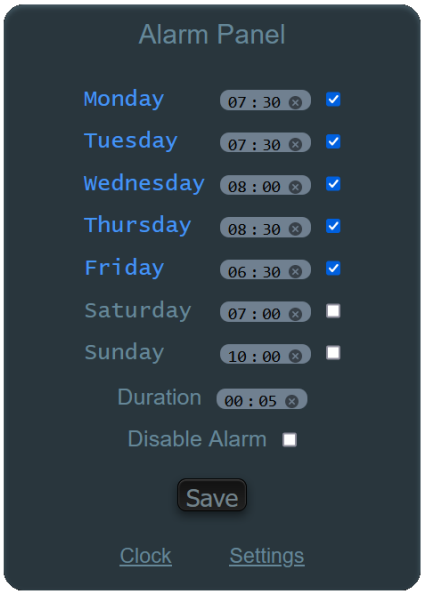


Figure 3. La page Alarme permet de définir l'heure de réveil pour chaque jour de la semaine.

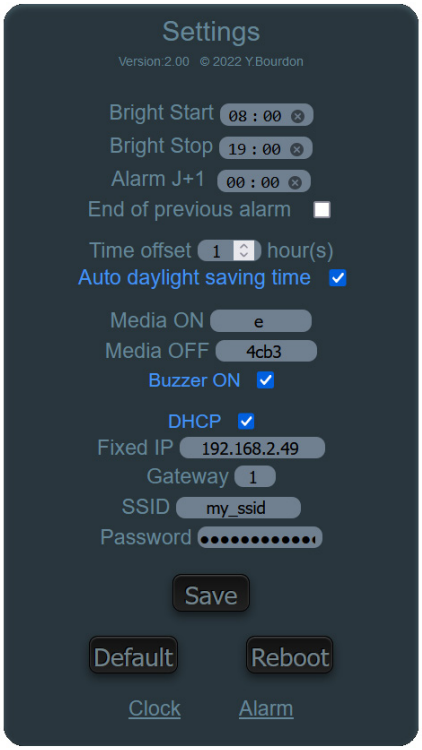


Figure 4. Tous les paramètres configurables par l'utilisateur sont sur la page Paramètres.

Tout pour séduire : beauté et simplicité

Je voulais un réveil facile à construire, sans composants CMS et s'intégrant le mieux possible dans un boîtier standard du commerce. Je voulais aussi garder la touche rétro des afficheurs 7 segments dont la luminosité variable autorise une très bonne lisibilité la nuit, sans gêner quand on aime dormir dans le noir.

L'interface Web

Cloc se connecte à un réseau wifi, il peut donc accéder à un serveur de temps existant. Il se connecte avec une adresse IP fixe ou dynamique (*DHCP*). Une fois connecté, l'accès au serveur web permet de savoir l'heure et de régler les alarmes et paramètres du réveil. L'interface graphique *HTML* optimisée est compatible avec la plupart des navigateurs et des appareils portables. L'interface comprend trois pages : Principale, Alarme et Paramètres.

Page principale

Cette page (**figure 2**) affiche l'heure et la date en cours, ainsi que l'heure de début et de fin de l'alarme à venir. Elle affiche aussi le SSID du réseau wifi auquel il est connecté et le signal *RSSI* reçu. La couleur de ce dernier (vert, orange ou rouge) symbolise sa valeur. La page principale est rafraîchie à 1 Hz environ.

Page d'alarme

La page d'alarme (**figure 3**) permet de régler les alarmes pour chaque jour de la semaine. Décocher un jour de la semaine désactive l'alarme correspondante. On peut également désactiver toutes les alarmes d'un seul clic.

La durée de l'alarme est également réglable. Un début d'alarme lance la commande `Media ON` et l'envoie à la *LED IR*. Une fin d'alarme lance la commande `Media OFF` et l'envoie à la LED IR. L'alarme peut en outre faire retentir le buzzer (v. la page Paramètres).

Sur modification d'un paramètre d'alarme, un bref signal sonore est émis. Un paramètre modifié est sauvegardé dans l'EEPROM.

