thermostat connecté à ESP32

Conservez votre vin à la bonne température !

Yves Bourdon (France)

Pour un bon vieillissement à long terme, le vin doit être conservé dans un endroit sombre et frais. Même si les spécialistes ne sont pas d'accord sur la meilleure température pour cela, ils recommandent de maintenir une valeur de température aussi constante que possible. Le thermostat présenté ici tente non seulement de le faire, mais il envoie également des alertes par courrier électronique lorsque la température devient trop élevée.

L'idée de ce projet m'est venue alors que je construisais l'excellent thermostat Wi-Fi d'Elektor [1], conçu autour de l'ESP8266 d'Espressif. J'ai essayé de porter le logiciel de ce projet sur un ESP32, mais j'ai abandonné à cause du travail que cela impliquait. J'ai donc décidé de repartir de zéro en m'inspirant de ce qui avait déjà été fait (interface web, horloge NTP, régulation de température avec hystérésis programmable, etc.) et d'améliorer l'interface web en corrigeant ses défauts (en particulier, le fait que l'interface utilisateur ne soit pas rafraîchie en temps réel). J'ai également ajouté un petit écran OLED très pratique, qui permet d'utiliser ce thermostat même si la connexion Wi-Fi est interrompue (mode dégradé).

Ce projet est facilement adaptable à d'autres applications. J'ai par ex. utilisé le même matériel avec un détecteur d'humidité DHT11 (ou BME280) pour commander le système de ventilation mécanique de ma salle de bains. L'entrée opto-isolée détecte si la lumière est allumée (spots de 12 V) afin de pouvoir éteindre temporairement l'extracteur d'air bruyant lorsqu'on occupe la salle de bains.

Le circuit

La **figure 1** donne le schéma du thermostat à ESP32. Pour optimiser le coût, la fiabilité et la taille, j'ai décidé d'utiliser le module WiFi Kit 32 de Heltec Automation (**fig. 2**) au cœur du thermostat. On trouve facilement sur l'internet ce module basé sur l'ESP32, doté d'un bel écran OLED bleu de 2,4 cm et 128 × 64 pixels ainsi que d'une LED blanche programmable par l'utilisateur. Il se branche sur K5 et K6.

Notez qu'il est possible de remplacer le module WiFi Kit 32 par un module ESP32

Spécifications

- Sonde de température DS18B20, précision de ±0,5 °C, affichage à 0,1 °C près.
- Relais avec contacts amortis permettant de commuter des charges CA jusqu'à 2.200 W.
- Alimentation 100-240 VCA.
- Bus d'extension I2C compatible avec modules Grove et Qwiic.
- Le circuit imprimé est compatible avec le WiFi Kit 32 de Heltec (avec écran OLED intégré) ou le DevKitC ESP32 avec écran OLED externe.
- Buzzer pour alarme sonore.
- Entrée pour interrupteur externe.
- Entrée isolée optiquement.
- Trois seuils programmables : minimum, maximum et alarme.
- L'alarme peut envoyer un courriel.
- Maintien de l'heure basé sur NTP.
- Minuterie hebdomadaire.
- Peut commander un refroidisseur ou un réchauffeur.

DevKitC (connecteurs K7 et K8) avec un écran séparé OLED I2C de 2,4 cm et 128 × 64 pixels, tous deux assez courants (**fig. 3**). L'écran OLED existe en deux versions de brochage (avec VCC et GND inversés) et donc deux connecteurs sont prévus pour lui (K9 et K10). Le logiciel est identique pour les deux modules.

Alimentation électrique

Le bloc d'alimentation est un petit convertisseur CA/CC de 3 W. Il accepte une tension d'entrée comprise entre 100 et 240 VCC pour une sortie en 5 V jusqu'à 600 mA. La LED3 indique si l'appareil est sous tension. Un fusible protège l'appareil et le secteur, tandis que la varistance VAR1 assure une protection contre les surtensions (très appréciée par les normes de conformité). Le raccordement au secteur s'effectue via le bornier K1.

Le module ESP32 est alimenté à partir du 5 V, car il dispose de son propre régulateur de 3,3 V. Le module peut également être alimenté directement via le bus micro-USB.

