talkie-walkie avec ESP-NOW

pas tout à fait Wi-Fi, pas tout à fait Bluetooth non plus…

Clemens Valens (Elektor)

Imaginez que votre projet sans fil nécessite à la fois des temps de réponse rapides et une longue portée ? Le Wi-Fi et le Bluetooth ne conviennent pas dans ces cas. L'ESP-NOW est peut-être une bonne alternative ? Les connexions sont établies presque instantanément et des portées de plusieurs centaines de mètres sont possibles. Dans cet article, nous l'essayons dans une application simple de talkie-walkie ou d'interphone sans fil.

L'ESP32 d'Espressif est souvent utilisé pour ses compétences Wi-Fi et Bluetooth, des domaines dans lesquels il excelle. Ce sont d'excellents protocoles pour toutes sortes d'applications sans fil, mais ils ont leurs limites.

L'un des inconvénients du Wi-Fi est le temps nécessaire pour établir une connexion. En outre, il ne permet pas une communication directe (pair-à-pair, **figure 1**) entre les appareils. Un routeur est toujours impliqué. Pour cette raison, ce n'est pas vraiment adapté aux simples télécommandes à faible latence permettant d'ouvrir une porte de garage ou d'allumer et d'éteindre une lumière. Ces tâches nécessitent une réponse immédiate. Pour contourner ce problème, les applications Wi-Fi ont tendance à être allumées et connectées en permanence. Elles consomment donc beaucoup d'énergie, même lorsqu'elles sont inactives.

Le Bluetooth, quant à lui, se caractérise par un établissement rapide de la connexion et une communication de pair-à-pair (aussi poste-à-poste ou *peer-to-peer* en anglais, ou encore P2P), et est excellent pour les télécommandes à faible latence. Toutefois, il est destiné à des applications à courte portée, avec des appareils communicants espacés d'une dizaine de mètres. Il est vrai que le Bluetooth à longue portée existe, mais il n'est pas encore très répandu.

La solution : ESP-NOW

Le protocole sans fil ESP-NOW [1] d'Espressif est une solution pour les situations qui nécessitent à la fois des temps de réponse rapides et une longue portée tout en utilisant la même bande de fréquence que le Wi-Fi et le Bluetooth.

Le protocole combine les avantages du Wi-Fi et du Bluetooth. L'ESP-NOW est destiné à la domotique et à la maison intelligente. Comme il permet des topologies un-à-plusieurs (*one-to-many*) et plusieurs-à-plusieurs (*many-to-many*) (**figure 2**), il n'a besoin ni de routeur, ni de passerelle, voire pire, de cloud.

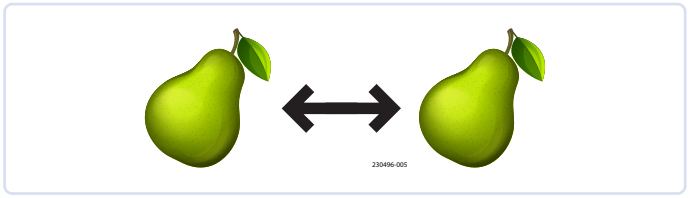

Figure 1. La communication de pair-à-pair telle qu'illustrée ici n'est pas possible avec le Wi-Fi ; un routeur est toujours nécessaire entre les deux nœuds.

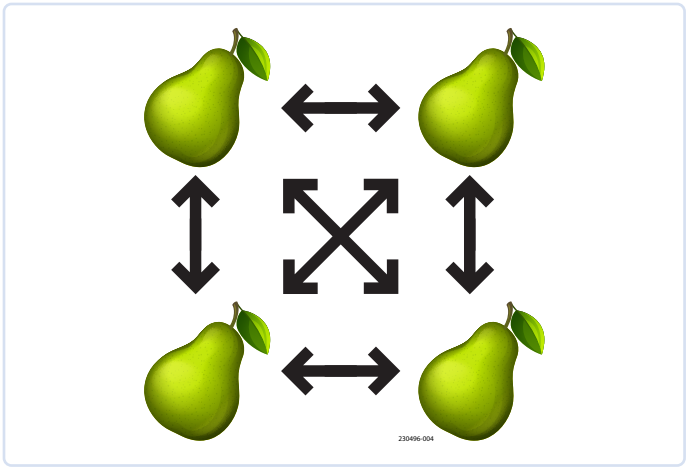

Figure 2. L'ESP-NOW prend en charge les réseaux de plusieurs-à-plusieurs. Dans un tel réseau, chaque nœud peut communiquer directement avec les autres nœuds sans passer par un routeur.