Page de configuration

Les paramètres ci-après sont réglables (voir **figure 4**).

- Période d'affichage à haute luminosité (`Bright Start` et `Bright Stop`).
- Le moment auquel s'affiche l'heure de la prochaine alarme (*Alarme J+1*), par ex. le soir quand on va se coucher ou bien tout de suite après la dernière alarme (cocher Fin de l'alarme précédente).
- Heure d'été, automatique ou manuelle.
- Code IR `Media ON` et `Media OFF` pour contrôler un autre appareil (par ex. une radio).
- Activer/désactiver le buzzer.
- L'identification du réseau wifi utilisant soit le DHCP, soit une adresse IP fixe.

Boutons-poussoirs

Cloc a deux boutons-poussoirs d'interaction avec l'utilisateur : S1 (désactivation) et S2 (arrêt). En fonctionnement normal, S1 active/désactive l'alarme (allume/éteint l'affichage du bas). L'interface web dispose aussi de cette fonction. En appuyant sur S1 à la mise sous tension de Cloc, on restaure les valeurs par défaut définies dans le programme (pensez à les adapter à vos propres besoins avant de programmer l'horloge) C'est à faire à la toute première mise sous tension de l'horloge.

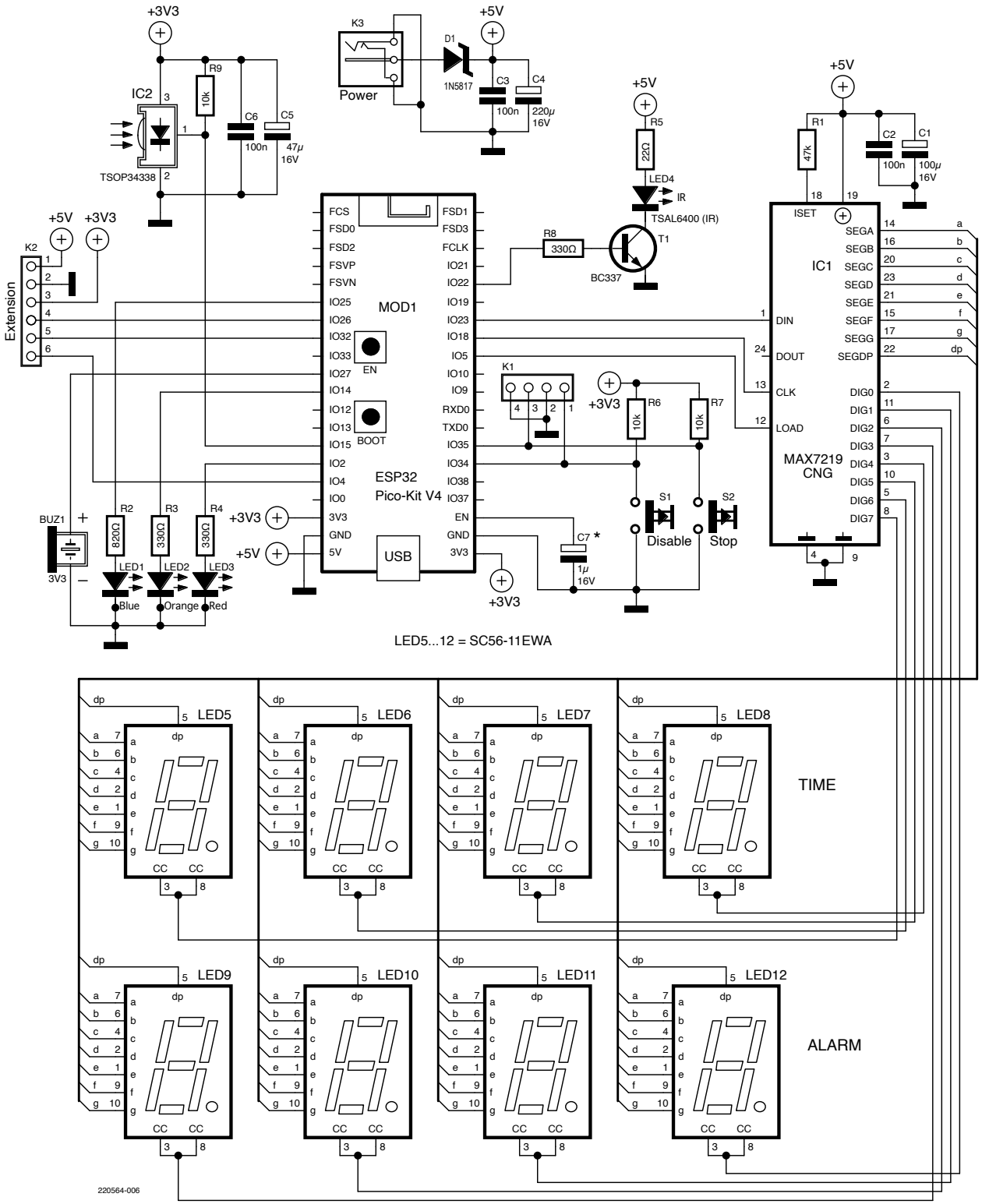


Figure 5. L'ESP32-PICO-KIT fournit la connectivité Internet et l'heure ; le MAX7219 contrôle huit afficheurs 7 segments répartis en deux rangées. Notez l'interface IR autour d'IC2 (entrée) et de LED4 (sortie).

Si une alarme a été déclenchée en mode normal, un appui sur S2 l'arrête immédiatement et l'affichage du bas cesse de clignoter. Un appui sur S2 quand aucune alarme ne retentit, envoie le code IR de mise en marche/arrêt à l'appareil de votre choix. L'horloge se comporte alors comme une télécommande.

Un appui sur S2 pendant la mise sous tension, fait passer l'horloge en mode point d'accès (AP) à l'adresse 192.168.4.1. Vous pouvez vous y connecter via tout appareil (téléphone portable, tablette ou ordinateur) pour accéder à l'interface web de configuration des paramètres de l'horloge.