Relais

La charge du thermostat est commutée par le relais RE1, qui, à son tour, est commandé par un petit transistor NPN (T1) connecté au port IO18. La LED2 indique l'état du relais. La diode D1 protège l'électronique des courants de commutation induits par le relais, tandis que le réseau de filtrage C4/R9 ou C4/R10 (ne mettre qu'une résistance) protège les contacts du relais en amortissant toute étincelle de commutation en cas de forte charge.

Le relais utilisé peut piloter des charges de 250 VCC jusqu'à 10 A. Les contacts normalement ouverts (NO) et normalement fermés (NC) sont tous deux disponibles sur K2. Notez que K1 et K2 peuvent être combinés en un seul bornier à cinq voies.

Buzzer

Un buzzer avec oscillateur intégré (Buz1), piloté par T2 via le port IO19, est disponible pour un retour auditif. Comme il dispose d'un oscillateur intégré, il suffit de l'alimenter pour qu'il émette un bip. On peut aussi utiliser un buzzer à courant alternatif, mais l'ESP32 devra alors produire lui-même le son.

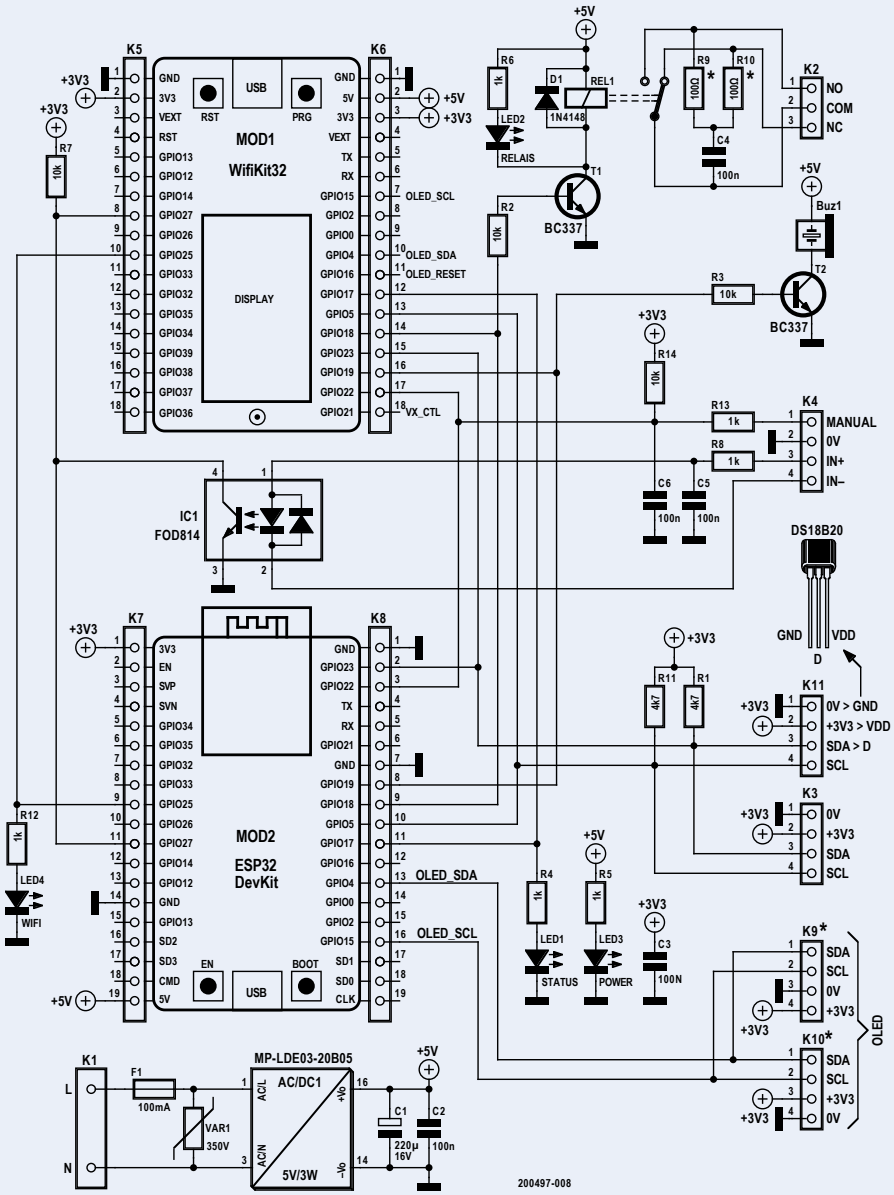


Figure 1. On trouve de nombreux connecteurs sur le schéma du thermostat à ESP32, car il y a plusieurs options de configuration.

