cadre photo couleur à encre électronique Wifi

Jeroen Domburg, Espressif

Les écrans à encre électronique occupent une large place sur le marché, mais ils sont principalement en noir et blanc, et les quelques modèles polychromes sont plutôt chers et difficiles à mettre en œuvre dans un projet d'amateur. Dans cet article, nous verrons comment utiliser un écran sept couleurs d'un prix raisonnable pour obtenir une bonne fidélité des couleurs - obtenue grâce à des techniques de tramage - et avec la capacité de se connecter à un réseau Wifi, en utilisant un module ESP32-C3-WROOM-02 d'Espressif.

Notre petite dernière est née il y a déjà quelque temps. Malheureusement, la dispersion de ses (arrière-) grands-parents de par le monde rend les visites en personne assez peu fréquentes. Bien sûr, les appels vidéo sont monnaie courante de nos jours, ce qui nous permet de rester en contact, mais je voulais que tout le monde puisse aussi la voir lorsque nous ne sommes pas connectés. Alors, il m'est venue une idée : construire un cadre photo autour d'un écran couleur et d'une puce Wi-Fi. Comme elle grandit très vite, ce cadre serait quotidiennement rafraîchi avec la photo du jour.

Vous avez sans doute déjà vu un écran à encre électronique : sinon une liseuse, du moins dans un magasin comme le supermarché local, où ils ont remplacé les anciennes étiquettes de prix en papier. Ces écrans sont parfaits pour afficher des images statiques, car ils conservent l'image la plus récente sans consommer d'énergie. La plupart de ces écrans sont en noir et blanc, avec parfois du rouge ou du jaune en option. Ils sont agréables, mais une image en noir et blanc ne rend pas justice aux couleurs vibrantes d'un bébé heureux trottant avec ses jouets colorés.

Sur le marché actuel

Au moment où j'écris ces lignes (début 2023), les liseuses dotées d'un écran couleur commencent enfin à apparaître. Leur qualité semble bonne, avec des couleurs proches de celles d'un journal. Malheureusement, elles sont chères et les écrans eux-mêmes ne semblent pas encore être disponibles en tant que composants, de sorte que les récupérer pour un projet d'amateur est pratiquement impossible.

Cependant, il existe un modèle d'écran couleur à encre électronique utilisable pour les projets de bricolage. Son principal vendeur, Waveshare, est une entreprise chinoise spécialiste du marché des bricoleurs, et le modèle le plus courant est un écran de 5,65 pouces à sept couleurs (640×448) (**figure 1**). Avec seulement sept couleurs et un temps de rafraîchissement d'environ une minute, il semble destiné à devenir le grand frère des étiquettes de prix, pour l'affichage d'informations statiques, avec les couleurs utilisées pour, par exemple, remplir des aplats, mais sûrement pas pour produire des images photoréalistes.

C'est un peu dommage : un écran à encre électronique constitue une excellente imitation d'une photo imprimée, car il ne nécessite pas de rétroéclairage et est entièrement statique. Je ne suis pas le seul à le penser, et un certain nombre de personnes ([1], [2], [3]) ont déjà essayé d'utiliser le tramage pour obtenir un affichage d'images avec une certaine fidélité. Voici ma propre tentative.

Figure 1. L'écran de 5,65 pouces à sept couleurs et 640×448 de Waveshare. (Source : Waveshare)

◀

Exigences en matière de matériel

Commençons par le matériel, avec quelques exigences de conception. Je voulais que le cadre photo se connecte à un serveur par Wifi une fois par jour pour récupérer de nouvelles images et en afficher une. En cas de succès, c'est ce qu'il ferait, sinon il continuerait à en afficher une ancienne. Il se remettrait ensuite en veille pendant 24 heures. Le choix d'une puce n'a pas été difficile, car je travaille pour Espressif, une société qui fabrique d'excellents microcontrôleurs gérant le Wifi et disposant d'un mode sommeil profond qui fonctionne assez bien. Pour ce projet, j'ai choisi un ESP32C3 : sympathique et bon marché, il possède toutes les fonctions dont j'avais besoin. J'avais également quelques exigences concernant l'alimentation électrique. Je voulais que cet appareil ressemble le plus possible à un cadre photo normal (sauf, bien sûr, qu'il change d'image chaque nuit), donc pas question d'une alimentation externe. Elle devait donc être une sorte de batterie interne, et comme je désirais expédier l'appareil à l'étranger, il fallait qu'elle soit amovible, ce qui excluait tout type de cellule Li-ion rechargeable. Comme vous pouvez le voir sur le schéma de la **figure 2**, j'ai décidé d'utiliser deux piles AA ; elles sont disponibles partout et même les grands-parents n'auraient aucun mal à les remplacer le moment venu. Comme deux piles AA délivrent une tension totale qui démarre à environ 3,2 volts et qui diminue au cours de leur durée de vie, j'avais besoin d'un circuit élévateur pour atteindre les 3,3 V nécessaires pour l'écran et l'ESP32C3.

