Adaptateur ESP32-RS-232

liaison sans fil pour les équipements de test classiques

Georg Hofstetter (Allemagne)

RS-232 devient sans fil ! Un module ESP32 moderne et peu coûteux donne des ailes à une interface série. En plus de sa capacité de communication par liaison série, ce petit adaptateur permet aux équipements de test SCPI de recevoir des commandes et d'envoyer des données sur le réseau domestique via MQTT.

En travaillant sur différents projets électroniques au fil des ans, les équipements de test sur votre établi ne cessent de s'accumuler. Dans mon cas, j'utilise un SourceMeter 2400 de Keithley pour détecter les défauts dans les appareils électroniques, les alimenter ou même mesurer les propriétés de divers composants. C'est également un instrument très utile pour tester le courant de fuite des condensateurs électrolytiques ou mesurer leurs capacités nominales. J'utilise également un récepteur AOR AR5000 pour communiquer avec les radioamateurs du monde entier. Ces deux appareils, comme beaucoup d'autres dans le labo, peuvent être contrôlés par un PC via leurs interfaces série RS-232 intégrés. La disponibilité de câbles adaptateurs USB-RS-232 rend la connexion à un PC personnel assez simple. Le câble adaptateur que j'ai utilisé a bien fonctionné avec l'AR5000, mais la communication avec le SourceMeter de Keithley s'est avérée beaucoup moins stable. Des caractères se perdent parfois lors de l'envoi de commandes SCPI et la connexion a été souvent interrompue.

La connexion RS-232

La norme RS-232, établie dans les années 1960, permet de transmettre des bits de données sérielles d'un caractère en utilisant des changements de niveaux de tension. Les propriétés électriques ont été définies dans la norme V.28, tandis que le fonctionnement général a été réglementé par la norme V.24. La norme V.28 spécifie que, selon la valeur du bit logique, un « 0 » logique est transmis comme un front montant allant jusqu'à 5...12 V et un « 1 » logique comme un front descendant allant jusqu'à -5...-12 V. Le récepteur a une plage de tension légèrement plus large et interprétera un signal dans la plage de 3 à 12 V comme un «0» logique et un signal dans la plage de -3 à -12 V comme un «1» logique. Cela tient compte de la dégradation du signal dans le câble due au bruit, à la chute de tension et à la déviation des fronts du signal causée par l'impédance du câble.

Dans les anciens adaptateurs USB-série peu coûteux et dans certains ordinateurs portables, la variation du niveau du signal du côté de l'émetteur dépasse à peine ±5 V, voire ±3 V, en raison des simplifications apportées à la conception du pilote du signal RS-232 visant à minimiser les coûts de développement et de composants. Cela a conduit à la croyance populaire que les ordinateurs portables et les dispositifs RS-232 ne fonctionnent souvent pas bien ensemble. L'image de la **figure 1**, extraite d'un document de TI, montre les niveaux de tension conformes aux normes.

Que le problème de communication susmentionné avec l'instrument Keithley soit dû à l'utilisation d'un câble d'extension USB, à une boucle de masse ou aux caractéristiques électriques de l'émetteur-récepteur intégré, il n'a pas d'importance ici. Après tout, un câble USB tendu à travers le labo n'est pas seulement encombrant, mais il présente également un risque de trébuchement. Ce dont nous avons besoin, c'est une solution sans fil compacte ! J'ai cherché en vain une solution prête à l'emploi à mon problème, mais je n'ai trouvé que des modules assez volumineux montés

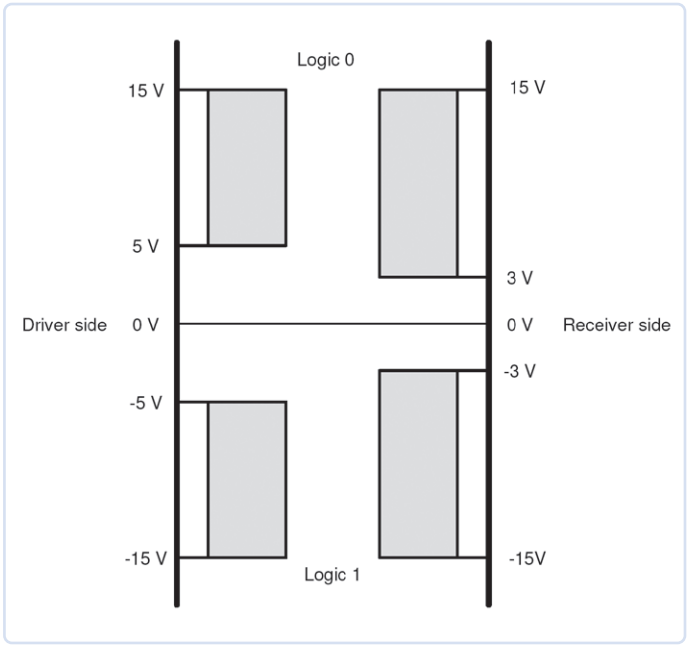


Figure 1. Niveaux de signal RS-232 spécifiés à la sortie de l'émetteur et à l'entrée du récepteur. (Source : Texas Instruments [9])

sur rail DIN, ou des circuits *DIY* plus conçus pour résoudre un problème spécifique et présentant trop peu de fonctionnalités pour moi. Inutile : si je voulais une unité de communication radio RS-232 propre et universelle, je devais prendre le temps de la concevoir moi-même.