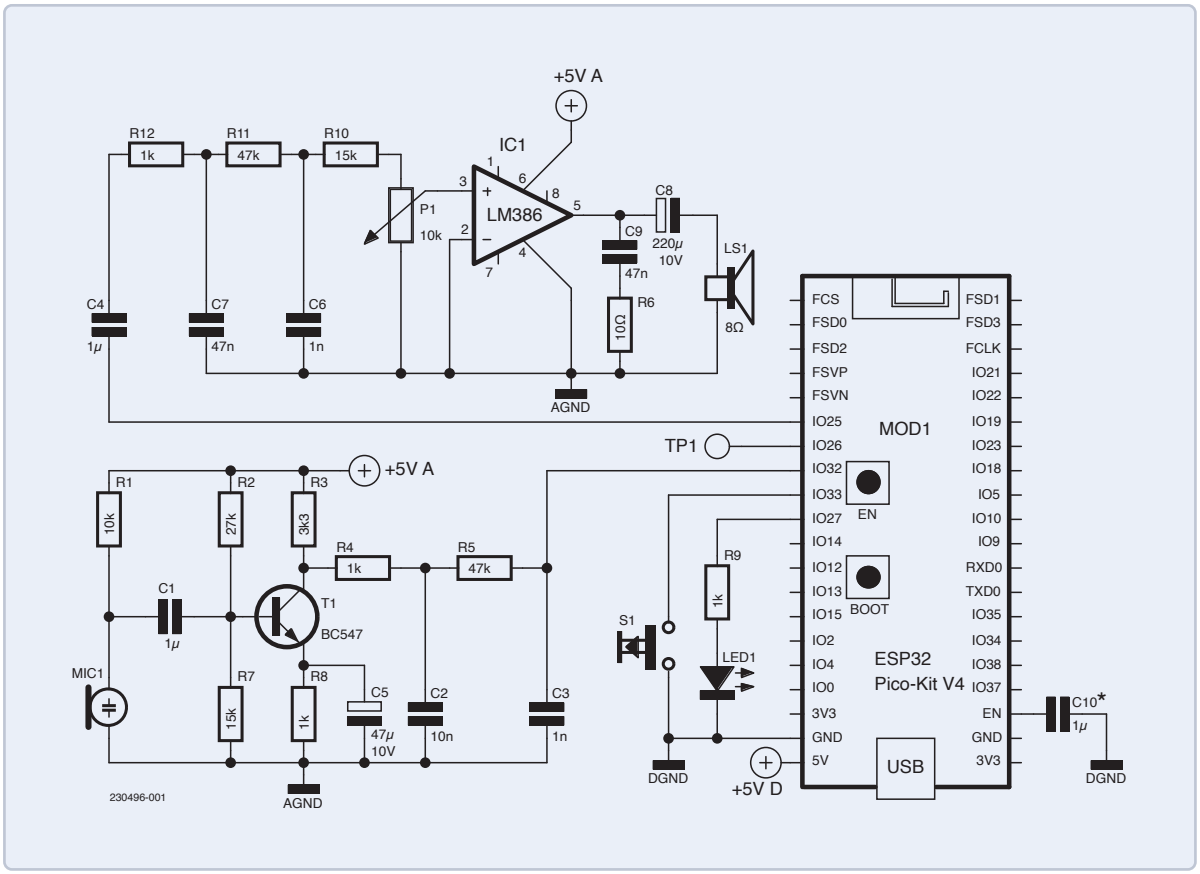

Figure 3. Un simple préamplificateur de microphone à l'entrée et un LM386 classique comme amplificateur de puissance à la sortie. Notez comment les alimentations des parties analogiques et numériques sont séparées.

L'ESP-NOW ne met pas en œuvre de connexion fantaisiste ou de protocoles de communication complexes. L'adressage est basé sur l'adresse MAC Ethernet du nœud, et une étape d'appariement est nécessaire pour que les nœuds puissent communiquer entre eux. De plus, il n'est pas garanti que les paquets de données arrivent dans l'ordre. Pour des applications simples de commande à distance, tout cela suffit.