Description du circuit

La **figure 5** donne le schéma du Cloc 2.0. Les deux afficheurs 7 segments et leur commande IC1 en occupent près des deux tiers. Pour piloter les huit chiffres, quatre par affichage (heure et alarme), j'ai choisi le MAX7219. Il exploite un bus SPI à 2 fils (le DIN n'est pas utilisé) et se programme facilement (bibliothèques open-source). La résistance R1 règle la luminosité max. de l'affichage (plus R1 est élevée, moins l'affichage brille). Notez que les afficheurs à haute luminosité sont vraiment lumineux, peut-être trop pour un réveil.

Pour Cloc 2.0, j'ai choisi le module *ESP32-PICO-KIT* qui offre de nombreux ports d'E/S et qui a les dimensions parfaites pour ce projet. Le condensateur C7 n'est utile que si le module ESP32 reste en mode *reset* après la mise sous tension. Des modules ESP32 (anciens ?) souffriraient de ce problème que je n'ai jamais observé. Si vous décidez de monter C7, couchez-le orienté vers l'opposé de R6 et R7.

Le buzzer d'alarme BUZ1 (sans oscillateur interne) est piloté en MLI par le port IO27 de l'ESP32.

Les ports IO34/35 de l'ESP32 sont reliés aux boutons-poussoirs S1 et S2, et au connecteur K1. Dans une réalisation type, S1 et S2 sont fixés sur le dessus de l'horloge plutôt que sur le PCB, d'où la présence de K1. Bien que l'ESP32 en fournisse, des résistances de polarisation R6 & R7 de 10 kΩ sont prévues. En effet, ne compter que sur celles de l'ESP32, qui valent plusieurs centaines de kΩ, rendrait un bouton câblé plus sensible au bruit.

Trois LED de débogage (LED1/2/3) donnent l'état des connexions : réseau wifi (bleu), interface du serveur Web (orange) et trafic IR entrant/sortant (rouge).

