caméra ESP32

si simple qu'on n'a même pas besoin du wifi

Bera Somnath (Inde)

Vous avez une idée d'application avec une caméra ? Avant d'en acheter une nouvelle en magasin, pourquoi ne pas en bricoler une vous-même ? Avec une carte ESP32 équipée d'une caméra et quelques composants supplémentaires, vous pouvez construire une solution personnalisée simple, mais efficace.

Tous les smartphones peuvent prendre et stocker des photos et les afficher. Par conséquent, personne n'aurait sérieusement l'idée de fabriquer encore un autre appareil photo, et certainement pas moi. Sauf si c'est vraiment facile à faire. Avec quelques modules peu coûteux comme une carte ESP32 avec caméra, un petit écran OLED, une horloge en temps réel et un mécanisme de déclenchement, vous pouvez en faire un qui fonctionne sur batterie et prendra des photos entièrement sous votre contrôle. En appuyant sur un bouton ou par détection de présence avec un capteur de mouvement ou même déclenché par un capteur de température sans contact, il prend (discrètement) la photo de quelqu'un dont la température corporelle est trop élevée. De telles applications ont du sens car vous ne pouvez utiliser votre smartphone pour cela.

Caméra ESP32

En ajoutant un capteur IRP à la carte ESP32-Cam, vous pouvez la configurer pour prendre secrètement des photos d'intrus ! Ou bien, en utilisant uniquement la caméra ESP32 et l'écran OLED, vous pouvez réaliser une petite webcam qui se connecte à n'importe quel réseau disponible dans le voisinage et publie ensuite le flux vidéo sur l'Intranet. De même, grâce à la fonction de redirection de port de votre modem ou routeur, vous pouvez publier l'image sur l'Internet. C'est une fonction très pratique pour ceux qui ont besoin d'une deuxième caméra pour leurs webinaires et/ou cours en ligne. Le module ESP32-Cam que nous utiliserons dans cet article coûte environ 10 €.

Une démo sympa pour commencer

Après avoir installé la dernière version de l'EDI Arduino et le paquetage de cartes ESP32 le plus récent en utilisant le Gestionnaire de cartes de l'EDI, sélectionnez comme carte la carte «AI Thinker ESP32-CAM» (Outils -> Type de carte -> ESP32 Arduino) et le port auquel elle est connectée (Outils -> Port). Ouvrez ensuite l'exemple de projet de la webcam ESP32 (Fichier -> Exemples -> ESP32 -> Caméra -> Serveur Web). Essayez-le, vous serez surpris. Les deux projets qui suivent sont des extensions de ce projet.

Projet 1 : Webcam avec écran OLED

Le projet webcam ne permet pas de sélectionner un réseau wifi à la volée (les informations d'identification du réseau sont codées en dur dans le programme) et la seule façon de connaître l'adresse IP de la webcam est de la connecter à un PC puis d'utiliser un terminal série. Ici, nous améliorons ce projet pour pouvoir se connecter à n'importe quel réseau disponible à proximité. Une fois la webcam connectée, le petit écran OLED indique le réseau auquel elle est connectée et son adresse IP. Si elle se déconnecte pour une raison quelconque, elle essaie de se reconnecter à l'un des réseaux connus.

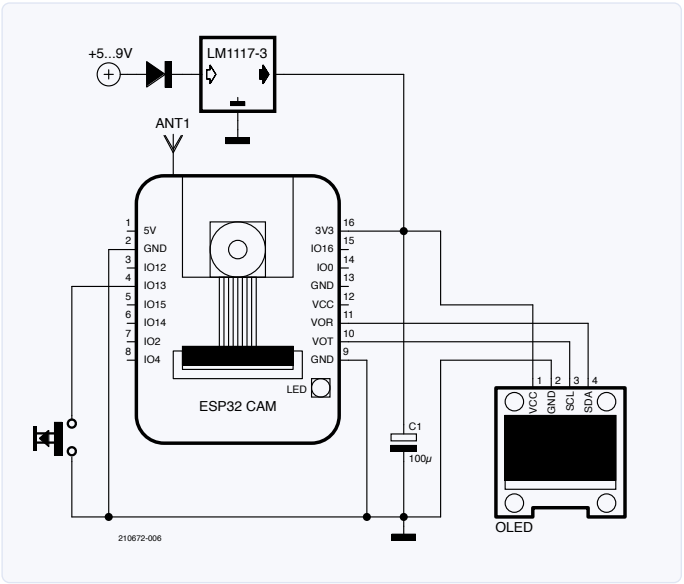


Figure 1. Ajoutez un écran OLED à la caméra ESP32-Cam pour plus de commodité.

