Walkie-talkie met ESP-NOW

niet echt WiFi, niet echt Bluetooth, maar...

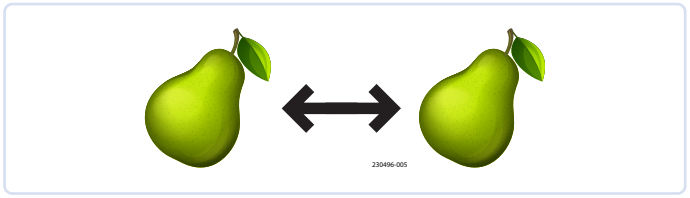
Clemens Valens (Elektor)

Heeft je draadloze project zowel snelle reactietijden als een groot bereik nodig? WiFi en Bluetooth zijn ongeschikt voor dergelijke toepassingen. Misschien is ESP-NOW een goed alternatief? Verbindingen worden vrijwel onmiddellijk tot stand gebracht en een bereik van enkele honderden meters is haalbaar. In dit artikel proberen we het uit in een eenvoudige walkie-talkie of draadloze intercomtoepassing.

De ESP32 van Espressif wordt vaak ingezet vanwege zijn WiFi- en Bluetooth-mogelijkheden, waar hij werkelijk topprestaties levert. WiFi en Bluetooth zijn prima protocollen voor allerlei draadloze toepassingen, maar ze hebben hun beperkingen.

Een ongemak van WiFi is de tijd die nodig is om een verbinding tot stand te brengen. Bovendien maakt WiFi directe communicatie (peer-to-peer, **figuur 1**) tussen apparaten niet mogelijk. Er is altijd een router bij betrokken. Daarom is WiFi niet echt geschikt voor eenvoudige afstandsbedieningen met een geringe latentie om een garagedeur te openen of een licht aan en uit te doen. Dergelijke taken vereisen een onmiddellijke reactie. Om dit te omzeilen, zijn WiFi-toepassingen vaak de hele tijd ingeschakeld en verbonden. Hierdoor verbruiken ze veel energie, zelfs als ze niet actief zijn.

Bluetooth daarentegen maakt een snelle verbinding en peer-to-peer communicatie mogelijk en is uitstekend geschikt voor afstandsbedieningen met een geringe latentie. Bluetooth is echter bedoeld voor de korte afstand met communicerende apparaten die maximaal tien meter van elkaar verwijderd zijn. Bluetooth voor lange afstanden bestaat wel, maar is nog niet algemeen beschikbaar.

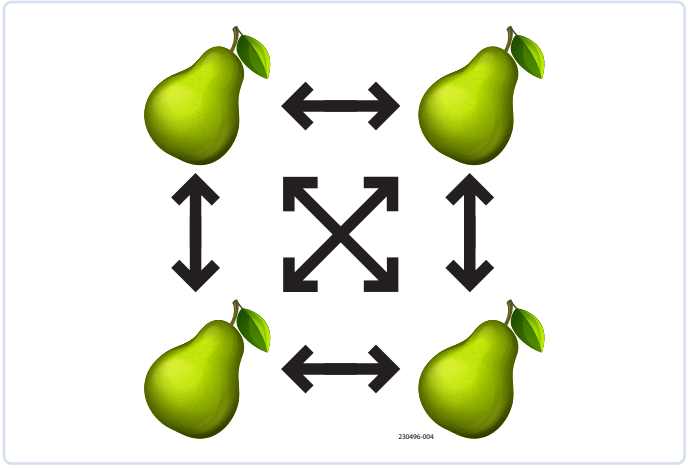


Figuur 1. Peer-to-peer communicatie zoals hier getoond is niet mogelijk met WiFi; er is dan altijd een router nodig tussen de twee nodes.