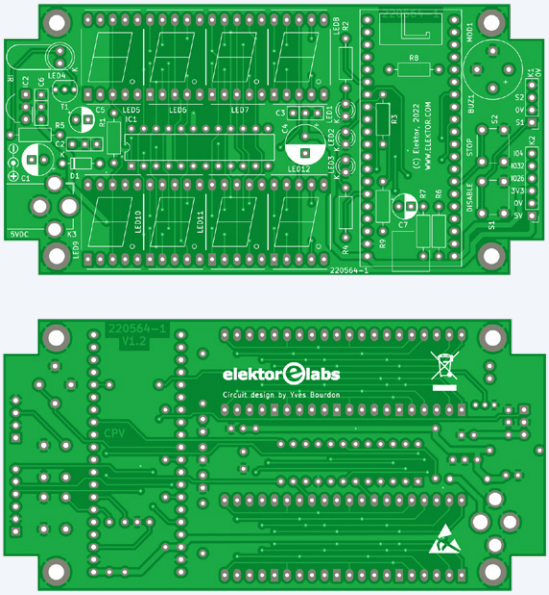
Interface IR

C5 et C6 découplent le récepteur infrarouge IC2 pour nettoyer le signal IR. La LED infrarouge TX (LED4) est pilotée par le transistor T1 et R5 limite à environ 150 mA le courant qui la traverse. La sortie IR est assez puissante pour contrôler tout équipement proche aussi bien que la télécommande d'origine de celui-ci.

Alimentation électrique

L'alimentation externe doit fournir au moins 500 mA sous 5 V_{DC}. La diode D1 protège le circuit contre une inversion de polarité.

LISTE DES COMPOSANTS



Résistances

R1 = 47 kΩ
R2 = 820 Ω
R3, R4, R8 = 330 Ω
R5 = 22 Ω
R6, R7, R9 = 10 kΩ

Condensateurs

C1 = 100 µF, 16 V
C2, C3, C6 = 100 nF
C4 = 220 µF, 16 V
C5 = 47 µF, 16 V
C7 = 1 µF, 16 V

Semi-conducteurs

D1 = 1N5817
IC1 = MAX7219
IC2 = TSOP34338
LED1 = bleu, 3 mm
LED2 = orange, 3 mm
LED3 = rouge, 3 mm
LED4 = TSAL6400, IR LED, 5 mm
LED5-12 = Chiffre à 7 segments 0.56", CC
(par exemple SC56-11EWA)
T1 = BC337

Divers

BUZ1 = Buzzer piézoélectrique sans oscillateur, 12 mm, 3,3 V
K1 = Tête de broche 1x4
K2 = Tête de broche 1x6
K3 = Jack baril
MOD1 = ESP32-PICO-KIT
S1, S2 = Interrupteur tactile 6x6 mm
PCB 220465-1
Boîtier Hammond 1591CTDR (rouge translucide)
Module d'horloge temps réel (RTC) DS3231

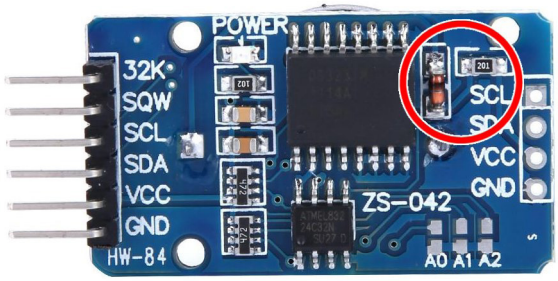


Figure 6. Si le module RTC DS3231 (option) est utilisé avec une pile bouton CR2032 non rechargeable, retirez la résistance ou la diode (ou les deux) dans le cercle rouge pour éviter d'endommager la pile.