Critères de conception

Ce projet est destiné aux labos de *maker* et ne sera pas un produit joliment emballé que l'on peut acheter dans un magasin d'électronique. Cependant, il est préférable d'examiner attentivement la conception de l'appareil afin qu'il puisse effectuer toutes les tâches pour lesquelles il est prévu.

Un ESP32 fait tout le travail

La passerelle RS2-32 est principalement destinée à transmettre des données vers et depuis le réseau. Cela se fait généralement via le port TCP 23 (Telnet). Cependant, des modifications logicielles spécifiques à l'application devraient également être possibles, que l'utilisateur peut adopter en fonction des exigences de l'application. Pour moi, un service SCPI-to-MQTT était important, pour que l'équipement de test SourceMeter puisse transmettre ses mesures pendant la nuit à Grafana (voir ci-dessous) à des fins d'évaluation.

L'ESP32 offre une puissance de calcul suffisante (en plus de l'envoi de simples paquets TCP) pour que l'appareil connecté puisse être facilement intégré à Home Assistant ou à un service similaire via MQTT ou pour le partager via un frontal dédié via le serveur web de l'ESP32.

Une alternative serait l'ESP8266, qui prend un peu moins de place et est un peu moins cher. Cependant, l'encombrement réduit se fait au prix d'un processeur considérablement plus sollicité et d'une mémoire RAM réduite. En outre, l'ESP8266 ne dispose pas d'un module Bluetooth (avantageux dans cette application) en plus de la fonction Wifi souhaitée.

L'inconvénient de l'ESP32 est, bien entendu, sa consommation de courant considérablement plus élevée, qui atteint un pic d'environ 500 mA sur le rail de 3,3 V. Le CPU ne se chargera évidemment pas de calculer en permanence, mais selon le degré d'optimisation finale du logiciel, le courant de crête atteindra probablement cette valeur, comme l'indique le tableau 4.2 de la fiche technique [1]. Une comparaison avec la fiche technique de l'ESP8266 [2] montre que la consommation d'énergie de l'ESP32 est supérieure d'environ 35 % en émission et d'environ 80 % en réception.

Un design soigné et compact

La plupart des gens conviendront probablement qu'il y a déjà suffisamment de câbles qui traînent sur et sous les établis. Il est possible de souder ce circuit imprimé directement à un connecteur D-sub DE-9. Il n'est pas beaucoup plus grand qu'une prise secteur IEC C13 standard (y compris le rayon de courbure). Cela signifie qu'une fois connectée, la carte n'augmente pas l'encombrement de l'appareil RS-232. Si un boîtier est nécessaire, il est possible de l'imprimer facilement en 3D ; dans l'idéal, le circuit imprimé peut même s'insérer dans un boîtier standard disponible sur le marché.

Alimentation par micro-USB

La carte est alimentée via un connecteur micro-USB (désormais le standard pour de nombreux petits appareils). La tension +3,3 V requise par l'ESP32 est fournie par le régulateur de tension linéaire Advanced Monolithic Systems [3] AMS1117 à partir de l'alimentation +5 V du port USB. Bien que l'AMS1117 ait une capacité de 1 A, il chauffe beaucoup sur ce circuit imprimé relativement petit. Une alternative plus efficace est le convertisseur abaisseur à découpage de Texas Instruments [4] TPS62291, livré dans un boîtier WSON-6 de 2 × 2 mm. Il fournit également un courant maximum de 1 A et permet de réduire l'encombrement. À moins de disposer d'un microscope et tous les outils et l'expérience nécessaires pour travailler à ce niveau, cette puce est un peu plus difficile à assembler en raison de la petite taille du boîtier. Ceux qui doutent de leurs compétences en soudage peuvent trouver des conseils utiles dans la vidéo [5].

Il convient également de noter que la disponibilité de ce composant particulier a été irrégulière au cours des deux dernières années. En conséquence son prix a grimpé à plus de 20 € chez certains distributeurs, au lieu des 1,70 € habituels. En comparant les prix sur internet, on peut trouver des exemplaires à des prix qui rappellent le bon vieux temps. Pour simplifier les choses, j'ai utilisé le AMS1117 dans cette version, ce qui facilite grandement la soudure à la main.

Alimentation via le connecteur sub-D à 9 broches

Si vous n'êtes pas opposé à l'idée de sortir des sentiers battus, vous pouvez également alimenter la carte via le connecteur D-Sub à 9 broches. En modifiant légèrement le ou les terminaux, il est possible de fournir les 5 V requis via le connecteur RS-232 (et d'éviter ainsi l'utilisation d'une alimentation supplémentaire). Cependant, la norme ne prend pas en charge une telle option d'alimentation via le connecteur, nous devons donc noter que cette modification n'est pas non plus conforme à la norme.

Selon la norme V.28, toutes les lignes de signal peuvent supporter des tensions de +15 V à -15 V, pour l'alimentation, il est logique de remplacer l'une des fonctions des broches qui est rarement utilisée. La broche 9 est ce qu'on appelle l'indicateur de sonnerie (RI) et n'est utilisée par un modem que pour signaler un appel entrant. Les modems RS-232 d'aujourd'hui sont pratiquement redondants, ce qui fait de la broche RI un candidat idéal pour l'alimentation.