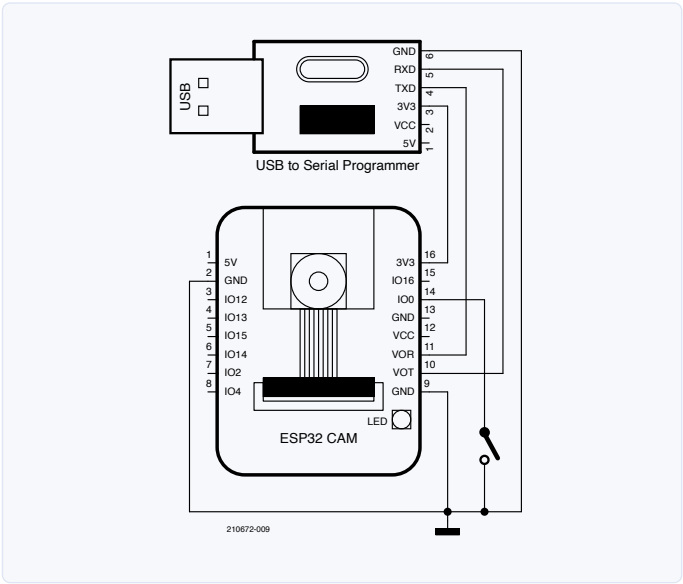


Figure 2. L'ESP32-Cam n'a pas d'interface USB-série intégrée. Par conséquent, pour télécharger le croquis, il faut disposer d'un adaptateur USB-série. Pour entrer dans le mode de chargement du programme, il faut appuyer sur le bouton RST de l'ESP32 (ou l'éteindre et le rallumer) pendant que IO0 est forcé à la masse.

Fabrication de la Webcam

La **figure 1** montre comment connecter l'écran OLED à la carte ESP32-Cam. Il reste très peu de broches GPIO utilisables après la connexion de la carte SD et de la caméra. Les seules broches libres disponibles pour notre utilisation sont IO1, IO3, IO4, IO12, IO13 (voir encadré « Notes du labo d'Elektor »). IO1 (« U0T ») et IO3 (« U0R ») sont également utilisées comme port série pour télécharger le programme vers l'ESP32 en mode de programmation flash. Par conséquent, ces broches doivent être libres de toute connexion pendant le téléchargement des programmes.

Dans ce projet de webcam, nous pouvons facilement utiliser les broches réservées à la carte SD car nous n'utilisons pas cette fonction. Cependant, nous avons utilisé IO1 et IO3 comme broches I²C pour connecter l'écran OLED. Nous avons ainsi maximisé le nombre de broches GPIO disponibles. N'oubliez pas de déconnecter le bus I²C lorsque vous téléchargez un programme sur l'ESP32 (**figure 2**).

Utilisation

Le principe de fonctionnement de la webcam ESP32 avec écran OLED est resté très simple. L'ESP32 recherche les réseaux wifi fournis sous forme de liste dans le programme et essaie de se connecter à l'un d'eux. Dès que la connexion réussit, le statut « camera ready » s'affiche sur l'écran OLED avec le SSID du réseau et l'adresse IP attribuée à la webcam. Pour accéder à la caméra, connectez-vous au réseau indiqué et faites pointer un navigateur sur l'adresse IP de la webcam. Par défaut, l'adresse du port reste 8080.

On peut télécharger le logiciel pour la webcam ESP32 avec écran OLED à partir de [1]. Notez qu'il nécessite la bibliothèque « *ESP8266 and ESP32 OLED driver for SSD1306 displays* » (nous avons utilisé la version 4.1.0) [2]. Elle peut être installée avec le gestionnaire de bibliothèque de l'EDI Arduino. N'oubliez pas de déconnecter le bus I²C avant de télécharger un croquis.