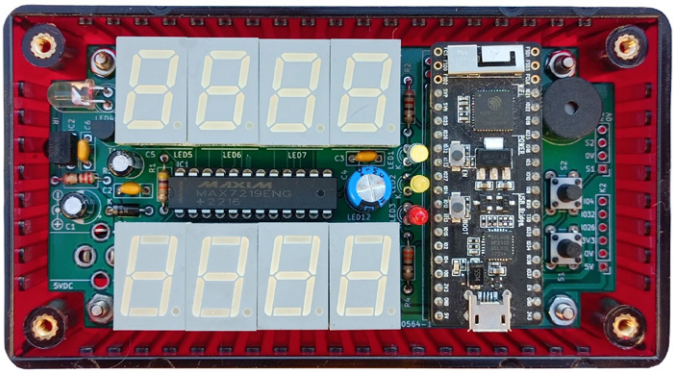


Figure 7. La carte assemblée est à l'aise dans le boîtier. Notez qu'elle peut aussi se monter sur le couvercle pour masquer les vis du boîtier. Le connecteur d'alimentation peut être monté des deux côtés de la carte.

Deux condensateurs électrolytiques (C1 & C4) amortissent les pics de courant. En parallèle des condensateurs de 100 nF (C2 & C3) atténuent le bruit HF.

Extensions

Le connecteur K2 est prévu pour une extension, par ex. à l'aide d'un module d'horloge en temps réel (v. ci-après). Trois ports d'E/S restent libres pour cela : IO4/26/32.

Option horloge en temps réel (RTC)

On peut connecter un module RTC *DS3231* au bus d'extension K2. Il prend le relais si le serveur NTP est inaccessible pour une raison quelconque (par ex. wifi en panne, problème chez le fournisseur d'accès à Internet). Le module RTC est détecté et configuré automatiquement par le logiciel et remis à l'heure une fois par jour (à 12h00). Le DS3231 est très performant et compense la dérive du quartz grâce à un capteur de température intégré.

Le module RTC DS3231 communique en I²C et n'est relié à K2 que par quatre fils : GND, 3,3 V, SDA et SCL. Si un module RTC est détecté, le port IO26 devient le signal SDA et le port IO32 prend en charge le signal SCL du bus I²C.

Danger !

Attention, le RTC DS3231 comporte un risque. En effet, ce module intègre un circuit de charge de la batterie, mais souvent, il n'est livré qu'avec une batterie CR2032 non rechargeable installée qui risque alors d'exploser. Il existe deux solutions à ce problème. La plus simple (et la plus chère) est de remplacer la pile par une pile rechargeable LIR2032. La 2e solution consiste à retirer la diode ou la résistance de 200 Ω (ou les deux) près de la broche SCL du connecteur à 4 broches (**figure 6**).

Détails mécaniques

Le 1591CTDR de Hammond fait un bon boîtier pour Cloc. Il mesure 120 x 63 x 38 mm. Il est rouge translucide (IR) et parfait pour les afficheurs 7 segments de l'heure du jour et de l'heure d'alarme (rouge et/ou orange). La transmission et la réception des signaux IR bénéficient aussi d'un tel filtre optique. Notez que grâce à ce filtrage, la lumière pauvre en contenu rouge (par ex. la LED bleue) est à peine visible.

À ce propos, l'ESP32-PICO-KIT possède une LED d'alimentation rouge très brillante que vous voudrez probablement cacher ou supprimer.

Le circuit imprimé (PCB) a été conçu en tenant compte de cette enceinte. Le projet KiCad6 peut être téléchargé depuis [1]. Le boîtier loge facilement la carte montée sur quatre goujons de 16 mm (**figure 7**). Pour fixer la carte, il suffit de percer le fond de quatre trous de 3,2 mm ; il faut aussi un trou de 8 mm sur le côté pour le connecteur d'alimentation, et deux trous de 12 mm sur le dessus pour les deux boutons poussoirs.

Le module RTC en option peut être connecté à K2 sous le PCB principal (côté composant vers le haut) en utilisant l'embase vide à 4 broches. On peut préférer retirer l'embase coudée à 6 broches pour éviter les courts-circuits. On peut aussi le coller sur le PCB principal avec du ruban adhésif double face (ce que je l'ai fait).