Toutefois, nous devons d'abord vérifier ce qui se passe si nous appliquons subitement la

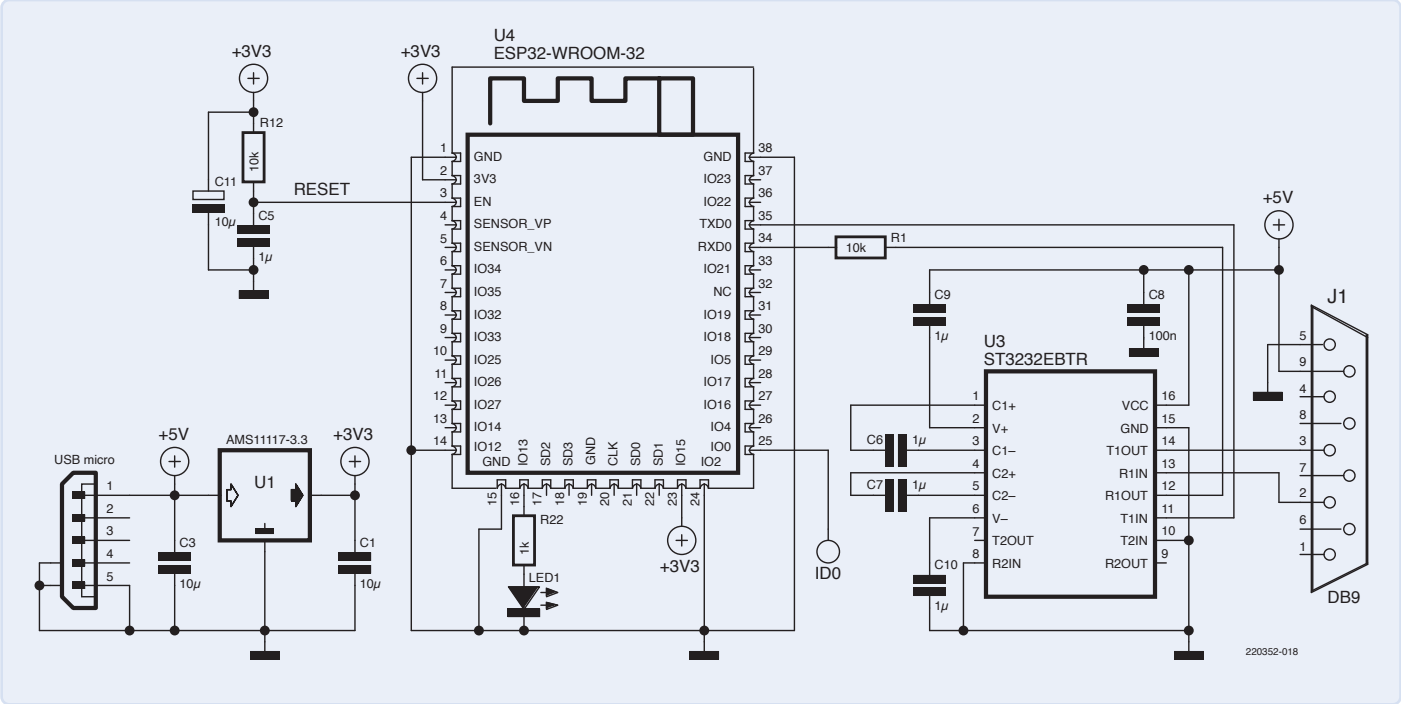


Figure 2. Le schéma simple du dongle ESP32/RS-232.

tension maximale, par exemple en connectant un terminal qui alimente notre ligne d'alimentation conformément à la norme V.28.
La tension +15 V ne poserait pas de problème à l'AMS1117 lui-même ; cette tension est dans ses limites spécifiées. Cependant, si une tension -15 V y est appliquée, les diodes de protection internes du régulateur AMS conduiront les 10 à 20 mA du signal RS-232 à la masse. Empiriquement, ces diodes supportent cette tension et, selon leurs spécifications, les pilotes RS-232 sont protégés contre les courts-circuits, pour qu'aucun dommage n'est à déplorer ici non plus.
La valeur de la tension d'alimentation VDD de notre émetteur-récepteur RS-232 est spécifiée jusqu'à 5,5 V et est également connectée au +5 V. Dans ce cas, la tension chute à environ 3,5 V en raison de la limitation du courant par les pilotes RS-232. Certes, nous sommes un peu courts ici, mais nous n'avons pris en compte tous ces facteurs que pour le cas rare où RI sur la broche 9 est effectivement piloté par l'appareil final. La plupart du temps, cette broche n'est pas connectée.