Figure 2. Schéma complet du projet, comprenant le convertisseur DC-DC 3,3 V, le module ESP32-C3-WROOM-02 et l'alimentation de l'écran.

▼

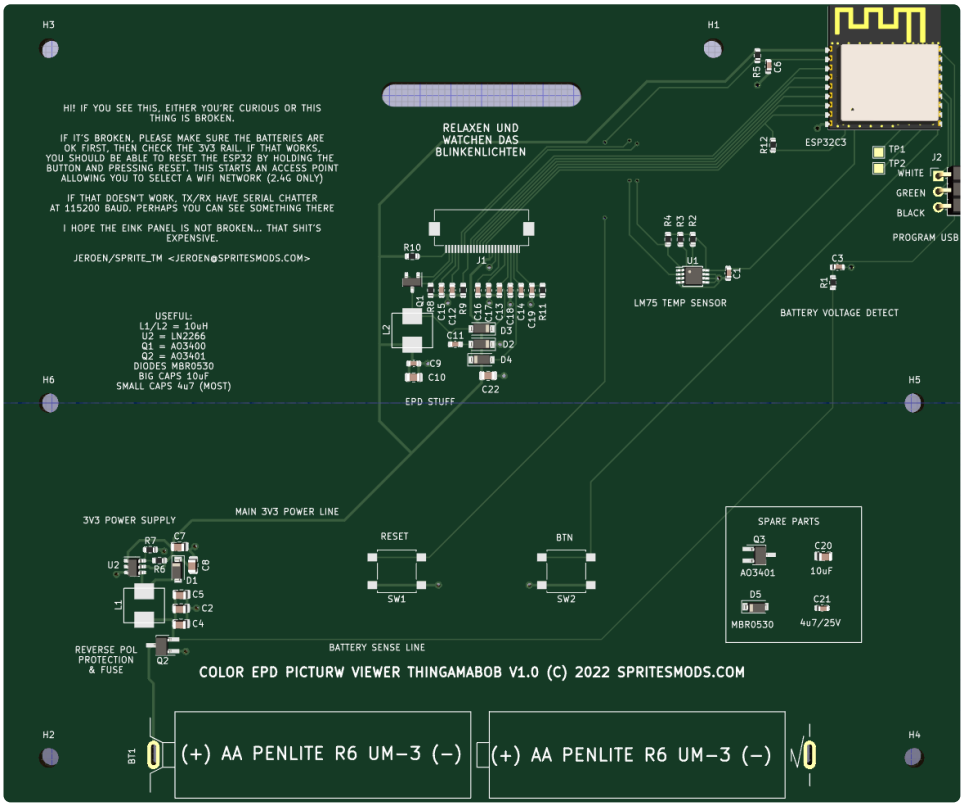

Figure 3. Sérigraphie (côté composant) du circuit imprimé spacieux du projet.

Les courbes de décharge [4] montrent que si je voulais tirer le meilleur parti d'une paire de piles alcalines, le circuit élévateur devait fonctionner avec une tension d'entrée de 2,0 V environ. De plus, comme l'ESP32-C3 doit être correctement alimenté même en veille profonde, le courant de repos doit être suffisamment faible. Avec ces exigences, je me suis mis à la recherche d'un convertisseur élévateur réalisable.