L'environnement de développement logiciel

Pour développer le logiciel de Cloc 2.0, j'ai utilisé l'EDI Arduino après ajout du paquet des cartes ESP32. Pour éviter les erreurs de compilation ou de gestion mémoire, il faut sélectionner la carte ESP32 PICO-D4 à une fréquence CPU de 240 MHz. Pour le *Partition Scheme* choisir *Default* (c.-à-d. 4 Mo avec SPIFFS). En sélectionnant le bon port série, compilation et téléchargement du programme se déroulent sans problème.

Avant compilation, il faut installer les bibliothèques externes suivantes dans l'IDE. J'ai indiqué dans le code source où les obtenir. La plupart peuvent aussi s'obtenir par le gestionnaire de bibliothèques de l'IDE.

- AsyncElegantOTA
- AsyncTCP
- ESPAsyncWebServer
- IRremote (v. ligne 126 de `Alarm_Clock.ino`).
- LedControl
- RTClib

Important : Il faut modifier `LedControl.h` en commentant la ligne 30 (`#include <avr/pgmspace.h>`). Voir aussi le commentaire dans le code source.

Le téléchargement [1] contient un dossier nommé *libraries* qui comprend toutes les bibliothèques utilisées, dont le `LedControl.h` pré-modifié. Copiez ces bibliothèques dans le sous-dossier *libraries* du dossier *Arduino sketchbook* (chemin indiqué dans la fenêtre *Preferences* de l'IDE).

Pour savoir comment ajouter l'option *ESP32 Sketch Data Upload* à l'IDE, voir [2]. C'est nécessaire pour charger l'interface web de l'horloge en mémoire de données du module ESP32. N'oubliez pas de charger ces données, sinon le serveur web restera muet.

Sans-fil (OTA)

Pour mettre le micrologiciel du réveil à jour sans fil via un navigateur, ciblez-y la page `IP_of_cloc/update` où `IP_of_cloc` représente l'adresse IP du réveil. Choisir alors un exécutable situé sur votre ordinateur et le télécharger dans le réveil. Une explication détaillée de l'interface OTA est disponible [3].

Au cœur du programme

Le programme est assez bien documenté (souvent en français), et après avoir lu ce qui suit, vous devriez être à l'aise. Surveillez le port série de l'ESP32 car beaucoup d'informations de débogage y sont envoyées.

Après l'inclusion des bibliothèques, définissez les valeurs par défaut utilisées soit au 1er démarrage, en maintenant l'appui sur le bouton (S1), soit depuis la page Settings en cliquant sur Defaults. L'ordre des constantes suit l'ordre des paramètres de la page Paramètres (lignes 40 à 65 d'*Alarm_Clock.ino*). Vérifiez en particulier l'adresse IP fixe et la passerelle, ainsi que les identifiants wifi. Pour la page Alarme (lignes 67 à 84), l'organisation est la même.

À la mise sous tension, on configure d'abord les fonctions. Il est facile de suivre l'ordre d'initialisation du matériel Cloc. La fonction `initRTC` vérifie si un module RTC DS3231 est connecté. Si le bouton S1 est enfoncé à ce moment-là, on charge les valeurs par défaut.

Au premier démarrage de l'horloge ou si Stop (S2) est pressé pendant la mise sous tension, on passe en mode Point d'accès (AP). Dans ce cas, comme l'horloge l'indique, un navigateur peut se connecter à son interface web : 192.168.4.1.

Après récupération de l'adresse IP (DHCP ou fixe) et s'être connecté au réseau wifi, on cherche l'heure NTP. Si elle est trouvée et qu'une RTC est présente, ils sont synchronisés. L'adresse IP de l'horloge s'affiche brièvement avant de passer à l'affichage de l'heure. L'écran commence à clignoter « HH:HH » pour l'heure et « AA:AA » pour l'alarme, si le serveur NTP n'a pas pu être trouvé et il faut redémarrer l'appareil (si aucun RTC n'est présent, sinon il affiche l'heure RTC).