Le débit de données de l'ESP-NOW est de 1 Mbit/s par défaut (configurable), et un paquet de données peut avoir une charge utile allant jusqu'à 250 octets. Avec les octets d'en-tête et de somme de contrôle, etc., cela donne une taille maximale de paquet de 255 octets.

Construisons un talkie-walkie

Mon objectif était de créer un appareil de type talkie-walkie ou un interphone basé sur l'ESP-NOW. Un rapide coup d'œil aux spécifications de l'ESP32 montre qu'il intègre tout ce qu'il faut pour cela : un convertisseur analogique-numérique (CAN), un convertisseur numérique-analogique (CNA), beaucoup de puissance de calcul et, bien sûr, tous les éléments de la radio. Dans la pratique, cependant, les choses sont un peu moins roses.

Le CAN de 12 bits s'avère plutôt lent, j'ai mesuré une fréquence d'échantillonnage maximale d'environ 20 kHz. Quelque part sur internet, il a été mentionné que sa bande passante analogique n'est que de 6 kHz. Le CNA a une largeur de huit bits (mais il y en a deux), ce qui limite encore plus la qualité audio possible.

Cependant, un talkie-walkie peut s'en sortir avec ces chiffres si la bande passante audio est limitée à la bande passante standard de téléphonie de 3,5 kHz. Une fréquence d'échantillonnage de 8 kHz donne un débit de données de (8 000 / 250) × 255 × 8 = 65 280 bit/s (n'oubliez pas que la taille maximale de la charge utile est de 250 octets). C'est bien inférieur au débit par défaut de 1 Mbit/s. Ces spécifications ne nous permettront pas d'obtenir un son très fidèle, mais ce n'est de toute façon pas notre objectif. L'intelligibilité est plus importante.

Le circuit

Pour garder les choses simples, j'ai utilisé un préamplificateur de microphone à condensateur à bande limitée à un transistor comme entrée audio et j'ai ajouté un amplificateur classique à base de LM386 comme sortie audio. Le schéma est montré dans la **figure 3**. La bande passante d'entrée est limitée à l'extrémité basse par C1 et C5, qui sont légèrement sous-dimensionnés. L'extrémité haute est limitée par les filtres passe-bas R4/C2 et R5/C3. Des filtres passe-bas similaires sont placés à la sortie du CNA. Le signal sur le côté chaud de P1 ne doit pas être supérieur à 400 mV_{PP}.

Comme module ESP32, j'ai opté pour l'ESP32-PICO-KIT. Il existe de nombreux autres modules, mais ils n'exposent pas toutes les sorties CNA sur GPIO25 et GPIO26. Nous avons également besoin d'une entrée CAN. J'ai utilisé GPIO32 pour cela, qui correspond à ADC1, canal 4. Le point de test TP1 sur GPIO26 (la deuxième sortie CNA) est fourni comme une sortie de contrôle pour le signal du microphone. Un bouton poussoir sur GPIO33 fournit la fonctionnalité *push-to-talk* (PTT, appuyer pour parler en français), et la LED sur GPIO27 est la LED multifonction obligatoire d'un circuit à microcontrôleur.

Notez que l'alimentation est divisée en une partie analogique et une partie numérique. La raison n'est pas d'éviter le couplage du bruit de commutation numérique à haute vitesse dans l'entrée audio, mais d'éviter un cliquetis dans la sortie. Apparemment, une tâche exécutée sur l'ESP32 produit des perturbations périodiques qui peuvent devenir audibles lorsque le circuit n'est pas câblé avec soin. La meilleure façon que j'ai trouvée pour éviter cela est d'utiliser deux alimentations séparées (**figure 4**). Le module ESP32 doit être traité comme un composant qui a besoin d'une alimentation (tout comme le LM386), et non comme un module qui peut aussi fournir de l'énergie au reste du circuit ; dans cette application, il ne le peut pas. Gardez à l'esprit que le LM386 a une plage d'alimentation de 4 V à 12 V.