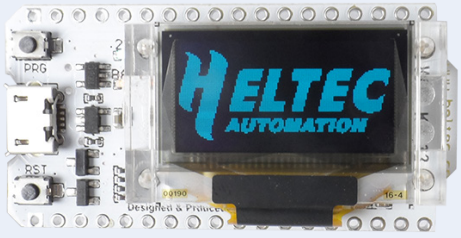


Figure 2. Le WiFi Kit 32 de Heltec Automation.

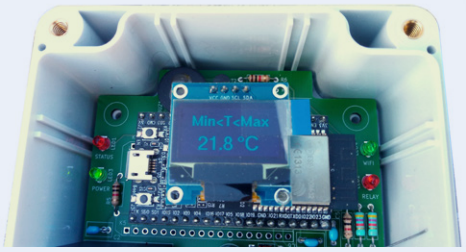


Figure 3. Le thermostat construit par le labo d'Elektor utilise un module DevKitC ESP32 avec un module d'affichage OLED séparé.

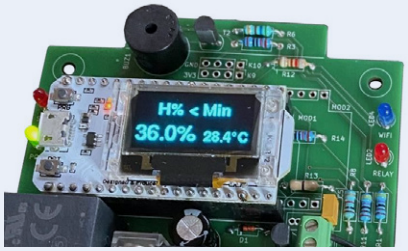


Figure 4. Le thermostat reprogrammé en contrôleur d'humidité, ici équipé d'un module Heltec WiFi Kit 32 avec écran OLED intégré.

Figure 5. La page principale affiche la température mesurée ainsi que les seuils de commutation et d'alarme.

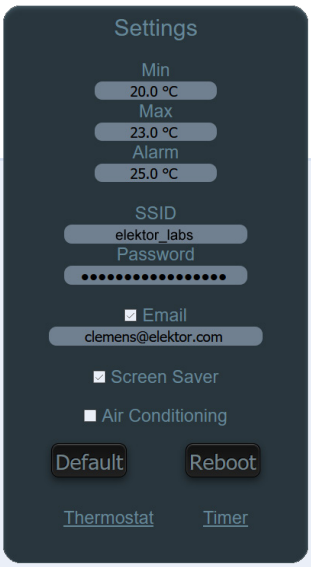


Figure 6. Les températures minimale et maximale ainsi que le seuil d'alarme sont définis ici. C'est également sur cette page que vous définissez le SSID et le mot de passe du réseau Wi-Fi que le thermostat doit utiliser.

LED d'état

La LED1, connectée au port IO17, est un indicateur d'état. Un clignotement rapide signale une erreur due à des données EEPROM mauvaises ou invalides, ou encore à des problèmes de communication du capteur DS18B20. Ces erreurs sont bloquantes, et le programme s'arrête jusqu'à ce que le problème soit résolu. Des clignotements courts en mode de fonctionnement normal indiquent que l'on accède au serveur web intégré.

Contrôles externes

Le bornier K4 permet la connexion d'un interrupteur externe (*manuel*) entre le port IO22 et GND. Avec le logiciel par défaut, cet interrupteur sera prioritaire sur le thermostat et activera le relais. Un réseau RC de filtrage/antirebonds (1,5 kHz avec les valeurs de composants données) peut être mis sur cette entrée. Lorsqu'on utilise un interrupteur bistable, on peut omettre C6 et remplacer R13 par un bout de fil.

Une entrée numérique *IN* isolée optiquement (optocoupleur IC1) connectée au port IO2 est disponible pour une extension future et n'est pas exploitée par le logiciel actuel. Elle me sert pour détecter l'allumage de l'éclairage de la salle de bains mentionné précédemment (**fig. 4**).

Connexion du capteur et port I²C

Le capteur de température, un DS18B20 de type One-Wire, est destiné à être connecté au bornier K3. Il faut relier sa broche 1 à GND, sa broche DQ (broche 2) à la broche 3 de K3 et enfin sa broche VDD au 3,3 V. Cela signifie que ses pattes 2 et 3 se croisent. Néanmoins, pour des applications comme un thermostat, il ne faut pas mettre le capteur de température à l'intérieur du boîtier, car l'alimentation et le module ESP32 y chauffent l'air, ce qui fausse la régulation de température.