True RS-232

Comme nous l'avons déjà mentionné, notre objectif est de respecter le plus possible la norme V.28, c'est pourquoi un émetteur-récepteur de signaux de données est absolument nécessaire. Le ST232EBTR de ST [6] fera l'affaire et est disponible dans un boîtier TSSOP-16. Ce CI est abordable (environ 1 €) et comporte un doubleur de tension interne et des circuits inverseurs, il génère des tensions d'alimentation de ± 10 V, qui sont utilisées pour les signaux RS-232 (qui sont typiquement d'environ 9 V dans des conditions normales de fonctionnemen).
Le schéma du dongle ESP32-RS-232 avec le module ESP32-WROOM (ou le WROVER avec PSRAM) et les deux CI (ST232EBTR et AMS1117), ainsi que tous les composants périphériques, sont représentés dans la **figure 2**.

Placement des composants

Pour un circuit imprimé aussi petit et comportant aussi peu de composants, il est peu judicieux de recourir à des professionnels pour le montage des composants ou d'engager des dépenses supplémentaires pour un pochoir de pâte à souder et un assemblage avec une plaque de refusion. Nous pouvons souder à l'ancienne (à la main) le circuit imprimé illustré à la **figure 3**, puisque le nombre de composants est faible et les pastilles sont accessibles.
Le premier composant à assembler est le ST232 (U3). Il est relativement plat, et quelques retouches peuvent être nécessaires pour obtenir des connexions nettes. Ensuite, vous pouvez souder les condensateurs, les résistances et la LED, puis l'AMS1117 (U1) et le connecteur USB. Il ne reste plus qu'à souder l'ESP32 et, enfin, le connecteur D-Sub à 9 broches.

Micrologiciel

La fonction principale du micrologiciel est d'acheminer les caractères reçus sur le réseau vers le port RS232 et vice versa, mais grâce à la polyvalence de l'ESP32, il est possible d'aller plus loin. Le micrologiciel, disponible dans le dépôt [7], offre actuellement deux modes de fonctionnement - Telnet et MQTT/SCPI - que nous expliquerons ci-dessous.

Telnet / Socket TCP brut

Comme nous l'avons déjà évoqué dans les critères de conception, la fonction principale est de transférer des données du réseau vers le port RS-232. Pour de tels adaptateurs, il est courant de transmettre des caractères bruts via le port TCP 23. Ce port était auparavant utilisé pour les connexions Telnet à un serveur mais est rarement utilisé aujourd'hui (sauf dans les systèmes embarqués). Pour notre objectif : l'envoi de commandes ou de mesures vers et depuis des périphériques série, il est suffisant dans un réseau domestique.

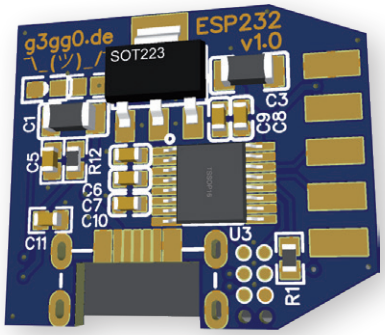


Figure 3. Ce petit circuit imprimé avec des pastilles facilement accessibles est idéal pour souder à la main des CMS.

Figure 4. Exemple de tableau de bord Grafana, montrant des formes d'onde générées à partir de valeurs mesurées au fil du temps (sans rapport avec ce projet).

Telnet est un protocole de communication réseau développé pour permettre d'accéder à distance à un ordinateur sur un même réseau. Il fournit un canal de communication textuel permettant à un utilisateur d'accéder à un ordinateur distant et d'exécuter des commandes. La connexion est bidirectionnelle, de sorte que les entrées et les réponses sont toutes deux transmises sur le réseau. Développé à l'origine à l'époque plus primitive des premiers standards Internet, Telnet est souvent utilisé dans des applications où une simple communication textuelle est suffisante. Il est cependant considéré comme peu sécurisé car il n'est pas crypté, ce qui permet potentiellement l'interception de toutes les données transférées, y compris les mots de passe et d'autres informations sensibles.
Pour les premiers tests de communication, vous pouvez utiliser l'utilitaire *telnet.exe*, fourni par défaut dans les versions antérieures de Windows. S'il n'est pas disponible, vous pouvez utiliser l'alternative gratuite et plus universelle PuTTY [8].

MQTT/SCPI

Pour les applications qui nécessitent des relevés de mesures réguliers sur des périodes de plusieurs heures, vous pouvez développer des scripts Python appropriés ou même lancer des outils prêts à l'emploi sur votre PC et les lire via le port Telnet. Une alternative pratique pour ceux qui ont déjà mis en place une infrastructure appropriée dans leur réseau domestique consiste à se passer complètement du PC gourmand en énergie et à utiliser une solution de mesure plus simple.
De nombreux ménages utilisent déjà des services tels que MQTT, InfluxDB et Grafana pour collecter les données de capteurs avec un Raspberry Pi ou via des conteneurs Docker à l'aide de leur NAS. Sans entrer dans les détails et en termes simples :

- **MQTT** est un protocole de messagerie léger conçu pour une communication efficace entre les appareils IdO et les réseaux à faible bande passante et à latence élevée. Il utilise un courtier en informations avec un modèle de publication/abonnement pour permettre un échange de données transparent avec une surcharge minimale.
- **InfluxDB** est une base de données de séries temporelles conçue pour gérer efficacement des charges d'écriture/demande élevées pour des données horodatées. Les balises sont des paires clé/valeur pour l'indexation, tandis que les champs contiennent les données réelles.
- **Grafana** extrait les valeurs de données d'une base de données et affiche les informations de manière conviviale sous la forme de graphiques définis par l'utilisateur.