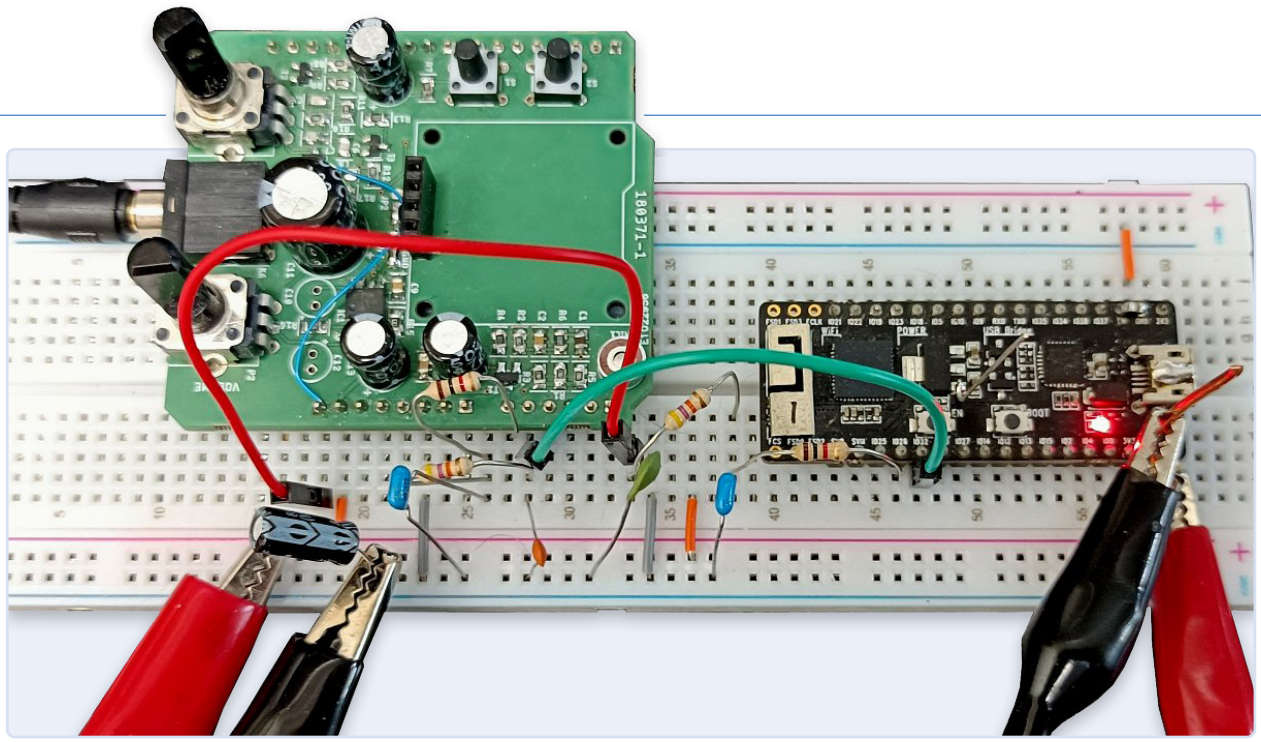

Figure 4 . Une preuve de concept construite sur une carte avec un ESP32-PICO-KIT et un Elektor Snore Shield [2] légèrement modifié pour les amplificateurs d'entrée et de sortie. Notez les deux paires de pinces crocodiles qui fournissent les alimentations analogiques et numériques séparées.

Le condensateur C10 est optionnel et n'est nécessaire que dans quelques rares cas de modules ESP32 anciens qui ne démarrent pas correctement lorsqu'ils ne sont pas connectés à un ordinateur (ou autre). Il se trouve que j'ai quelques-uns de ces premiers modules, et j'ai donc inclus le C10 dans mon design.

Le logiciel

J'ai basé le programme du talkie-walkie sur l'exemple `ESPNow_Basic_Master` fourni dans la bibliothèque d'Espressif pour Arduino. Après l'avoir adapté à mes besoins, j'y ai ajouté l'échantillonnage et la lecture audio. Il y a plusieurs choses que vous voudrez peut-être savoir à propos du programme.