La résistance R1 est nécessaire pour le bon fonctionnement du DS18B20. Si on remplace ce capteur de température par quelque chose comme un capteur d'humidité (tel qu'un BME280, SHT11 ou DHT22), on peut ne pas avoir besoin de R1.

En réalité, K3 est câblé de manière à être compatible avec les standards de facto des fabricants de composants à bus I2C comme Grove (Seeed Studio) et Qwiic (SparkFun). De fait, K11 est un connecteur Grove (au pas de 2 mm). De nombreuses cartes d'extension compatibles avec ce format de connecteur sont disponibles, ce qui fait du circuit du thermostat une excellente base pour d'autres applications.

Circuit imprimé

Elektor a mis au point un circuit imprimé qui s'insère dans un boîtier bon marché doté d'un couvercle transparent. Les pistes du relais qui se connectent à la charge peuvent et doivent être renforcées avec de la soudure.

Logiciels et programmation

Le programme du thermostat est écrit en C/C++ dans l'EDI Arduino (avec la version 1.8.12). Il est assez gros – plus de 1.500 lignes de code – mais il est abondamment commenté (bien que dans un mélange d'anglais et de français). Il est assez facile de l'adapter à vos besoins et je n'ai utilisé que des bibliothèques *open source* gratuites et éprouvées.

Avant de pouvoir utiliser l'ESP32 avec l'EDI Arduino, vous devez d'abord installer le paquetage de cartes correspondant. De nombreuses pages web ont été écrites pour expliquer cette procédure, référez-vous à [2] pour plus de détails (avec vidéo). Le paquetage de cartes officiel ESP32 inclut la prise en charge du Heltec WiFi Kit 32 et du DevKitC (« ESP32 Dev Module »). Quel que soit le module choisi, vous pouvez laisser tous les autres paramètres (vitesse de l'UC, taille de la mémoire flash, etc.) à leurs valeurs par défaut. Veillez toutefois à sélectionner le bon port COM.

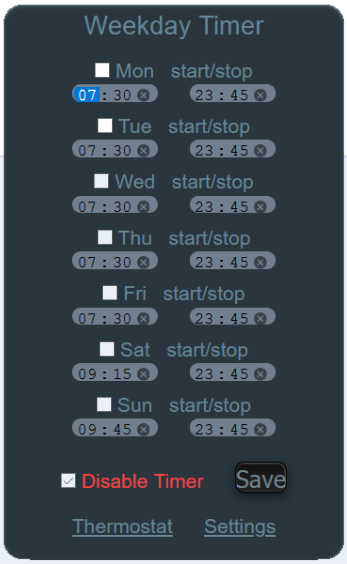


Figure 7. La page « Timer » vous permet de définir pour chaque jour de la semaine une période pendant laquelle le thermostat doit réguler la température.

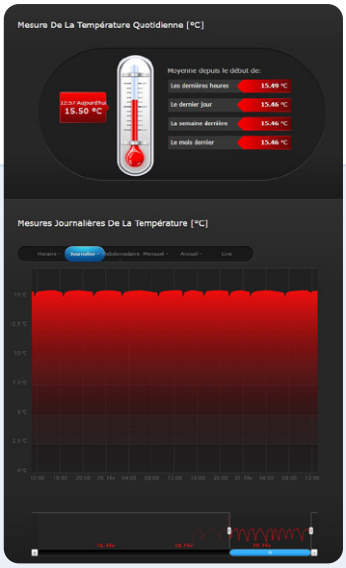


Figure 8. La régulation est plutôt bonne comme le montre cette capture d'écran de mon système de domotique. Il faut savoir qu'une variation de 1 °C dans le sous-sol correspond en fait à une variation de seulement 0,1 °C de la température du vin dans la bouteille.