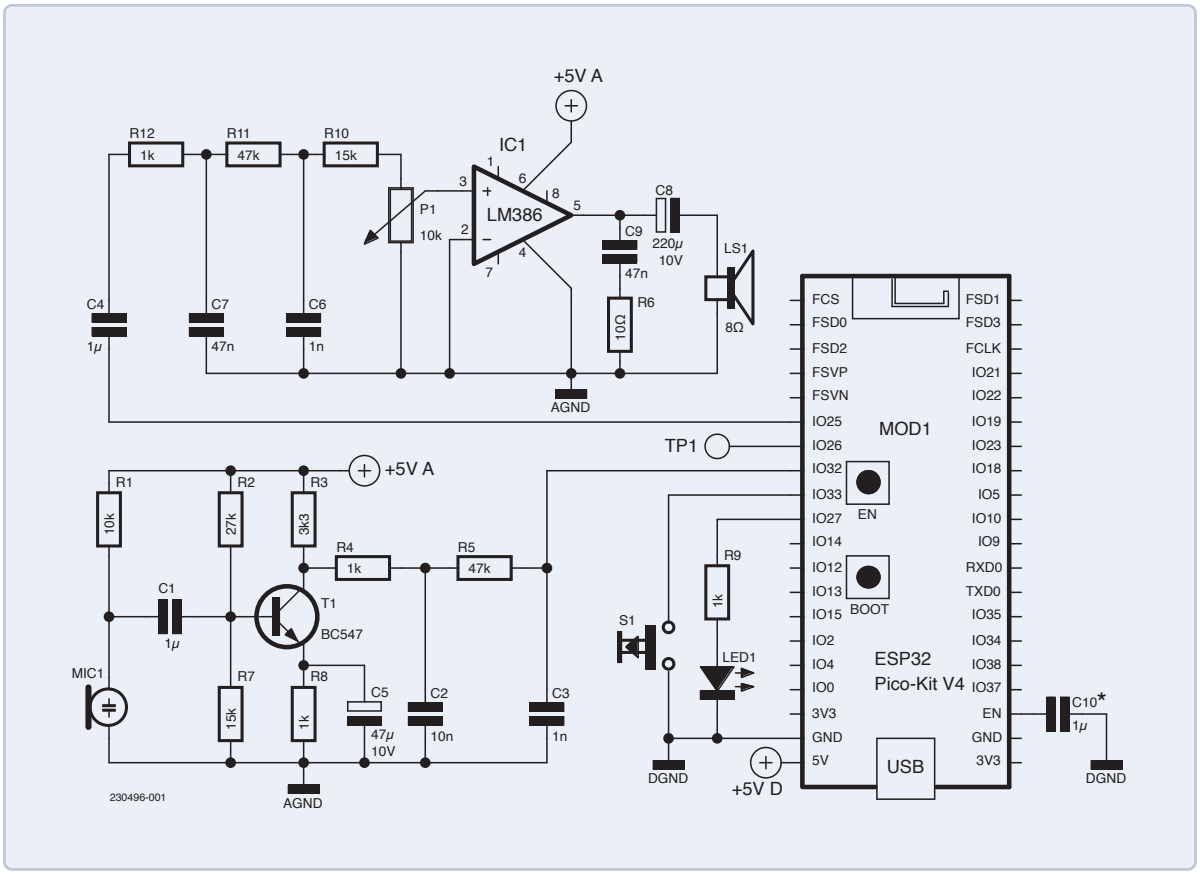
De oplossing: ESP-NOW

Het draadloze protocol ESP-NOW [1] van Espressif is een oplossing voor situaties die zowel snelle reactietijden als een groot bereik vereisen, terwijl het dezelfde frequentieband gebruikt als WiFi en Bluetooth. Het protocol combineert de voordelen van WiFi en Bluetooth. ESP-NOW is gericht op domotica en smart home. Omdat het geschikt is voor one-to-many en many-to-many topologieën (**figuur 2**), heeft het geen router, gateway of, erger nog, een cloud nodig.

ESP-NOW implementeert geen ingewikkelde verbindingsprotocollen of communicatieprotocollen op hoog niveau. Adressering is gebaseerd



Figuur 2. ESP-NOW ondersteunt many-to-many netwerken. In een dergelijk netwerk kan elk knooppunt direct met de andere knooppunten praten zonder dat er een router nodig is.



Figuur 3. Een eenvoudige microfoonvoorversterker aan de ingang en een klassieke LM386 als eindversterker aan de uitgang. Merk op hoe de voedingen voor de analoge en digitale delen gescheiden zijn.

op het Ethernet MAC-adres van het knooppunt en er is pairing nodig om ze met elkaar te laten praten. Ook is het niet gegarandeerd dat gegevenspakketten in de juiste volgorde aankomen. Voor eenvoudige toepassingen met afstandsbediening is dit alles geen probleem. De gegevenssnelheid van ESP-NOW is standaard 1 Mbit/s (instelbaar) en een datapakket kan een payload van maximaal 250 bytes hebben. Samen met de header en checksum bytes enzovoort resulteert dit in een maximale pakketgrootte van 255 bytes.