Projet 2 : Caméra à déclenchement avec OLED & RTC

Dans ce projet, une photo est prise chaque fois qu'un événement se produit. Les photos sont stockées sur une carte SD avec un numéro de série et un horodatage.

Le circuit (**figure 3**) utilisé est une extension du circuit précédent. Un module d'horloge en temps réel (RTC) DS3231 (ou DS1307) est ajouté sur le bus I²C, et une source de déclenchement est connectée à IO13. Il peut s'agir d'un bouton poussoir ou d'un capteur PIR

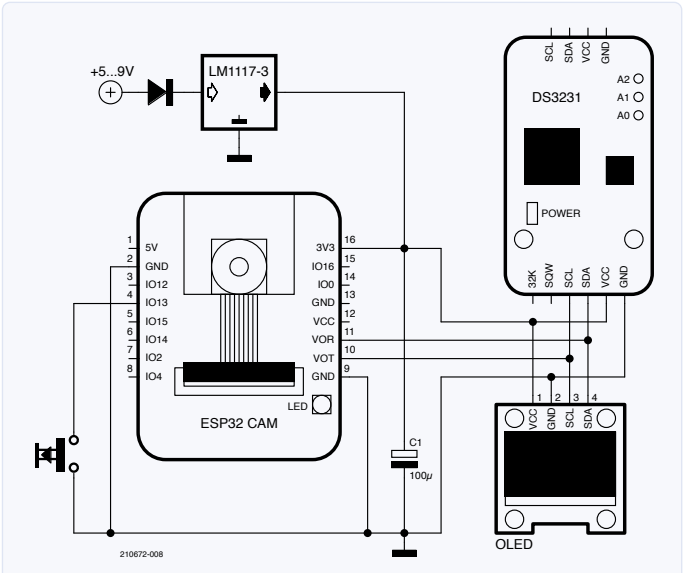


Figure 3. Un bouton poussoir et un module d'horloge en temps réel transforment la webcam de la figure 1 en une caméra de surveillance à déclenchement et horodatage.

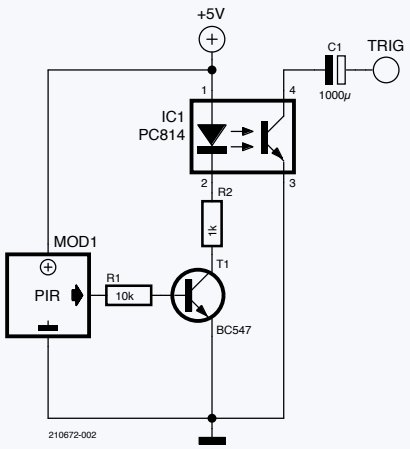


Figure 4. On peut utiliser un capteur IRP en 5 V - certains fonctionnent même en 3,3 V - pour déclencher la caméra ESP32. Cependant, pour l'isoler complètement, nous recommandons d'utiliser un optocoupleur comme celui-ci (ou un relais).

(**figure 4**) ou de tout autre élément qui génère une courte impulsion active-basse lorsqu'une photo doit être prise.

IO4 est utilisé comme flash, en utilisant la LED blanche super lumineuse intégrée de l'ESP32-Cam comme source d'éclairage supplémentaire pour la caméra. IO13 est utilisé comme broche de déclenchement pour prendre des photos. Le programme appose la date et l'heure sur le nom du fichier et le stocke sur la carte SD. Il passe ensuite en mode de sommeil profond pour préserver la batterie pendant les périodes d'inactivité où aucune photo n'est prise. Une impulsion active-basse sur IO13 réveille la caméra pour prendre une nouvelle photo.

Ce programme se compose uniquement de la fonction `setup` ; la fonction `loop` n'est pas utilisée. Ceci est dû à l'utilisation du mode de sommeil profond. Lorsque l'appareil se réveille de ce mode, il redémarre, exécute la fonction `setup` et prend une photo avant de retourner au mode de sommeil profond, de sorte que la fonction `loop` n'est jamais atteinte.