PRODUITS

- **ESP32 DevKitC**
  www.elektor.fr/esp32-devkitc-32d
- **Écran OLED I²C, 128×64 pixels, de 2,3 cm**
  www.elektor.fr/blue-0-96-oled-display-i2c-4-pin
- **Combiné Elektor : kit ESP32 & livre en anglais « The Complete ESP32 Projects Guide »**
  www.elektor.fr/19033

Outre l'installation du paquetage de cartes ESP32, il est également nécessaire d'installer l'outil SPIFFS *ESP32 Sketch Data Upload*. Là encore, reportez-vous à [2] pour savoir comment procéder.

Téléchargez le croquis du thermostat depuis [3] et décompressez-le dans le carnet de croquis de l'EDI Arduino. Le fichier *Thermostat.ino*, les fichiers qui l'accompagnent et le dossier *data* doivent se trouver dans un dossier nommé *Thermostat*. Le sous-dossier *data* contient les fichiers HTML et CSS ainsi que les programmes JavaScript nécessaires au serveur Web. Le fichier *D7MR.woff2* contient la police à 7 segments que j'ai utilisée. Un grand « Merci ! » à son auteur.

Pour votre confort, j'ai inclus dans le téléchargement toutes les bibliothèques nécessaires au projet [3]. Elles doivent être installées correctement, c'est-à-dire en les copiant dans le sous-dossier *libraries* du carnet de croquis de votre EDI. L'emplacement de ce dossier sur votre ordinateur dépend de la façon dont vous avez configuré l'EDI Arduino. Reportez-vous à [4] pour plus de détails sur l'EDI Arduino.

Une fois que tout le logiciel est téléchargé et installé, vous devez régler certains paramètres par défaut du programme pour les adapter à votre environnement domestique ou votre application. Ils sont situés en haut du programme. Comme le thermostat utilise une adresse IP fixe, vous devez en définir une avec la passerelle et d'autres éléments typiques du réseau.

Configuration du courriel

Comme le thermostat peut envoyer des courriels, il est nécessaire d'entrer les caractéristiques du serveur de messagerie. Je vous recommande de créer un compte Gmail pour cela. Dans ce cas, le serveur de messagerie est *smtp.gmail.com* et le port à utiliser est 465. Pour des raisons de sécurité, vous devez configurer le compte Gmail. Si votre compte utilise une authentification à un facteur (c'est-à-dire qu'il ne nécessite qu'un mot de passe), configurez le compte en suivant le lien [5]. En cas d'authentification à deux facteurs (c.-à-d. un mot de passe et un code par SMS), référez-vous à [6].

Il est maintenant temps de compiler et de télécharger le croquis comme tout autre croquis Arduino. Vous ne devriez pas rencontrer de problèmes sérieux, voire aucun. Sinon, vérifiez votre installation.

Après avoir programmé le croquis, téléchargez les pages web et les autres fichiers de données avec la commande *ESP32 Sketch Data Upload* disponible dans le menu *Outils* de l'EDI. Cette commande permet de transférer tout ce qui se trouve dans le dossier *data* vers la mémoire flash (SPIFFS) de l'ESP32.

Premier démarrage

Lorsque vous démarrez l'appareil pour la première fois, connectez-vous à son propre point d'accès Wi-Fi (AP) avec le SSID « Thermostat ESP32 » ; il n'y a pas de mot de passe. Ouvrez un navigateur et allez à l'adresse IP 192.168.4.1. L'interface web devrait maintenant apparaître. Cliquez sur le lien *Settings*, puis sur le bouton *Default*. Cela permettra d'initialiser tous les paramètres – y compris les réglages de la minuterie et de la température – avec leurs valeurs par défaut et d'activer le SSID et le mot de passe afin que le thermostat puisse se connecter à votre réseau Wi-Fi. Redémarrez l'appareil pour que les paramètres soient actifs. Le bouton *Reboot* permet de réinitialiser le thermostat.

Notez que lorsque le SSID et/ou le mot de passe sont modifiés, le système redémarre automatiquement.

Pour les paramètres de la minuterie et de la température, il n'est pas nécessaire de valider un champ après avoir modifié sa valeur. Il suffit de déplacer la souris hors du champ. De plus, il y a un contrôle de validité sur les seuils de température minimum et maximum. Si la valeur maximale est inférieure à la valeur minimale, les deux sont permutées.