Pour ceux qui disposent d'une telle installation chez eux, ce micrologiciel offre la possibilité de publier les relevés générés par l'appareil

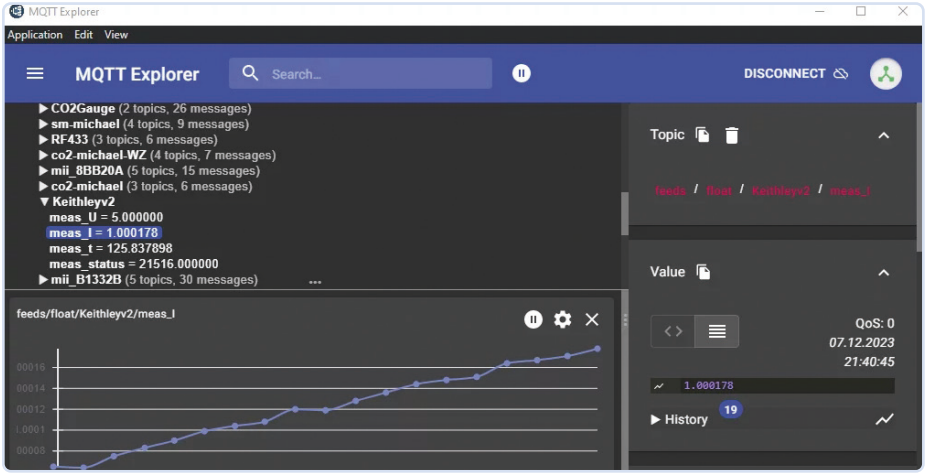


Figure 5. MQTT Explorer montre les mesures publiées par l'adaptateur ESP32/RS-232 envoyées via MQTT.

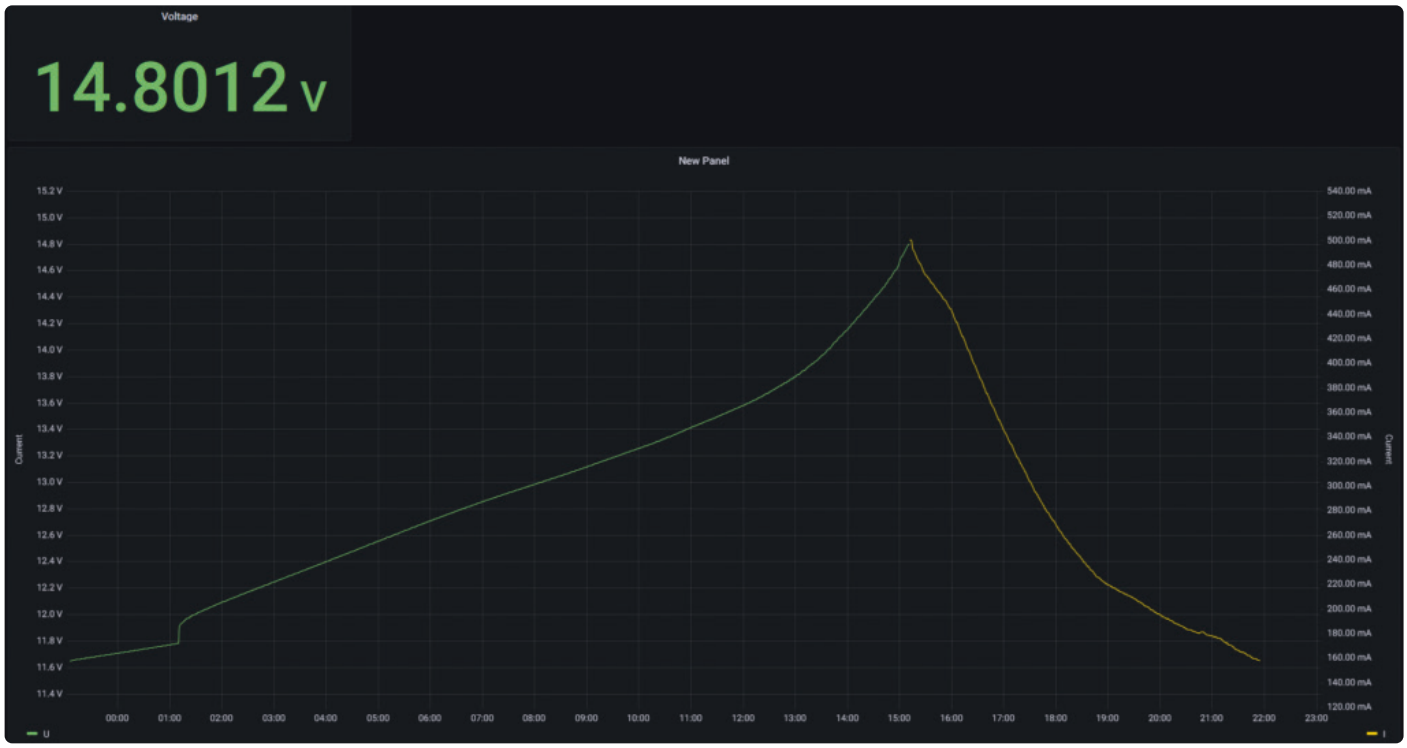


Figure 6. Tension (verte) et courant (jaune) de la batterie pendant la charge d'une batterie au plomb de 12 V, mesurés via un adaptateur ESP32/RS232.