L'échantillonnage et la lecture audio sont contrôlés par une interruption de minuterie fonctionnant à 8 kHz. Pour l'échantillonnage, la routine de service d'interruption (ISR) de la minuterie lève uniquement un drapeau pour signaler qu'un nouvel échantillon doit être acquis. La fonction `loop()` surveille ce drapeau et prend les mesures nécessaires. En effet, le CAN ne doit pas être lu à l'intérieur d'un ISR lors de l'utilisation de la bibliothèque CAN fournie par Espressif. La fonction `adc1_get_raw()` utilisée ici appelle toutes sortes d'autres fonctions qui peuvent faire des choses sur lesquelles vous n'avez aucun contrôle. Comme le logiciel ESP32 fonctionne dans un environnement multi-tâche, il est important d'assurer la sécurité des fils d'exécution. Lorsque vous utilisez Arduino pour la programmation de l'ESP32, beaucoup de choses sont gérées pour vous, mais si vous envisagez de porter mon programme sur l'ESP-IDF, vous devrez peut-être être plus prudent.

La lecture audio est facile, car l'ISR de la minuterie d'échantillonnage écrit simplement un échantillon sur le CNA s'il y en a un de disponible. Si ce n'est pas le cas, il fixe la sortie du CNA à la moitié de l'alimentation de l'ESP32, soit 1,65 V. La seule chose à savoir ici est qu'un tampon dit ping-pong est utilisé pour rationaliser la réception audio numérique (**figure 5**). Un tel tampon se compose de deux tampons, dont l'un est rempli pendant que l'autre est lu. Cela permet un chevauchement. En théorie, cela ne devrait pas se produire puisque l'émetteur et le récepteur utilisent la même fréquence d'échantillonnage et la même logique de synchronisation, mais en réalité cela se produit en raison des tolérances de synchronisation. Un tampon double ou ping-pong permet d'éviter des clics gênants pendant la lecture. Notez que la réception de paquets de données dans le désordre n'est pas gérée.

Couplage

Le micrologiciel du talkie-walkie est un système maître-esclave. Le maître fonctionne en mode station Wi-Fi (STA), tandis que l'esclave est en mode point d'accès (AP). Le maître se connecte immédiatement à un esclave lorsqu'il en détecte un, et il peut commencer à envoyer des données immédiatement. Toutefois, lorsque le maître se connecte à l'esclave, ce dernier ne se connecte pas au maître. L'esclave ne peut pas envoyer de données au maître et la communication bidirectionnel n'est pas possible (du moins, je n'y suis pas parvenu ; si vous savez mieux, faites-le moi savoir).

Pour que l'esclave se connecte au maître, il est possible d'utiliser la fonction de rappel de réception de données. Lorsque des données sont reçues, l'adresse de l'expéditeur est transmise à cette fonction en même temps que les données. Ainsi, dès que l'esclave reçoit quelque chose, il peut se connecter à l'expéditeur des données. Pour ce faire, j'ai utilisé les mêmes fonctions et procédures que celles utilisées par

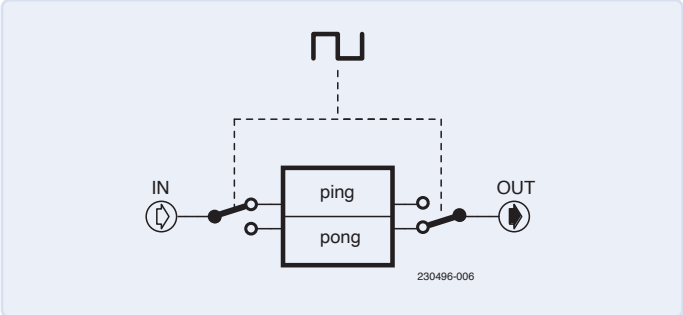

Figure 5. La mise en mémoire tampon double ou ping-pong permet d'éviter les discontinuités dans un flux de données.

le maître pour se connecter à l'esclave. Il y a cependant une subtilité qui n'est pas très bien (voire pas du tout) documentée : l'esclave doit définir son champ d'interface Wi-Fi sur `ESP_IF_WIFI_AP`, sinon il ne fonctionnera pas. Ce champ prend par défaut la valeur `ESP_IF_WIFI_STA`, selon les besoins du maître, et il n'a donc pas besoin de le définir explicitement. Par conséquent, ce champ n'apparaît nulle part dans les exemples et nous, les utilisateurs, ignorons qu'il existe.

