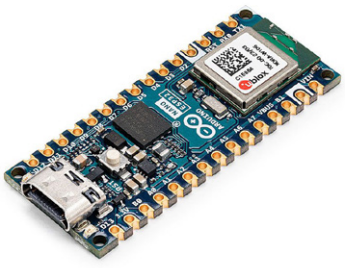
en détail :
entretien avec Arduino sur le Nano ESP32

Alessandro Ranellucci et Martino Facchin abordent la collaboration avec Espressif

Questions de Saad Imtiaz et de Clemens Valens (Elektor)

À l'été 2023, Arduino a lancé la carte Nano ESP32. Basée sur l'ESP32-S3 d'Espressif, la nouvelle carte offre une connexion Wifi 2,4 GHz, 802.11 b/g/n, et une connectivité Bluetooth 5 (basse consommation) à longue portée dans un format Nano. Le Nano ESP32 n'est pas la première carte Arduino équipée d'un processeur d'Espressif, mais cette fois-ci, il s'agit de l'élément principal et non d'un simple module de communication sans fil supportant un autre microcontrôleur. Elektor a interviewé Alessandro Ranellucci (Directeur, Arduino Makers Division) et Martino Facchin (Responsable Hardware & Firmware, Arduino) à propos de cette collaboration entre Espressif et Arduino.



Elektor : pouvez-vous expliquer pourquoi vous avez choisi l'ESP32 pour la nouvelle série Arduino Nano plutôt qu'un autre microcontrôleur ?

Alessandro Ranellucci : la série Arduino Nano représente un groupe de cartes conçues pour être compatibles les unes avec les autres en termes de forme, de disposition de broches et de caractéristiques techniques essentielles. Cette uniformité est appréciée par notre communauté de *makers* car elle permet une interchangeabilité sans faille au niveau de la série Nano. Cependant, nous n'avions pas encore introduit de carte basée sur la très répandue architecture ESP32. Nous avons estimé qu'il était nécessaire de fournir une option efficace au sein de la série Nano qui utilise l'ESP32, ce qui nous a amenés à prendre cette décision.

Martino Facchin : oui, au cours des quatre dernières années à peu près, nous avons développé la série Nano en y intégrant des microcontrôleurs de divers fournisseurs de puces avec lesquels nous n'avions pas travaillé auparavant. Nous avons commencé par intégrer des microcontrôleurs de Nordic, puis de Raspberry Pi. Il s'agit essentiellement de notre terrain d'expérimentation pour tester des architectures nouvelles et répandues, tout en ajoutant une utilité qui n'est pas disponible ailleurs. Par exemple, sur la puce simple ESP32, il n'y avait pas d'USB. Dans la C3, il y a un port USB, mais il est associé à un identifiant de fournisseur et de produit fixe, de sorte qu'il n'est pas possible de faire la distinction entre les différentes cartes. La S3 est la première carte à disposer de cette fonctionnalité, c'est pourquoi nous l'avons utilisée, plutôt qu'une carte plus ancienne.

Elektor : quels sont les avantages techniques uniques de l'ESP32 par rapport aux autres options ?

Martino Facchin : oui, en effet, nous avons eu l'occasion de collaborer directement avec u-blox pour acquérir une puce spécialisée qui n'est pas disponible à l'achat, qui comprend de la PSRAM intégrée dans la puce elle-même. Nous proposons une puce avec PSRAM et la plus grande quantité possible de mémoire flash externe. Ces détails sont des approfondissements techniques, pour dire ainsi, conçus pour les passionnés et les geeks. Les caractéristiques de la carte poussent l'ESP32 dans ses derniers retranchements les plus élevés disponibles. Ensuite bien sûr, elle inclut le WiFi, le Bluetooth à basse consommation (BLE), le traitement double-coeur, et toutes les autres fonctionnalités qui viennent avec l'ESP32. Nous

Martino Facchin

Alessandro Ranellucci

avons choisi de ne pas inclure de fonctionnalités supplémentaires sur l'Arduino Nano ESP32 comme nous l'avons fait avec d'autres cartes. Celle-ci est conçue comme un bloc de construction. Contrairement à l'Arduino BLE, ce n'est pas un dispositif que vous pouvez utiliser dès sa sortie de la boîte pour des applications conséquentes. Vous devez attacher des composants supplémentaires à la carte, tels que des capteurs ou des modules. Elle est équipée d'une LED RVB, mais en plus de cela, l'utilisateur a la liberté de la compléter avec d'autres composants, en l'utilisant comme une base bien adaptée pour ses projets.