J'ai opté pour le Natlinear LN2266. Cette petite puce est fabriquée par un fabricant chinois, ce qui signifie qu'elle souffre un peu moins de la pénurie actuelle de silicium. Elle fonctionne à partir de 2 V, a un courant de repos de 56 µA et peut fournir les 500 mA de démarrage du Wifi dont l'ESP32-C3 a besoin. J'ai conçu le circuit de telle sorte que le LN2266 tire son énergie des deux piles via un MOSFET à canal P configuré pour protéger les piles si elles sont insérées dans le mauvais sens. La tension de la pile est également connectée via un filtre RC à l'une des broches CAN de l'ESP32-C3 afin qu'il puisse évaluer l'état des piles. J'avais prévu un fusible de protection, mais il provoquait une chute de tension inacceptable, alors je l'ai remplacé par une résistance de 0 ohm sur mes circuits imprimés et je l'ai entièrement supprimé dans les fichiers de conception.

L'ESP32-C3 se présente sous la forme d'un module ESP32-C3-WROOM-02, visible à droite sur le schéma de la figure 2. Ce module contient le microprocesseur Wifi et 4 Mo de mémoire flash, ainsi que tous les composants RF nécessaires, y compris une antenne imprimée. Pour le programmer, j'ai ajouté un bornier interne, sur lequel je peux souder un câble USB. Le convertisseur USB-JTAG-série interne s'occupe du reste. J'ai ajouté une fonction de mise à jour OTA (*Over the Air*) du micrologiciel à partir de mon serveur, qui m'offre la possibilité d'effectuer des mises à jour sur des unités qui sont déjà fermées et en service.

Ensuite, il y a l'écran à encre électronique. Il est connecté au circuit imprimé à l'aide d'un câble plat flexible, et il a besoin de quelques composants pour fonctionner, comme on peut le voir en bas à gauche de la figure 2 : un MOSFET, un inducteur, et quelques condensateurs et diodes pour générer les tensions dont il a besoin, quelques condensateurs de découplage et une ou deux résistances. L'écran dispose également d'une connexion pour un capteur de température LM75 externe. J'en ai prévu un, mais le micrologiciel actuel ne l'utilise pas, car l'écran fonctionne tout aussi bien sans lui.

Tout cela est placé sur un circuit imprimé très spacieux, dont la sérigraphie côté composants est illustrée à la **figure 3**. Comme l'écran à encre électronique est en

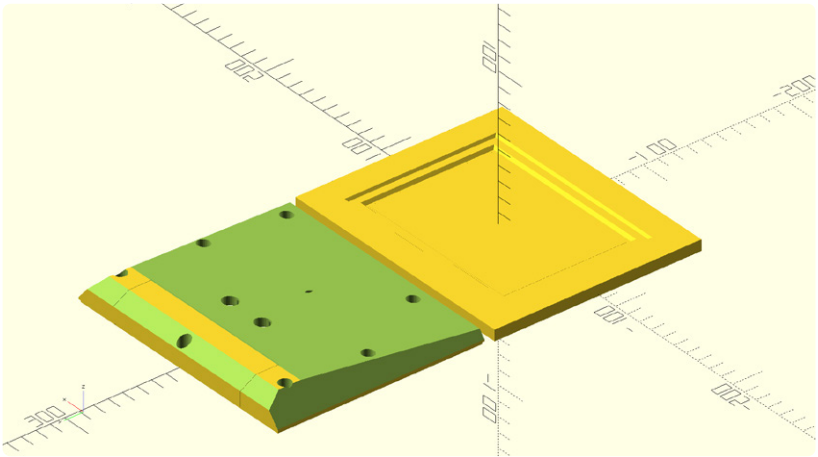

Figure 4. La structure du boîtier conçu avec OpenSCAD.

verre et quelque peu fragile, j'ai conçu le circuit imprimé pour qu'il serve de support et que le produit soit moins susceptible de se briser. Le circuit imprimé est un peu plus grand que cela, car les trous de montage, les composants traversants (tels que les contacts des piles) et l'antenne Wifi doivent tous être placés à l'extérieur de l'espace occupé par l'écran.