Interface utilisateur chatoyante

Une grande attention a été portée à la conception de l'interface utilisateur. L'objectif était de créer une interface web soignée qui s'affiche correctement sur les navigateurs les plus courants et récents ainsi que sur les ordiphones iPhone et Android. L'interface est très réactive, sans latence notable, et les valeurs sont rafraîchies toutes les secondes. L'interface utilisateur basée sur le web comporte trois pages : *Main* (**fig. 6**), *Settings* (**fig. 6**) et *Timer* (**fig. 7**).

Sur la page principale, des codes de couleur

servent à indiquer l'état des paramètres. Par exemple, l'intensité du signal Wi-Fi (RSSI) est affichée en vert, orange ou rouge selon la qualité du signal. L'heure, obtenue à partir d'un serveur NTP (*Network Time Protocol*) situé quelque part sur l'internet, est affichée en rouge lorsque le minuteur commande le relais ; sinon, elle est en vert. De même, la température est affichée en bleu clair si la température est inférieure à la valeur minimale ; elle est en vert si la température est comprise entre le minimum et le maximum. Lorsque la température est supérieure à la valeur maximale, elle est en rouge. Lorsqu'elle commence à clignoter en orange, le système a rencontré un problème. Une petite LED est symbolisée sur la page principale pour montrer en temps réel l'état du relais (rouge s'il est activé, vert sinon). La feuille de style CSS a demandé beaucoup de travail pour rendre possible cet affichage original.

L'écran OLED affiche également l'état du système. Comme on a constaté que ces écrans ont tendance à se dégrader assez rapidement lorsqu'ils restent allumés en permanence, on a ajouté une option d'économiseur d'écran. En appuyant sur un bouton-poussoir connecté entre GPIO05 (K11, broche 4) et GND (K11, broche 1), l'écran s'allume pendant quelques minutes. Cette fonction peut être activée ou désactivée sur la page des paramètres.

Page de la minuterie

Les horaires hebdomadaires d'activation et de désactivation, ainsi que la case à cocher d'activation correspondante, doivent être enregistrés avec le bouton *Store*. En cliquant sur ce bouton, la page entière sera sauvegardée. La case à cocher *Timer Disabled* ne nécessite pas de validation avec le bouton *Store*. Lorsque la minuterie est désactivée, le thermostat régule la température en continu.

Mode débogage

En surveillant le port série via l'interface USB intégrée (115.200 bauds), on peut observer le processus d'initialisation du thermostat après sa mise sous tension. Très utile pour identifier des choses telles que des problèmes de connexion Wi-Fi.

Un système fiable

Le thermostat décrit dans cet article régule la température de mon sous-sol depuis plus d'un an maintenant et je n'ai constaté aucun problème majeur. Le système est fiable et la régulation est très bonne (**fig. 8**). J'ai pu tester l'interface web sur la majorité des navigateurs internet (Chrome, Edge, Safari, etc.) ainsi que sur iPhone et certains téléphones Android et je n'ai observé aucun problème de compatibilité.

Je tiens à remercier Stéphane Calderoni (docteur en informatique) qui vit à La Réunion (une île d'Afrique de l'Est) qui m'a beaucoup aidé dans le développement de l'interface HTML et du JavaScript qui va avec. J'ai profité de la période d'enfermement de mars 2020 en France pour améliorer mes connaissances sur cette partie que je ne maîtrisais pas du tout.

(200497-04)

Figure 9. Le capteur DS18B20 a été monté dans une petite boîte en plastique séparée, fixée à un endroit stratégique de la cave. Un deuxième capteur, identique, transmet la température à mon système de domotique via un module spécifique. Cela ajoute une redondance de sécurité au cas où le thermostat tomberait en panne. Si cela se produit, mon système de domotique me prévient.

Contributeurs

Idée, conception et texte : **Yves Bourdon**
Conception des circuits imprimés et rédaction : **Clemens Valens**
Mise en page : **Harmen Heida**
Traduction : **Denis Lafourcade**

Des questions, des commentaires ?