L'initialisation s'achève par le démarrage du serveur web et du service `AsyncOTA` qui permet de faire une mise à jour à distance du µlogiciel et des SPIFFS.

La fonction `loop` exécute le reste du programme en une boucle sans fin. Si la RTC est présente (`flagRTC=1`), elle lit l'heure RTC. Sans RTC, le programme essaie l'heure NTP (`NTPflag=1`). Une fois l'horloge connectée, le programme lit l'heure RTC, tandis que `strTime` contient l'heure NTP valide en format « hh:mm:ss », sinon elle contient « 0 » .

Ensuite, l'état des boutons est lu (`checkButtons`) et on vérifie si l'horloge doit émettre un bip et si un code IR a été reçu. Le serveur étant asynchrone, il faut utiliser des drapeaux pour signaler au programme principal les changements de paramètres et les déclenchements d'alarme.

À chaque seconde l'écran est rafraîchi (`dispTime`) et les alarmes sont lues (`checkAlarm`). Le programme vérifie aussi s'il est en période de haute luminosité ou non (`checkBrightness`) et s'il doit synchroniser le RTC avec l'heure NTP (`checkSyncRTC`). Cela se produit une fois par jour à 12 h (`syncRTC=12* 60`). La fonction `checkTimer` vérifie si une alarme doit être activée. Pour cela, l'heure courante et l'heure de l'alarme sont converties en minutes (`computeMinutes`) avant comparaison (`testRange`).

Comme dit ci-avant, le serveur est asynchrone. Chaque fois que la page Paramètres ou Alarme est rafraîchie, les données à afficher (`ParametresProcessor` et `TimerProcessor`) sont renvoyées. Les balises `%value%` de la page HTML doivent donc être remplacées par les valeurs réelles.

Spécifications

- Basé sur un processeur ESP32.
- Deux rangées de 4 afficheurs à 7 segments : l'une indique l'heure qu'il est, l'autre l'heure d'alarme.
- Réglage automatique de l'heure par connexion à un serveur de temps en ligne.
- Heure d'alarme propre à chaque jour de la semaine.
- Sortie d'alarme : buzzer et code infrarouge, par ex. pour une radio, chaîne hi-fi ou TV.
- Deux boutons-poussoirs d'interaction avec l'appareil.
- Serveur web intégré pour la configuration à distance via Wi-Fi.
- Tous paramètres enregistrés en EEPROM
- Le mode Over-the-Air (OTA) permet la mise à jour sans fil du micrologiciel.
- Module optionnel d'horloge temps réel (RTC) basé sur DS3231 secouru par batterie.

Pour la page principale, c'est un peu différent car il faut afficher presque en temps réel l'heure du jour, la date et les heures de début et de fin de l'alarme. Une fonction écrite en JavaScript interroge environ chaque seconde la boucle principale pour récupérer les valeurs à afficher (`onDataUpdate`). À chaque fois, la LED2 clignote brièvement pour indiquer l'accès au serveur (`flashRqstBeacon`). Pour vérifier que la page affiche bien les valeurs en temps réel, appuyez sur S1 (*Alarm Disable*). L'heure de l'alarme est affichée ou désactivée simultanément sur l'afficheur 7 segments et sur la page HTML.

Bibliothèque *IRremote*

La bibliothèque `IRremote` fonctionne parfaitement, mais nécessite quelques règles d'intégration. Ne déplacez pas le code d'initialisation, sauf si vous savez ce que vous faites.

Pour le réveil en musique au lieu d'un bip, j'utilise une radio Bose. J'ai donc limité la réception et le décodage aux télécommandes Bose (voir ligne 125 de `Alarm_Clock.ino`). Vous pouvez supprimer cette ligne pour décoder d'autres télécommandes. Je recommande de limiter la réception infrarouge à la marque de votre radio ou de votre chaîne hi-fi. Pour les autres options possibles, voir [4].

De même, la transmission IR est limitée au format Bose. Il est donc fort probable que vous deviez adapter le code à la marque de votre équipement Hi-Fi. Remplacez la ligne `IrSender.sendBoseWave` des fonctions startMedia et `stopMedia` de `Alarm_Clock.ino` par une fonction correspondant à votre marque (voir [4]).