compatible SCPI directement vers un courtier MQTT pour une diffusion ultérieure.
L'ensemble des commandes utilisées par les différents appareils SCPI varie considérablement. Le seul appareil actuellement pris en charge par le code est le SourceMeter 2400 de Keithley Toutefois, il s'agit d'un bon point de départ permettant à chacun d'ajouter ses propres versions de micrologiciels. Les développeurs embarqués sont invités à étendre les fonctionnalités du logiciel et à les partager avec la communauté.
Pour voir les représentations des mesures dans différentes applications, voir la **figure 4** (Grafana), la **figure 5** (MQTT Explorer, un client MQTT complet pour différentes plateformes [10]) et la **figure 6** (Grafana également).
L'ESP32 doit donc effectuer certaines tâches et pour cela, il a besoin d'informations que nous devons lui fournir via une interface web.

Configuration du Wifi

Si aucun point d'accès configuré n'est trouvé ou si aucun n'est configuré, l'ESP32 devient actif et se propose comme point d'accès nommé *esp232-config*. Il est préférable de se connecter à ce point d'accès avec un smartphone. Après s'être connecté, vous pouvez configurer les paramètres principaux en allant sur *http://192.168.4.1/* dans un navigateur. La page web hébergée sur l'ESP32 est présentée dans la **figure 7**. Dans le **tableau 1**, vous pouvez vérifier ce que vous doit être saisi dans chaque ligne.

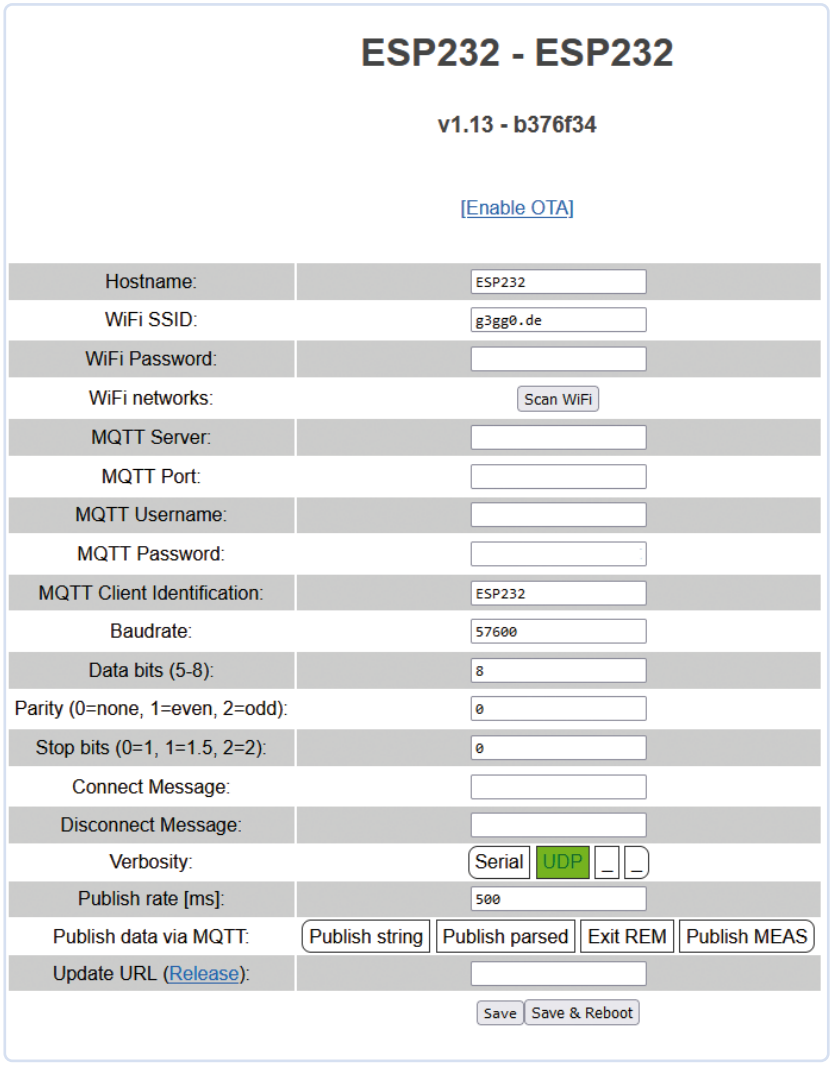


Figure 7. Configuration Wifi de l'ESP32.

Tableau 1. Données de la configuration Wifi de l'ESP32.