Elektor : avez-vous rencontré des difficultés pour intégrer la puce ESP32 dans le format Arduino Nano, et si oui, comment avez-vous résolu ces problèmes ?

Alessandro Ranellucci : en effet, le premier défi consistait à former un partenariat avec Espressif. Ils ont créé un excellent noyau Arduino pour les cartes ESP32 pour de nombreuses années, un noyau qui est profondément intégré dans le système Arduino, mais qui a également respecté leurs préférences techniques distinctes. Le plus difficile était de trouver comment combiner nos efforts pour créer une expérience utilisateur plus cohérente et améliorée. C'est ainsi que nous avons mis en place cette collaboration. Je pense que Martino peut également mentionner d'autres défis, comme, par exemple, la numérotation des broches.

Martino Facchin : vraiment pas tant que cela, car du point de vue matériel, il n'y a pas eu de problèmes. Bien sûr, c'est facile par rapport aux autres cartes que nous avons fabriquées ces deux dernières années. D'un point de vue logiciel, c'est très différent car la numérotation des broches était très liée au monde des ESP. Les broches étaient numérotées exactement de la même manière que sur le support du processeur. Sur le dessous de la carte, vous pouvez voir des étiquettes d'autres fabricants avec des numéros comme 31, puis à côté un 4, puis un 55, et ainsi de suite. Cela ne convient pas au Nano, ce n'est pas la solution que nous recherchions. Nous avons donc développé un moyen de convertir les numéros de broches logiques en numéros de broches internes. Nous avons fait accepter cette solution au centre de la communauté du noyau ESP32. À l'heure actuelle, toutes les cartes équipées d'un ESP32, quel que soit le fabricant, peuvent utiliser la même logique si elles souhaitent adopter notre philosophie, et il s'agit d'une contribution directe à un noyau dont nous n'assurons pas la maintenance. Cela a été difficile parce que la date de sortie de cette fonctionnalité devait correspondre parfaitement à la date de sortie de la carte. Cependant, nous avons finalement réussi.

C'était un défi, car c'était la première fois que nous sortions quelque chose que nous ne contrôlions pas entièrement, et c'était difficile, mais nous l'avons fait.

Alessandro Ranellucci : nous avons tiré des enseignements importants de l'ensemble du processus de validation car, comme l'a indiqué Martino, tout fabricant peut désormais opter pour l'utilisation de numéros de broches logiques. Il est plus convivial de numéroter les broches de manière séquentielle que d'utiliser la numérotation des broches du contrôleur, comme PA1 et PD1.

Elektor : le développement du logiciel ESP32, l'écosystème et le soutien de la communauté ont-ils influencé votre décision de l'utiliser dans le nouvel Arduino Nano ?

Martino Facchin : non. Nous aurions choisi de l'utiliser même s'il n'y avait pas eu de soutien de la part de la communauté. Il est certain qu'avec le travail considérable de la communauté et le développement de nos projets, nous bénéficions grandement de leurs contributions, en particulier pour le gestionnaire de bibliothèque. Certaines bibliothèques étaient déjà compatibles avec notre carte, ce qui était avantageux. D'autres étaient exclusives à l'ESP32, ce qui était un inconvénient, mais nous pouvons maintenant les utiliser également. Cette compatibilité a été un facteur dans notre décision, mais nous aurions continué avec l'ESP32 de toute manière.