Le boîtier

Enfin, il y a le boîtier, que j'ai conçu avec OpenSCAD, comme le montre la **figure 4**. J'ai utilisé quelques astuces pour le rendre moins encombrant : il doit être assez épais pour loger les piles AA, et je ne voulais pas d'une grosse bosse dans le bas, alors j'ai chanfreiné les bords et donné une pente à l'arrière. Cela ne se voit pas si l'on regarde le cadre photo de face et rend son aspect plus élégant. Le cadre photo semblait également un peu encombrant par rapport à la zone visible de l'écran E-ink. Pour y remédier, j'ai ajouté un passe-partout. Comme ce passe-partout est imprimé 3D en blanc et qu'il contraste joliment avec le boîtier noir, il brise la « mer de noir » et donne un équilibre visuel à l'ensemble.

À l'arrière, il y a le logement des piles. Il s'ouvre en dévissant une vis (comme fermeture, je n'ai pas voulu me fier à de fragiles ergots imprimés en 3D) et, comme la sérigraphie du circuit imprimé indique l'orientation correcte des piles, leur remplacement devrait être facile.

Logiciel/micrologiciel et connexion Wifi

Comme l'ESP32-C3 est alimenté par des piles, j'ai essayé de faire en sorte qu'il en fasse le moins possible. Il doit se connecter au Wifi, communiquer avec mon serveur, voir s'il y a de nouvelles images qu'il n'a pas encore téléchargées et, si c'est le cas, les télécharger, déterminer quelle est la meilleure image à montrer (la plus récente ou celle qui a été montrée le moins souvent), l'afficher et s'éteindre. Évidemment, il y a d'autres choses à faire, comme la gestion des erreurs et la gestion des piles faibles.

Pour se connecter au Wifi, l'appareil a besoin d'un SSID et d'un mot de passe. Je n'ai pas voulu les coder en dur, au cas où ils devraient être changés. Ainsi, le micrologiciel inclut une copie de ESP32-WiFi-Manager [5], modifié pour corriger quelques bogues. Plus précisément, lorsque vous appuyez sur l'un des boutons à l'arrière du cadre photo tout en réinitialisant le cadre avec l'autre bouton, il démarre un point d'accès auquel vous pouvez vous connecter. Un serveur web intégré vous présente alors une interface utilisateur pour choisir un nouveau SSID et saisir son mot de passe.

Une fois que le cadre photo s'est connecté au Wifi, il tente de récupérer des instructions à partir d'une URL codée en dur. L'URL encode également certaines données d'état, telles que l'adresse MAC de l'appareil, la tension des piles et la version du micrologiciel installé. Le serveur renvoie la version du micrologiciel le plus récent pour l'appelant, ainsi qu'une liste des dix images les plus récentes. Certaines préférences sont également envoyées, telles que le fuseau horaire et l'heure à laquelle le cadre photo doit se réveiller pour tenter une mise à jour.

Le cadre photo dispose d'un espace de stockage pour dix images dans sa mémoire flash. Si le serveur dispose d'images non encore présentes localement, elles sont téléchargées et viennent écraser les images les plus anciennes. De cette manière, le cadre dispose toujours d'une réserve d'images relativement récentes, ce qui est utile en cas de dysfonctionnement de la connexion. Il permet même de déplacer le cadre photo d'un endroit à un autre : bien qu'en l'absence de connexion Wifi, il réutilise les anciennes photos, il aura toujours quelque chose de différent à afficher chaque jour.

La plupart des logiciels côté serveur ne sont pas très compliqués : il y a une page web frontale simple créée autour de *Cropper.js* [6], que vous pouvez ouvrir sur votre téléphone ou votre PC. Elle vous permet de sélectionner une image et d'en découper la partie que vous souhaitez afficher sur le cadre. Côté client, un peu de code JavaScript découpe et redimensionne l'image et envoie le résultat au serveur, lequel prend ces données, les convertit en données brutes pour l'écran, qu'il stocke dans une base de données MariaDB.

Lorsqu'un cadre photo se connecte, le serveur stocke les données reçues, de sorte que je dispose d'un journal des tensions des piles et que je peux voir si une mise à jour du micrologiciel a effectivement eu lieu. Le serveur vérifie ensuite dans la base de données MariaDB les dix dernières images, ainsi que d'autres informations, telles que la dernière version du micrologiciel, encode le tout en JSON et l'envoie. Tout cela n'est pas très compliqué.