Nom d'hôte	Le nom que l'appareil utilisé pour s'identifier via MDNS sur le réseau.
WiFi SSID/mot de passe Wifi	Le réseau Wifi auquel l'appareil doit se connecter au démarrage.
Réseaux Wifi	Affiche une liste des réseaux Wifi trouvés ; en cliquant sur un réseau, son nom s'affiche.
MQTT [...]	Paramètres de configuration pour le client MQTT, en conjonction avec le SCPI ci-dessous.
Débit en bauds / Bits de données / Bits d'arrêt	Paramètres correspondants pour la communication RS232.
Dis-/Connect Message	[cas particuliers] Message à envoyer via RS232 pour une connexion TCP.
Verbosity	[Experts] Communication de débogage via UDP.
Publier des données via MQTT	[Experts] Pour le Source Meter 2400 de Keithley : analyse les données d'affichage ou lance une mesure et publie les résultats via MQTT.
URL de mise à jour	Pour les mises à jour du micrologiciel, il suffit d'entrer les URL HTTP et le fichier .bin attendu sera chargé et mis à jour.

Tableau 2. Affectation des broches de l'interface de programmation.

Broche	Rôle	Niveau
IO0	Définit le mode de démarrage, n'a d'effet qu'après une réinitialisation.	Flash : GND (normal : flottante, 3,3 V)
RESET	Broche reset de l'ESP32 (CHIP_PU).	active : GND, inactive : flottante (3,3 V)
TXD0	Signal d'émission de l'ESP32	0V / 3,3 V
RXD0 (U3_34)	Signal de réception de l'ESP32	0V / 3,3 V
+5V	Tension d'alimentation	5 V
GND	Masse	GND

OTA : mises à jour à distance

Dans l'EDI Arduino ou dans PlatformIO, que je préfère, il est possible de mettre à jour l'ESP32 « à distance «. Le protocole sous-jacent est implémenté par le composant ArduinoOTA. Bien que cette fonctionnalité facilite les mises à jour fréquentes du microcontrôleur, en pratique, des inconvénients significatifs sont apparus. Il semble que l'allocation et la libération constantes de tampons pour chaque paquet UDP reçu fragmente la mémoire de l'ESP32, ce qui peut conduire à des réinitialisations imprévisibles. De plus, cela se produit indépendamment du fait que des paquets OTA soient en cours d'acheminement. Comme vous pouvez l'imaginer, un message de crash exécuté via l'UART de l'ESP32 est rarement bien reçu par l'appareil connecté. Heureusement, il existe des améliorations [11] qui réduisent de manière significative le taux de crash. À cause de ce niveau d'incertitude, la fonction OTA n'est activée que brièvement et à la demande de l'utilisateur. C'est pourquoi un champ (*Enable OTA*) est prévu à cet effet dans l'interface web.

Flashage

Si vous ne disposez pas déjà d'un adaptateur Pogo pour les modules ESP32, vous devez programmer le microcontrôleur avec micrologiciel correct après l'assemblage en utilisant les broches TX et RX de l'ESP32. Pour ce

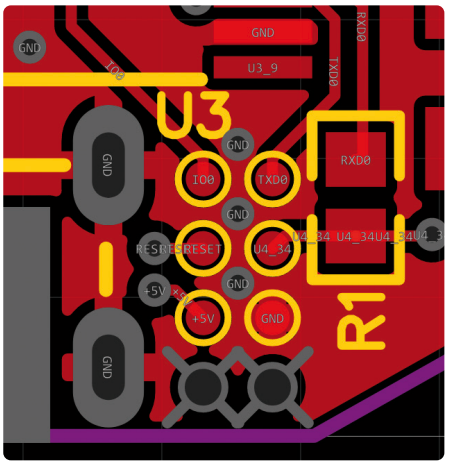


Figure 8. Connexions pour le programmateur sur le circuit imprimé.

faire, téléchargez le code source à partir de [7], importez-le dans PlatformIO, compilez-le et flashez-le sur la carte.

Un port de programmation est fourni en bas à droite de la carte. Pour une utilisation sporadique, vous pouvez simplement mettre à la masse le signal IO0 et souder les TXD0 et RXD0 (broche U4_34 du microcontrôleur, voir **figure 8**) à un adaptateur USB/UART.

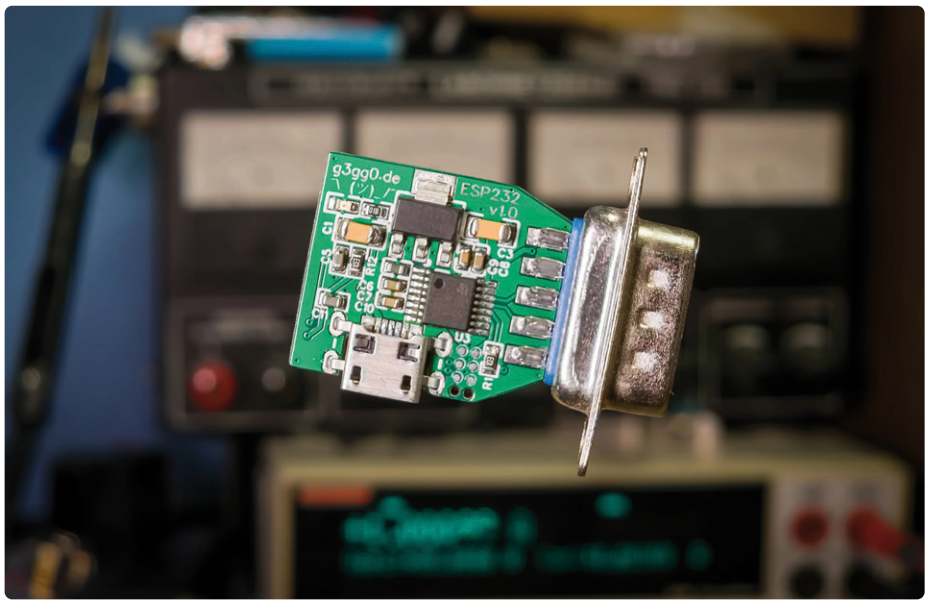
Comme le montre le **tableau 2**, l'ESP32 possède quelques broches de «strapping « que vous devez configurer lorsque vous chargez le code ROM en utilisant cette méthode. Dans cette configuration, nous n'avons à nous préoccuper que de l'une d'entre elles (IO0). Une fois le micrologiciel chargé, une LED devrait commencer à clignoter après environ 30 secondes.