Praktijkvoorbeeld: walkie-talkie

Mijn doel was om een portofoonachtig apparaat of intercom te maken op basis van ESP-NOW. Een snelle blik op de specificaties van de ESP32 laat zien dat deze alles in zich heeft wat hiervoor nodig is: een analoog/digitaal-omzetter (ADC), een digitaal/analoog-omzetter (DAC), veel rekenkracht, en natuurlijk al het radiospul. In de praktijk ziet het er echter iets minder rooskleurig uit.

De 12-bit brede ADC blijkt nogal traag te zijn, ik heb een maximale samplefrequentie gemeten van rond de 20 kHz. Ergens online werd vermeld dat de analoge bandbreedte slechts 6 kHz bedraagt. De DAC is acht bit breed (maar er zijn er twee), wat de haalbare audiokwaliteit nog meer beperkt.

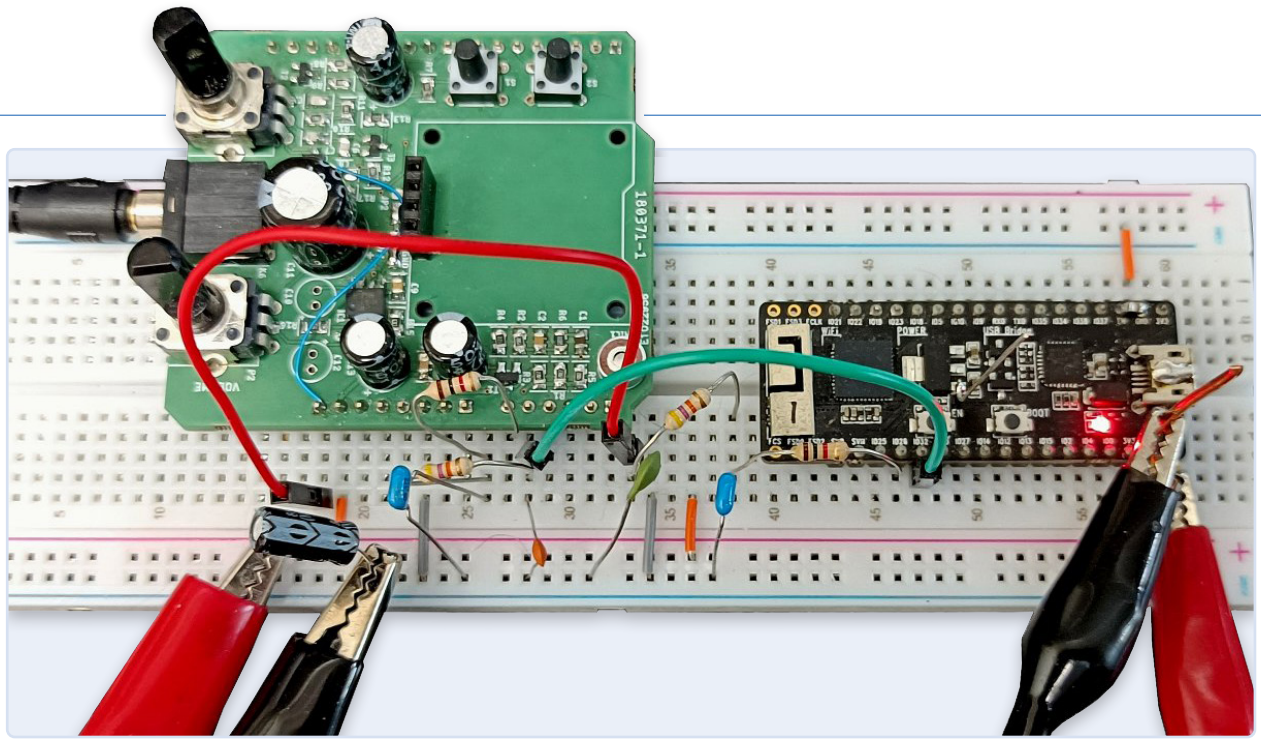
Een portofoon kan echter wegkomen met deze getallen als de audiobandbreedte wordt beperkt tot de standaard telefoniebandbreedte van 3,5 kHz. Een bemonsteringsfrequentie van 8 kHz resulteert in een datasnelheid van (8.000 / 250) × 255 × 8 = 65.280 bit/s (vergeet niet dat de maximale payload 250 bytes bedraagt). Dit blijft ver onder de standaardsnelheid van 1 Mbit/s. Met deze specificaties krijgen we geen natuurgetrouwe audio, maar dat is ook niet ons doel. Verstaanbaarheid is belangrijker.