Push-to-Talk

Lorsque l'ESP-NOW émet en continu, le microcontrôleur chauffe beaucoup. Dans l'application talkie-walkie, il n'y a aucune raison de diffuser en continu, et j'ai donc ajouté un bouton *push-to-talk* (alias PTT, « appuyer pour parler ») (S1). Appuyez sur ce bouton et maintenez-le enfoncé pendant la conversation. Si l'émetteur est apparié au récepteur, la LED s'allume. Du côté du récepteur, la LED s'allume également, indiquant qu'un appel est en cours. Pour éviter tout retour audio, la sortie audio du côté de l'émetteur est coupée lorsque le bouton PTT est enfoncé. Par conséquent, même si la communication est en principe full-duplex, les deux interlocuteurs ne doivent pas essayer de parler en même temps. C'est une excellente occasion de terminer toutes vos phrases avec « à vous ».

Un seul programme pour tous

Le programme consiste en un fichier Arduino *.ino* (croquis ou *sketch*). En dehors de la bibliothèque ESP32 d'Espressif, aucune autre bibliothèque n'est nécessaire.

Le talkie-walkie a besoin d'un appareil maître et d'un appareil esclave. Pour compiler le programme pour un dispositif maître, mettez en commentaire la ligne 12, qui indique `NODE_TYPE_SLAVE`. Pour le dispositif esclave, cette macro doit être définie. Vous pouvez également reconfigurer d'autres paramètres si vous le souhaitez. Il est également possible de compiler sans support d'entrée (`AUDIO_SOURCE`) et/ou de sortie (`AUDIO_SINK`) audio. Ceci est pratique pour le débogage ou pour une application qui ne nécessite qu'une communication unidirectionnelle. Le code source peut être téléchargé ici [3].

Une plus grande fidélité ?

Il ne devrait pas être trop compliqué de transmettre des données audio de bonne qualité avec le protocole ESP-NOW si, au lieu d'utiliser le simple amplificateur de microphone et les convertisseurs CAN et CNA intégrés de l'ESP32, vous passez à l'I²S. Cela rend le circuit et le programme un peu plus complexes, mais permettrait, en théorie, de diffuser des données audio de 16 bits à une fréquence d'échantillonnage de 48 kHz. Toutefois, la réception éventuelle de paquets dans le désordre doit être gérée correctement. Mais bon, le Bluetooth n'a-t-il pas été conçu pour cela ?

Test de portée

Pour voir si l'ESP-NOW permet une communication à longue distance, j'ai écrit un programme simple pour envoyer un message ping à l'esclave une fois par seconde. L'esclave n'est rien d'autre qu'un ESP32-PICO-KIT avec une LED connectée à un GPIO27, alimenté par une power bank USB. Chaque fois qu'un ping est reçu, la LED clignote brièvement (100 ms).

Avec l'émetteur placé à l'extérieur à 1 m au-dessus du sol, j'ai obtenu une distance de communication en visibilité directe (LOS) d'environ 150 m. À cette distance, la réception est devenue intermittente et l'esclave a dû être tenu en hauteur (environ 2 m au-dessus du sol). Cette situation peut probablement être améliorée en positionnant (et en concevant) soigneusement les deux modules.

VF : Maxime Valens — 230496-04

Questions ou commentaires ?

Vous avez des questions techniques ou des commentaires sur cet article ? Envoyez un courriel à l'auteur à l'adresse clemens.valens@elektor.com ou contactez Elektor à l'adresse redaction@elektor.fr.

À propos de l'auteur

Après une carrière dans l'électronique marine et industrielle, Clemens Valens a commencé à travailler pour Elektor en 2008 en tant que rédacteur en chef d'Elektor France. Il a occupé différents postes depuis lors et a récemment rejoint le département de développement de produits. Ses principaux centres d'intérêt sont le traitement du signal et la génération de sons.

Produits

- **ESP32-PICO-KIT V4**
  www.elektor.fr/esp32-pico-kit-v4
- **Elektor ESP32 Smart Kit Bundle**
  www.elektor.fr/elektor-esp32-smart-kit-bundle

LIENS

[1] En savoir plus sur l'ESP-NOW : https://espressif.com/en/solutions/low-power-solutions/esp-now
[2] Le shield anti-ronflement d'Elektor : https://www.elektormagazine.fr/magazine/elektor-71/42319
[3] Téléchargements pour cet article : https://www.elektormagazine.fr/230496-04