Tout n'est pas beau

L'ESP32 a prouvé ses capacités dans mes projets. Cependant, il y a eu au départ divers problèmes avec la bibliothèque Arduino concernant l'utilisation de la mémoire tampon UART, liés au code accédant directement aux bibliothèques du cadre IDF. Des bogues dans la bibliothèque *ArduinoOTA* m'ont également amené à passer plusieurs nuits à essayer d'identifier les problèmes. En conséquence, il existe certaines solutions de contournement dans le code source. Mais comme ce micrologiciel est open source, il continuera à évoluer et à bénéficier des améliorations apportées par la communauté.

La prochaine version

Ce qui m'a gêné dans la version v1.0 du projet jusqu'à présent, c'est la lourdeur du processus de flashage un peu (ou le flashage pour corriger des erreurs dans le code). Grâce à un adaptateur de flash avec des broches Pogo que j'utilise pour tous mes projets, l'effort est considérablement réduit, mais tout le monde n'a pas un adaptateur approprié à portée de



À propos de l'auteur

L'intérêt de Georg Hofstetter pour l'électronique est né à l'époque du Commodore 64 et des cartes perforées. Après une formation d'ingénieur en électronique, il a obtenu un diplôme en informatique. Depuis lors, il développe ou fait de la rétro-ingénierie de logiciels et de systèmes embarqués et publie des projets sélectionnés sur son site web www.g3gg0.de. Parmi les projets les plus connus, on peut citer des modifications de micrologiciels pour les DSLR Canon appelées Magic Lantern (*www.magiclantern.fm*) et pour le Toniebox (*https://gt-blog.de/toniebox-hacking-how-to-get-started*).

main ou n'aime pas travailler avec des câbles de connexion.
Pour cette raison, la version 2.0 de l'adaptateur ESP32-RS-232 est en cours de développement et utilise un ESP32-S3-PICO. Quelques prototypes initiaux ont déjà été produits et semblent prometteurs. Le PICO est un module complet avec une mémoire flash SPI et une mémoire RAM, le tout intégré dans un boîtier LGA-56. Bien Bien qu'à première vue, le PICO ressemble à un QFN, qui est généralement facile à souder à la main, la masse (GND) doit être connectée à une pastille visible, ce qui nécessite de souder la pastille à travers un trou lorsqu'il est assemblé manuellement. Ce boîtier n'a pas non plus de broche de connexion et il est difficile de souder les broches à partir du bord. En fin de compte, une (mini) plaque de refusion semble être la meilleure solution dans ce cas [12].

Le grand avantage du S3 est qu'il dispose de toute la périphérie USB à bord pour le flashage ou le débogage. Si nécessaire, il est possible d'utiliser l'ESP232 v2.0 comme adaptateur USB vers RS232.
La question de savoir si cela nous permet de sortir de notre point de départ est une question rhétorique. Si l'on considère que l'ESP8266 était idéal pour construire un adaptateur WiFi/série dès le départ, on peut se demander pourquoi un produit similaire n'est pas encore sur le marché aujourd'hui.

220352-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (elektor@g3gg0.de), ou contactez Elektor (redaction@elektor.fr).

Produits

- **ESP32-WROOM-32**
  www.elektor.fr/18421
- **ESP32-S2-WROVER**
  www.elektor.fr/19693

LIENS

[1] Fiche technique de l'ESP32 : https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf
[2] Fiche technique de l'ESP8266 : https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[3] AMS1117 : http://www.advanced-monolithic.com/pdf/ds1117.pdf
[4] TPS62291 : https://www.ti.com/product/de-de/TPS62291
[5] SMD soldering by hand, vidéo d'Elektor : https://www.elektormagazine.de/news/uberzeugendes-video-loten-von-smd-bauteilen
[6] ST232 : https://www.st.com/resource/en/datasheet/st202eb.pdf
[7] Dépôt du projet : https://github.com/g3gg0/ESP232
[8] PuTTY : https://www.putty.org/
[9] MQTT Explorer : https://mqtt-explorer.com/
[10] WiFiUDP Crash issue : https://github.com/espressif/arduino-esp32/issues/4104
[11] Texas Instruments Application Report: RS-232 Frequently Asked Questions : https://www.ti.com/lit/an/slla544/slla544.pdf
[12] M. Divito, « Mini plaque de refusion », Elektor 11-12/2023 : https://www.elektormagazine.fr/230456-04