De schakeling

Om het eenvoudig te houden, gebruikte ik een één-transistor condensatormicrofoon-voorversterker met begrensde bandbreedte als audio-ingang en voegde ik een klassieke versterker op basis van een LM386 als audio-uitgang. Het schema is getekend in **figuur 3**. De ingangsbandbreedte wordt aan de onderkant beperkt door C1 en C5, die iets te klein gedimensioneerd zijn. Laagdoorlaatfilters R4/C2 en R5/C3 zorgen voor de begrenzing aan de bovenkant. Vergelijkbare laagdoorlaatfilters zijn aan de uitgang van de DAC opgenomen. Het signaal aan de 'hete' kant van P1 mag niet groter zijn dan 400 mV_{PP}.

Als ESP32-module heb ik gekozen voor de ESP32-PICO-KIT. Er bestaan veel andere modules, maar die hebbe niet allemaal de DAC-uitgangen op GPIO25 en GPIO26. Ook hebben we een ADC-ingang nodig. Ik heb hiervoor GPIO32 gebruikt, corresponderend met ADC1, kanaal 4. Het testpunt TP1 op GPIO26 (de tweede DAC-uitgang) is bedoeld als monitoruitgang voor het microfoonsignaal. Een drukknop op GPIO33 zorgt voor push-to-talk-functionaliteit (PTT) en de LED op GPIO27 is de verplichte multifunctionele microcontroller-LED.

Merk op hoe de voeding is opgesplitst in een analoog en een digitaal deel. De reden hiervoor is niet om te voorkomen dat snelle digitale schakelruis op de audio-ingang wordt ingekoppeld, maar om een klikkend geluid aan de uitgang te voorkomen. Blijkbaar produceert een taak die op de ESP32 wordt uitgevoerd periodieke stroompieken die hoorbaar kunnen worden als de schakeling niet zorgvuldig is bedraad. De beste manier die ik vond om dit te vermijden is door twee aparte voedingen te gebruiken (**figuur 4**). De ESP32-module moet worden behandeld als een component die een eigen voeding nodig heeft (zoals de LM386), en niet als een module die ook de rest van de schakeling van stroom kan voorzien; in deze toepassing gaat dat niet.



Figuur 4. Een proof-of-concept gebouwd op een breadboard met een ESP32-PICO-KIT en een licht aangepast Elektor Snore Shield [2] voor de ingangs- en uitgangsversterkers. Let op de twee paar krokodillenklemmen die zorgen voor de afzonderlijke analoge en digitale voedingen.

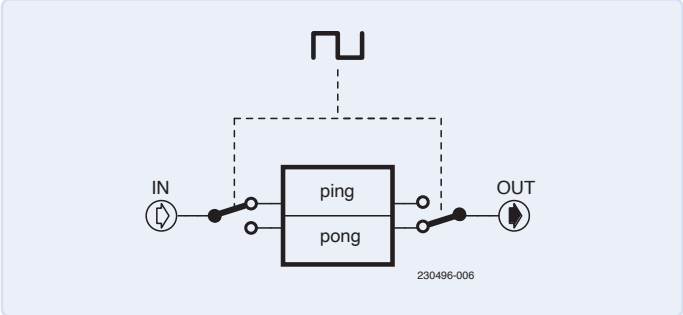
Houd er rekening mee dat de LM386 een voedingsspanningbereik heeft van 4...12 V.
C10 is optioneel en is alleen nodig in enkele zeldzame gevallen van vroege ESP32-modules die niet goed opstarten als ze niet zijn aangesloten op een computer (of iets dergelijks). Toevallig heb ik een paar van deze 'oude' modules en daarom heb ik C10 in mijn ontwerp opgenomen.