On peut télécharger le logiciel pour la caméra à déclenchement à partir de [1]. En plus de la bibliothèque de pilotes OLED requise pour le premier projet [2], ce projet nécessite également la bibliothèque « Rtc by Makuna » (nous avons utilisé la version 2.3.5) [3]. Cette bibliothèque peut également être installée avec le gestionnaire de bibliothèque de l'EDI. N'oubliez pas de déconnecter le bus I²C avant de télécharger un croquis (**figure 2**).

Conclusion

La caméra ESP32-Cam est très bon marché tout en offrant de vastes possibilités. En alignant plusieurs de ces caméras, on peut s'en servir comme scanner fixe et grand angle (50 à 100 mètres) pour prendre des photos des plaques d'immatriculation des véhicules qui s'approchent afin de les contrôler (pour la vitesse par ex.). En mode solo, avec un capteur PIR, elle peut servir à « capturer » des oiseaux ou d'autres animaux. On peut aussi imaginer des applications plus élaborées. Par exemple, en ajoutant la détection de visage et un scanner de température sans contact, elle peut contribuer à une certaine sécurité anti-COVID.

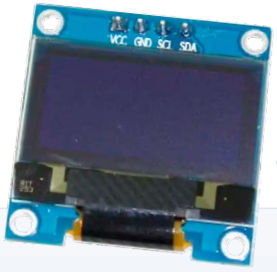
210672-04 — VF : Denis Lafourcade

Des questions, des commentaires ?

Contactez Elektor (redaction@elektor.fr).

COMPOSANTS

Carte ESP32-Cam
Carte SD
Écran OLED de 0,96 pouce compatible I²C avec SSD1306
Module RTC DS3231 / DS1307 I²C
Adaptateur USB-série (câble)
Régulateur 3,3 V LM1117
Capteur PIR ou bouton poussoir
En option : batterie, condensateurs, etc.



PRODUITS

- **Carte de développement ESP32-Cam-CH340(SKU 19333)**
  www.elektor.fr/19333
- **Écran OLED de 0,96 pouce compatible I²C (SKU 18747)**
  www.elektor.fr/18747
- **Module RTC de SparkFun – RV-8803 (Qwiic) (SKU 19646)**
  www.elektor.fr/19646

LIENS

[1] Téléchargements pour cet article à Elektor Labs : https://www.elektormagazine.fr/labs/esp32-camera
[2] Pilote OLED ESP8266 et ESP32 pour écrans SSD1306 : https://github.com/ThingPulse/esp8266-oled-ssd1306
[3] Rtc par Makuna : https://github.com/Makuna/Rtc
[4] Caméra à déclenchement version Elektor Labs : https://www.elektormagazine.com/210672-01

Notes du labo d'Elektor

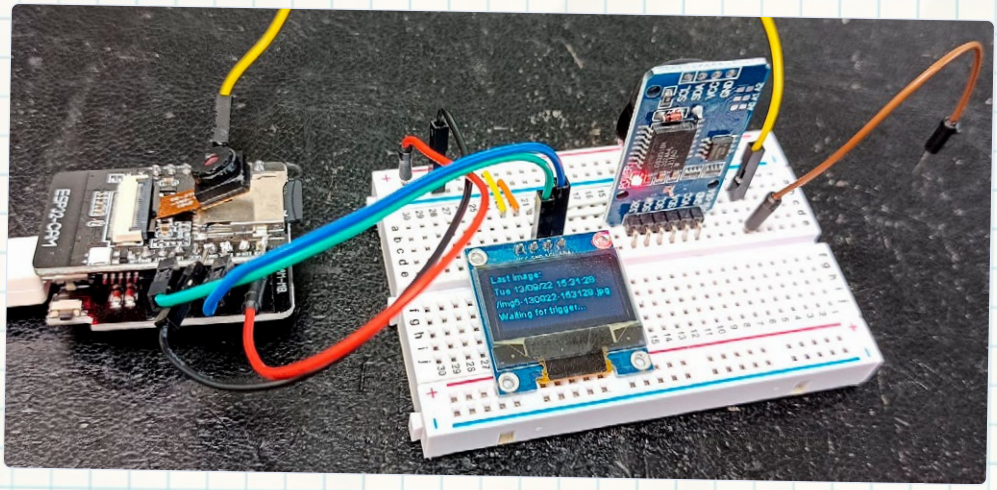


Figure 5. Le prototype Elektor Labs construit sur une platine d'expérimentation. L'ESP32-Cam avec la carte fille est à gauche, le module vertical à droite est la RTC.