Diffusion d'erreur

La seule partie réellement compliquée est la conversion de l'image RVB en les sept couleurs très spécifiques que l'écran est capable d'afficher. Prenons l'image de la **figure 5**, par exemple.

Si nous voulions convertir cette image en noir et blanc,

▲

Figure 5. L'image utilisée pour les essais.

Figure 6. Premier essai de conversion, en utilisant 100 % de noir et 100 % de blanc.

Figure 7. Conversion en noir et blanc par un processus de diffusion.

Figure 8. L'image de la figure 7 avec l'ajout de valeurs RVB (voir texte).

Figure 9. L'image obtenue par application de la norme CIEDE2000.

nous pourrions simplement vérifier la luminance (clarté) de chaque pixel et, si elle est plus proche du noir, rendre le résultat noir ; si elle est plus proche du blanc, rendre le résultat blanc. En d'autres termes, nous prenons la « couleur » la plus proche (en limitant les « couleurs » à 100 % de noir et 100 % de blanc) et nous obtenons l'image représentée sur la **figure 6**.

De toute évidence, cette image ne ressemble guère à l'image originale. Même en noir et blanc, nous pouvons faire mieux en utilisant ce que l'on appelle la *diffusion des erreurs*. Chaque fois que nous mettons un pixel gris en noir ou en blanc, nous prenons la différence entre la luminance du pixel dans la photo originale et la luminance du pixel que nous affichons réellement sur l'écran (l'« erreur » dans « diffusion d'erreur ») et en ajoutons une fraction aux pixels voisins (la « diffusion »). Le processus de diffusion peut être réalisé de plusieurs manières, la méthode Floyd-Steinberg étant la plus courante, et il permet d'obtenir une très bonne image noir et blanc tramée, comme illustré à la **figure 7**.

Nous pouvons également l'utiliser pour notre écran à sept couleurs. Le problème est que la définition de la « couleur la plus proche » devient beaucoup plus compliquée, de même que la définition de l'« addition ». Même la définition du terme « couleur » est délicate, car, en l'absence de rétroéclairage, les couleurs perçues dépendent de la lumière ambiante : à la lueur de la flamme d'une bougie, vous ne verrez pas les mêmes couleurs qu'en plein soleil. Pour obtenir les couleurs, j'ai pris une lampe à température réglable, je l'ai réglée sur 4800 K (la température de couleur moyenne à laquelle je pense que l'écran sera regardé), j'ai affiché les 7 couleurs sous forme d'aplats rectangulaires sur l'écran à encre électronique que j'ai photographié. J'ai importé cette photo dans mon ordinateur et j'ai ajusté les couleurs manuellement jusqu'à ce que les couleurs affichées à l'écran soient aussi proches que possible de celles de l'écran à encre électronique. J'ai ensuite pris les valeurs RVB moyennes des sept couleurs et les ai entrées dans mon programme.

Parmi ces sept couleurs, pour obtenir la couleur la plus proche sur n'importe quel pixel de l'image source, nous avons besoin d'un moyen de comparer deux couleurs et, comme nous utilisons des couleurs réelles et pas seulement du noir et du blanc, nous ne pouvons plus nous contenter d'utiliser la luminance. Une façon rapide et pratique de comparer deux couleurs consiste à considérer

l'espace RVB (linéarisé) comme un espace tridimensionnel et à utiliser la distance euclidienne pour mesurer la proximité de deux couleurs. Avec ce modèle, il suffit d'ajouter les valeurs RVB pour ajouter des couleurs. Si nous modifions l'algorithme de Floyd-Steinberg pour adopter cette méthode de sélection de la couleur la plus proche, nous obtenons une image raisonnablement acceptable, comme le montre la **figure 8**.

Ce n'est déjà pas si mal ! Cependant, il apparaît quelques étranges artefacts de couleur. Le plus évident est que le pot de fleurs n'est pas de la même nuance de bleu : C'est parce que l'écran à encre électronique ne dispose tout simplement pas des couleurs nécessaires pour reproduire cette nuance, et aucun algorithme ne peut compenser cela. Mais il y a d'autres bizarreries sous la forme d'étranges bandes de couleur, par exemple dans l'ombre sous le bras gauche du singe, et le ventre du singe est plus orange que sur l'image originale.