De software

Ik heb het programma voor de portofoon gebaseerd op het *ESPNow_Basic_Master* voorbeeld dat wordt geleverd met het Arduino ESP32 boards-package van Espressif. Nadat ik het had aangepast aan mijn behoeften, heb ik er audio-sampling en -afspelen aan toegevoegd. Er zijn een paar dingen die je misschien wilt weten over het programma. Het bemonsteren en afspelen van audio wordt geregeld door een timer-interrupt die op 8 kHz draait. Voor de bemonstering zet de interrupt-service-routine (ISR) van de bemonsteringstimer alleen een vlag om aan te geven dat er een nieuw sample moet worden opgehaald. De `loop()`-functie kijkt naar deze vlag en onderneemt de noodzakelijke acties. De reden hiervoor is dat de ADC niet in een ISR gelezen moet worden als de ADC-API van Espressif wordt gebruikt. De functie `adc1_get_raw()` die hier wordt gebruikt, roept allerlei andere functies aan die dingen kunnen doen waarover je geen controle hebt. Omdat de ESP32-software in een multitasking-omgeving draait, is het belangrijk om de veiligheid van threads te garanderen. Wanneer je Arduino gebruikt voor ESP32-programmering, wordt veel hiervan al voor je afgehandeld, maar als je van plan bent om mijn programma te porten naar de ESP-IDF, moet je misschien wat voorzichtiger zijn.
Het afspelen van audio is eenvoudig omdat de sample rate timer-ISR gewoon een sample naar de DAC schrijft als er een beschikbaar is. Zo niet, dan wordt de DAC-uitgang vastgezet op de helft van de ESP32-voedingsspanning (1,65 V). Het enige waar je hier op moet letten is dat er een zogenaamde pingpong-buffer wordt gebruikt voor het stroomlijnen van digitale audio-ontvangst (**figuur 5**). Zo'n buffer bestaat uit twee afzonderlijke buffers, waarvan de ene wordt gevuld terwijl de andere wordt gelezen. Hierdoor is overlapping mogelijk. In theorie zou dit niet mogen gebeuren omdat zender en ontvanger dezelfde sample rate en timinglogica gebruiken, maar in werkelijkheid gebeurt dit wel vanwege timingtoleranties. Een pingpong- of dubbele buffer helpt om vervelende klikken tijdens het afspelen te voorkomen. Merk op dat het niet in de juiste volgorde ontvangen van datapakketjes niet wordt behandeld.

Pairing

De portofoon-firmware is een master/slave-systeem. De master werkt in WiFi-station-modus (STA), terwijl een slave in access-point-modus (AP) werkt. De master maakt onmiddellijk verbinding met een slave als hij er een detecteert en kan meteen beginnen met het verzenden van gegevens. Wanneer de master echter verbinding maakt met de slave, wordt hierdoor de slave niet ook met de master verbonden. De slave kan geen gegevens naar de master sturen en tweewegbedrijf is niet mogelijk (dat is me tenminsteniet gelukt; als je het beter weet, laat het me dan weten).
Een manier om de slaaf verbinding met de master te laten maken, is door de callback voor gegevensontvangst te gebruiken. Wanneer data wordt ontvangen, wordt het adres van de afzender samen met de data doorgegeven aan deze functie. Daarom kan de slave,



Figuur 5. Dubbele of pingpong-buffering helpt om onderbrekingen in een gegevensstroom te vermijden.

zodra hij iets ontvangt, verbinding maken met de afzender van de data. Hiervoor heb ik dezelfde functies en procedure gebruikt als die de master gebruikt om verbinding te maken met de slave. Er is echter een subtiliteit die niet erg goed (of helemaal niet) gedocumenteerd is: de slave moet zijn WiFi-interfaceveld instellen op `ESP_IF_WIFI_AP`, anders werkt het niet. Dit veld staat standaard op `ESP_IF_WIFI_STA` zoals de master nodig heeft, dus het programma (of de programmeur) hoeft het niet expliciet in te stellen. Als gevolg hiervan verschijnt het veld nergens in de voorbeeldprogramma's, zodat de gebruiker niet weet dat het bestaat.