Envoyez un courriel au rédacteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

LISTE DES COMPOSANTS

Résistances (5%, 0,25 W)

R9, R10 = 100 Ω, 1 W (voir texte)
R4, R5, R6, R8, R12, R13 = 1 kΩ
R1, R11 = 4,7 kΩ
R2, R3, R7, R14 = 10 kΩ

Condensateurs

C2, C3, C5, C6 = 100 nF, pas de 2,5 ou 5 mm
C4 = 100 nF, X2, pas de 15 mm
C1 = 220 µF,16 V, pas de 3,5 mm

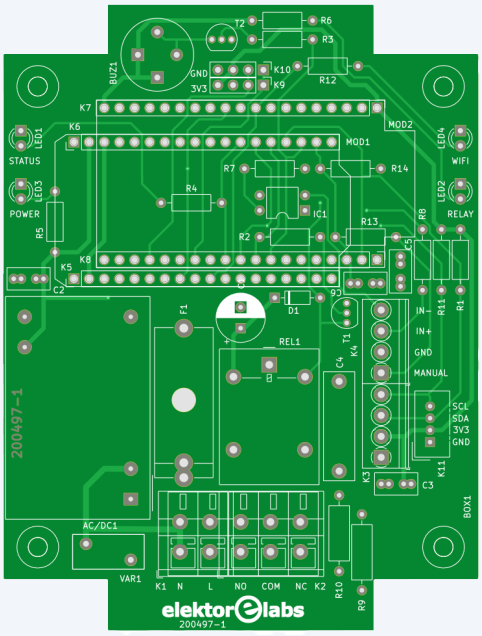
Semi-conducteurs

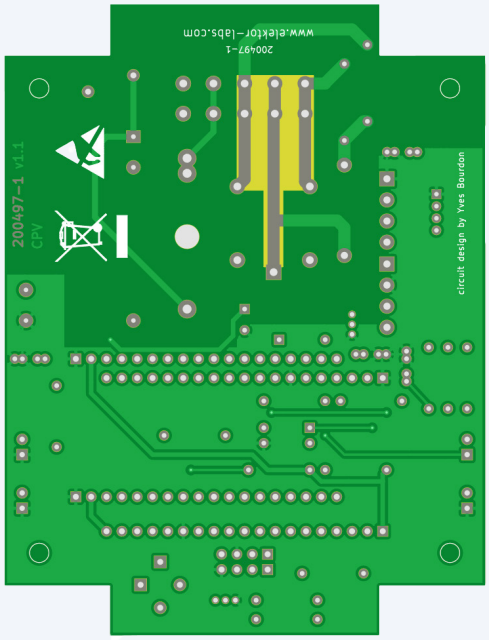
D1 = 1N4148
LED1, LED2 = LED, rouge, 3 mm
LED3, LED4 = LED, verte, 3 mm
IC1 = FOD814, DIP4
T1, T2 = BC337

Divers

BUZ1 = buzzer avec oscillateur, pas de 6,5 ou 7,6 mm
F1 = porte-fusible, 5×20 mm + fusible 250 V, 100 mA, lent
K1 = bornier à 2 voies, pas de 5 mm
K2 = bornier à 3 voies, pas de 5 mm
K3, K4 = bornier à 4 voies, pas de 3,5 mm
K5, K6 = barrette femelle à 18 positions, pas de 2,54 mm (uniquement MOD1)
K7, K8 = barrette femelle à 19 positions, pas de 2,54 mm (uniquement MOD2)
K9, K10 = barrette femelle à 4 positions, pas de 2,54 mm, hauteur de 20 mm (uniquement MOD2)
K11 = barrette mâle à 4 broches, pas de 2 mm (Grove)
REL1 = relais, 10 A, SPDT, 5 VDC
VAR1 = varistance 350 V

AC/DC1 = 5 V, 3 W, convertisseur CA-CC (par ex. Multicomp MP-LDE03-20B05)
BOX1 = boîtier, IP65, couvercle transparent (Multicomp MC001106)
MOD1 = WiFi Kit 32 de Heltec
ou
MOD2 = ESP32 DevKitC + écran OLED, I²C, 128×64 pixels





LIENS

[1] R. Aarts & C. Valens, « thermostat de bureau wifi », Elektor, 01-02/2018 : www.elektormagazine.fr/160269
[2] Vidéo « ESP32 – Getting started » : www.elektormagazine.fr/labs/1874
[3] Thermostat ESP32, Elektor Labs : www.elektormagazine.fr/labs/4216
[4] FAQ Arduino : www.elektormagazine.fr/labs/1876
[5] Authentification à un facteur de Gmail : https://myaccount.google.com/lesssecureapps
[6] Authentification à deux facteurs de Gmail : https://security.google.com/settings/security/apppasswords