L'approche par les espaces de couleurs

Le problème du calcul des différences de couleur dans l'espace RVB est que vos yeux ne travaillent pas réellement dans un espace RVB linéaire. La différence entre deux couleurs est en fait beaucoup plus difficile à définir, et il existe de nombreuses approches pour y parvenir. L'une des premières approches consistait à convertir les couleurs RVB dans l'espace colorimétrique CIELAB et à prendre la différence. Largement recommandée sur l'internet, cette approche ne donne pas vraiment de bons résultats si les couleurs ne sont pas complètement saturées. Pour moi, la meilleure approche a été d'appliquer la norme CIEDE2000 [7]. Il s'agit de l'un des modèles de perception les plus récents et, bien qu'il ne soit pas d'emploi facile, il donne les meilleurs résultats, comme le montre la **figure 9**. Heureusement que j'avais déjà décidé de faire ce calcul laborieux côté serveur : côté cadre, il aurait coûté cher en piles !

Notez que les bandes de couleur dans les ombres ont disparu et que le ventre du singe est d'un rouge plus précis. L'image présente encore des défauts, comme la couleur du pot de fleurs, mais, comme je l'ai déjà mentionné, c'est faute des couleurs nécessaires pour afficher correctement cette teinte particulière.

Pour plus de rapidité, toute cette logique est codée dans un simple programme C. Après avoir été éditée par un utilisateur (sur une page web utilisant *Cropper.js*), l'image est envoyée au serveur qui la soumet à ce programme qui la convertit en pixels d'écran à encre électronique lesquels sont stockés dans une table MariaDB, à la disposition des cadres photo à chaque fois qu'ils se connectent.

La page web est également utilisable sur téléphone mobile, de sorte que s'il vous arrive de prendre une image particulièrement belle, vous pouvez l'éditer immédiatement et la mettre en file d'attente pour la distribuer à tous les cadres photos (**figure 10**).

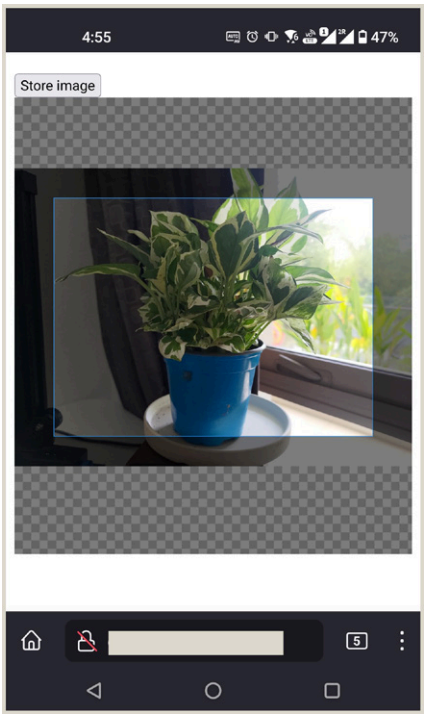

Figure 10. Édition d'image pour une utilisation sur téléphone mobile.

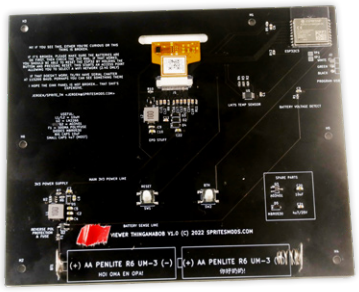

Figure 11. Circuit imprimé complètement assemblé.

Figure 12. Les deux parties du boîtier réalisé pour ce projet.

Résultat

Une fois le circuit imprimé réalisé et les pièces imprimées en 3D, il est temps d'assembler le tout, comme le montre la **figure 11**.

L'arrière du circuit imprimé contient toute l'électronique. Le circuit imprimé est en fait conçu pour être réparable : s'il se casse et que je ne suis pas là pour le réparer, il se peut que j'aie un ou deux amis qui s'y connaissent en électronique et qui peuvent y jeter un coup d'œil. Cela signifie qu'il y a des instructions de débogage à l'arrière et même quelques composants de rechange. Sinon, le circuit imprimé est assez peu peuplé : ses dimensions sont essentiellement définies par la taille de l'écran à encre électronique situé de l'autre côté. L'espace disponible m'aurait permis d'utiliser des composants plus gros, mais j'ai une multitude de bobines de composants 0603, qu'il faut bien que je consomme ; avec un bon fer à souder et un microscope binoculaire, je suis parfaitement à l'aise pour les souder à la main.