Push-to-talk

Als ESP-NOW continu streamt, wordt de MCU behoorlijk heet. In de portofoon-toepassing is er geen reden om continu te streamen, en daarom heb ik een push-to-talk-knop (S1, ook bekend als PTT-knop) toegevoegd. Houd deze knop ingedrukt terwijl je praat. Als de zender gekoppeld is met de ontvanger, gaat de LED branden. Aan de kant van de ontvanger gaat de LED ook branden om aan te geven dat er een oproep binnenkomt. Om audiofeedback te voorkomen, wordt de audio-uitgang aan de kant van de zender gedempt als de PTT-knop wordt ingedrukt. Hoewel de communicatie in principe full-duplex is, moeten de twee peers dus niet proberen tegelijkertijd te praten. Dit is een mooie gelegenheid om 'roger' en 'over' in je zinnen te verwerken.

Eén universeel programma

Het programma bestaat uit één Arduino *.ino*-bestand ('sketch'). Naast het Espressif ESP32 boards package zijn er geen andere bibliotheken nodig. De portofoon heeft een master- en een slave-apparaat nodig. Om het programma te compileren voor een master, moet je regel 12 uitcommentariëren, dat is de regel waarin `NODE_TYPE_SLAVE` staat. Voor een slave moet deze macro worden gedefinieerd. Je kunt ook enkele andere instellingen aanpassen als je dat wilt. Het is ook mogelijk om te compileren zonder ondersteuning voor audio-invoer (`AUDIO_SOURCE`) en/of -uitvoer (`AUDIO_SINK`). Dit is praktisch voor debugging of voor een toepassing die alleen eenrichtingscommunicatie nodig heeft. De broncode kan worden gedownload van [3].

Betere geluidskwaliteit?

Het zou niet al te ingewikkeld moeten zijn om audiogegevens van hoge kwaliteit over ESP-NOW te streamen als je, in plaats van de eenvoudige microfoonversterker en de ingebouwde ADC en DAC van de ESP32 te gebruiken, overschakelt op I²S. Dit maakt de schakeling en het programma iets complexer, maar zou het – in theorie althans – mogelijk maken om 16bit-audiogegevens te streamen met een bemonsteringsfrequentie van 48 kHz. Er moet echter wel goed worden omgegaan met de mogelijke ontvangst van pakketten die niet in de juiste volgorde zijn. Maar is Bluetooth daar niet voor ontworpen?

Bereiktest

Om te zien of ESP-NOW communicatie over lange afstanden mogelijk maakt, heb ik een eenvoudig programma geschreven dat één keer per seconde een 'ping' naar de slave stuurt. De slave was niets meer dan een ESP32-PICO-KIT met een LED aangesloten op GPIO27, gevoed door een USB-powerbank. Telkens als er een ping wordt ontvangen, licht de LED kort (100 ms) op.
Met de zender buiten geplaatst op 1 m boven de grond, haalde ik een line-of-sight (LOS) communicatieafstand van ongeveer 150 m. Op deze afstand werd de ontvangst onderbroken en moest de slave hoog gehouden worden (ongeveer 2 m boven de grond). Dit kan waarschijnlijk worden verbeterd door de twee peers zorgvuldig te positioneren (en te ontwerpen).

230496-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via clemens.valens@elektor.com of naar de redactie van Elektor via redactie@elektor.com.

Over de auteur

Na een carrière in de maritieme en industriële elektronica begon Clemens Valens in 2008 bij Elektor als hoofdredacteur van Elektor Frankrijk. Hij heeft sindsdien verschillende functies bekleed en is onlangs overgestapt naar de afdeling productontwikkeling. Zijn belangrijkste interesses zijn signaalverwerking en het genereren van geluid.

Gerelateerde producten

- **ESP32-PICO-KIT V4**
  www.elektor.nl/18423
- **Elektor ESP32 Smart Kit Bundle**
  www.elektor.nl/19033

WEBLINKS

[1] Meer over ESP-NOW: https://espressif.com/en/solutions/low-power-solutions/esp-now
[2] Het Elektor Snore Shield: http://www.elektormagazine.nl/magazine/elektor-72/42288
[3] Downloads voor dit artikel: http://www.elektormagazine.nl/230496-03