De nombreuses cartes ESP32-Cam sont vendues avec une ingénieuse carte fille convertisseur USB-série* (on devrait peut-être l'appeler carte mère puisque le module ESP32 s'y branche et en tire son alimentation). Ce combo (figure 5) rend la programmation de l'ESP32 très facile, mais il entre en conflit avec le port I²C sur les ports IO1 et IO3 tels qu'utilisés dans le projet de l'auteur. En utilisant ces broches, le port I²C doit être déconnecté à chaque fois que l'on veut reprogrammer l'ESP32, et nous avons donc cherché un autre moyen de connecter le bus I²C.

En théorie, sur l'ESP32, le bus I²C peut être configuré pour utiliser presque n'importe laquelle de ses broches d'entrée/sortie, mais cette configuration sur la carte ESP32-Cam entraîne toutes sortes de problèmes de démarrage et d'erreurs de mémoire PSRAM. Ceci à cause des résistances de rappel sur le bus I²C.

La carte ESP32-Cam possède deux connecteurs d'extension à 8 voies. À première vue, cela peut sembler correct, mais en y regardant de plus près, les choses ne sont pas si simples. Tout d'abord, six des seize broches sont utilisées pour l'alimentation, ce qui n'en laisse que dix pour les ports d'entrée/sortie. Six d'entre eux (IO2, IO4, IO12, IO13, IO14 et IO15) sont partagés avec le connecteur de la carte SD. Il reste donc les ports IO0, IO1, IO3 et IO16.

Le téléchargement d'un programme sur l'ESP32 avec la carte fille attachée nécessite les ports IO0, IO1 et IO3.

IO16 est partagée avec la broche CS de la puce PSRAM, elle ne peut donc être utilisée en toute sécurité que lorsque la PSRAM n'est pas nécessaire.

La bibliothèque de la carte SD utilise par défaut le mode de données 4 bits, mais elle peut être mise en mode de données 1 bit avec

```
SD_MMC.begin(«/sdcard»,true);
```

Cela libère les ports IO4, IO12 et IO13. Le port 4 est partagé avec la (lumineuse) LED flash blanche. Il possède une résistance de rappel haut de 47 kΩ et une résistance de rappel bas de 10 +1 kΩ qui pilote un transistor. Le port 13 possède également une résistance de rappel de 47 kΩ et est utilisé dans ce projet comme entrée de déclenchement de la caméra.

IO12 est l'une des broches de l'ESP32 qui doit être manipulée avec précaution à la mise sous tension car elle détermine la valeur de VDD_SDIO, la tension d'interface de la mémoire flash. Elle doit être forcée vers le bas à la mise sous tension de la carte ESP32-Cam.

Après quelques tâtonnements, nous avons découvert que le bus I²C peut être déplacé en toute sécurité vers les ports IO0 (SDA) et IO3 (SCL, ou l'inverse. Peu importe tant que le logiciel est configuré correctement). En utilisant ces ports, le bus I²C n'interfère pas avec le port de programmation série ni avec le mode de démarrage, car IO0 et IO3 sont censés être à l'état haut à la mise sous tension.

Enfin, et sans rapport avec ce qui précède, sachez que IO33 est connecté à une LED rouge du côté du module ESP32 de la carte, utilisable comme bon vous semble (tant que vous gardez à l'esprit qu'elle est active à l'état bas).

Figure 6. Ça marche, bravo !

Nous nous sommes un peu emportés avec le deuxième projet et avons modifié le programme (figure 6). Il n'utilise plus l'EEPROM émulée pour stocker le numéro de l'image mais un fichier sur la carte SD. Il suffit de supprimer le fichier « counter.txt » pour remettre le compteur à zéro. On peut télécharger notre programme à partir de [4].

** Voir l'encadré Produits connexes pour une autre version du module ESP32-Cam.*