Pour découvrir les codes IR à utiliser avec votre système (audio), reliez le port USB du module ESP32 à un ordinateur utilisant un moniteur série, par exemple le Serial Monitor intégré à l'IDE Arduino. Dirigez votre télécommande vers le récepteur IR de l'horloge et pressez une touche ; le code correspondant en hexadécimal s'affiche sur l'ordinateur. Saisissez les codes en hexadécimal dans les champs *Media ON* et *Media OFF* du panneau Paramètres.

Le clignotement bref de la LED3 signale l'activité IR. Cette LED clignote donc également lorsqu'une alarme est active.

Faites-le vous-même !

Depuis plusieurs mois déjà, j'utilise cette Cloc version 2.0 et je n'ai jamais manqué un réveil à cause d'un dysfonctionnement. La possibilité de définir un heure d'alarme pour chaque jour de la semaine est très pratique.

Je recommande vivement d'ajouter l'option RTC car si le serveur NTP ne répond pas (par ex. si votre modem s'est déconnecté pour une raison quelconque), le réveil perdra l'heure au moment suivant de resynchronisation de l'heure NTP. Il faudra alors arrêter et redémarrer le réveil une fois l'heure NTP à nouveau disponible. Pour une fiabilité accrue, j'ai réalisé un petit serveur NTP qui reçoit l'heure d'un GPS [5]. Ainsi, même en l'absence de connexion Internet, mon réveil se règle automatiquement.

J'espère que vous serez nombreux à réaliser ce réveil. N'hésitez pas à me contacter sur mon courriel privé (yb.electronique@orange.fr) si vous rencontrez des difficultés ou si vous apportez des améliorations à mon réveil préféré !

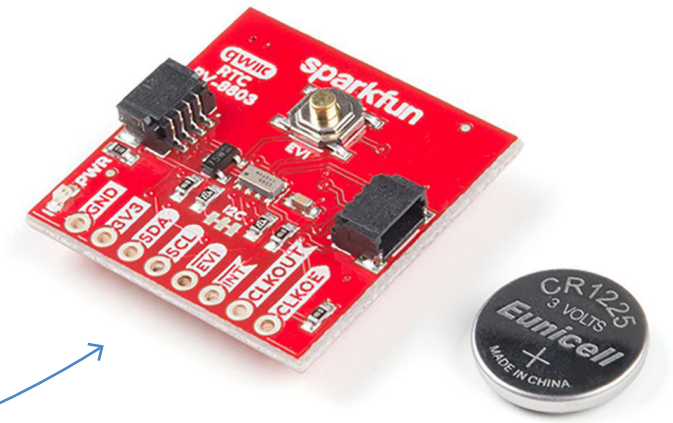
220564-04 — *VF : Yves Georges*

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (yb.electronique@orange.fr) ou contactez Elektor (redaction@elektor.fr).

Produits

- **ESP32-PICO-Kit V4 (SKU 18423)**
  www.elektor.fr/18423
- **Elektor Arduino Electronics Bundle (SKU 19440)**
  www.elektor.fr/19440
- **Module RTC de SparkFun – RV-8803 (Qwiic) (SKU 19646)**
  www.elektor.fr/19646



LIENS

[1] Téléchargements pour Cloc 2.0 (Elektor Labs) : https://www.elektormagazine.fr/labs/cloc-le-reveil-20
[2] Téléchargement de données ESP32 SPIFFS : https://randomnerdtutorials.com/install-esp32-filesystem-uploader-arduino-ide/
[3] Mises à jour OTA (Over-the-Air) de l'ESP32 : https://randomnerdtutorials.com/esp32-ota-over-the-air-arduino/
[4] Bibliothèque IRremote : https://github.com/Arduino-IRremote/Arduino-IRremote
[5] Serveur NTP (Elektor Labs) : https://www.elektormagazine.fr/labs/ntp-server