Le boîtier est illustré à la **figure 12** avec le passe-partout

Figure 13. Le compartiment à piles avec son couvercle.

Figure 14. Le résultat final, au pied de l'original.

blanc à l'intérieur. Celui-ci comporte une découpe pour l'écran à encre électronique : Même si l'adhésif qui fixe l'écran au circuit imprimé venait à se décoller, il resterait en place. Le boîtier se ferme à l'aide de vis qui sont vissées dans des inserts en laiton. Ces inserts sont mis en place en les chauffant à l'aide d'un fer à souder (avec une panne usagée – je ne veux pas endommager une panne utilisable) ce qui fait fondre le plastique.

Tout est assemblé à l'aide d'une série de vis M3. Le compartiment des piles est doté d'un petit couvercle (**figure 13**), de sorte qu'il n'est pas nécessaire de démonter l'ensemble pour remplacer les piles. D'après mes calculs, les piles devraient durer plus d'un an. Notez également qu'il y a deux boutons à l'arrière. L'un est connecté directement au reset de l'ESP32, et l'autre à un GPIO général. Vous pouvez utiliser le bouton de réinitialisation pour que l'appareil effectue un cycle manuel de connexion et de rafraîchissement, ce qui a pour effet secondaire d'afficher l'image suivante. Lorsque l'autre bouton est maintenu enfoncé pendant que le cadre photo est réinitialisé, il démarre la fonction point d'accès qui vous permet de vous connecter au cadre et de reconfigurer les paramètres de la connexion Wifi.

Enfin, sur la **figure 14**, vous pouvez admirer le résultat final. Cette image n'est pas la plus fidèle qui soit, mais c'est largement compensé par l'affichage d'une nouvelle photo chaque jour depuis l'autre bout du monde.

Comme d'habitude, ce projet est open source : avec une imprimante 3D, un accès à un serveur et quelques compétences, vous pouvez réaliser votre propre version de ce cadre. Tous les sources, les dessins des circuits imprimés, etc. sont disponibles sur GitHub [8].

VF : Helmut Müller — 230464-04

Produits

- **Waveshare 5.65" ACeP 7-Color E-Paper E-Ink Display Module (600×448)**
  www.elektor.fr/19847
- **ESP32-DevKitC-32E**
  www.elektor.fr/20518
- **Dogan Ibrahim,** *The Complete ESP32 Projects Guide,* **Elektor 2019**
  www.elektor.fr/18860

Questions ou commentaires ?

Envoyez un courriel à l'auteur (jeroen@spritesmods.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Jeroen Domburg est Senior Software and Technical Marketing Manager chez Espressif Systems. Avec plus de 20 ans d'expérience dans le domaine de l'embarqué, il est participe au processus de conception logicielle et matérielle des SoC d'Espressif. Pendant son temps libre, il aime bricoler avec l'électronique pour réaliser des appareils qui ont une utilité pratique ou pas !

LIENS

[1] Cadre photo papier électronique 7 couleurs de GitHub : https://github.com/robertmoro/7ColorEPaperPhotoFrame
[2] Cadre de papier électronique à Raspberry Pi : https://reddit.com/r/raspberry_pi/comments/10dbhnj/i_built_a_raspberry_pi_epaper_frame_that_shows_me
[3] Projet d'image papier électronique sur YouTube : https://youtu.be/YawP9RjPcJA
[4] Graphiques de décharge : https://powerstream.com/AA-tests.htm
[5] ESP32-WiFi-Manager : https://github.com/tonyp7/esp32-wifi-manager
[6] Cropper.js : https://fengyuanchen.github.io/cropperjs
[7] Norme CIEDE2000 : https://en.wikipedia.org/wiki/Color_difference#CIEDE2000
[8] Dépôt GitHub : https://github.com/Spritetm/picframe_colepd