Alessandro Ranellucci : en ce qui concerne le logiciel, nous avons la possibilité de créer un nouveau noyau, comme nous l'avons fait pour d'autres produits, tels que le RP2040, pour lequel nous avons développé notre propre noyau afin d'avoir un contrôle total sur le logiciel. Mais Espressif a réalisé un excellent travail au

Pin numbering was a challenge because the numbers were very tied to the ESP32 world. But we found a solution – logical pin numbers.

Martino Facchin

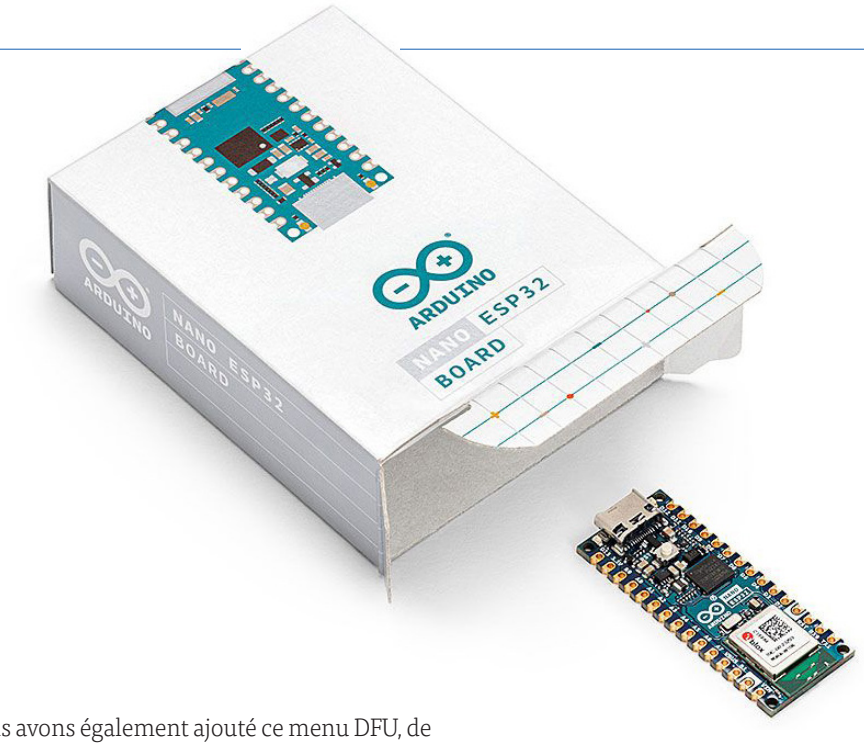
fil des ans et dispose d'une communauté solide. C'est pourquoi nous avons choisi de collaborer avec eux, une décision influencée par la force de l'écosystème.
Nous aspirons à rester neutres en matière de technologie, en évitant de nous engager exclusivement dans la gamme de microprocesseurs d'un seul fabricant. Notre objectif est d'offrir une plateforme Arduino polyvalente et interopérable, car c'est ainsi que les utilisateurs considèrent Arduino et leurs attentes envers la plateforme. C'est pourquoi nous expérimentons et recherchons en permanence de nouveaux produits à développer.
Martino Facchin : les personnes qui passent à Arduino à partir d'autres environnements, tels que les BSP (*Board Support Package*) ou les environnements intégrés avec des processus d'installation et de configuration laborieux, disent souvent qu'Arduino fonctionne, tout simplement. C'est, je crois, notre plus grande valeur, lorsqu'un utilisateur peut commencer à utiliser Arduino de la manière la plus simple et la plus rapide possible. Et comme Alessandro l'a mentionné, nous ne sommes pas fermés au monde extérieur. Il y a six ans, je ne pouvais pas affirmer cela parce qu'Atmel était notre principal sponsor, mais aujourd'hui nous sommes complètement indépendants.

Elektor : pouvez-vous nous parler des améliorations ou des changements que vous avez apportés à la plate-forme ESP32, pour la rendre plus compatible avec les objectifs de l'Arduino Nano ESP32 ?
Martino Facchin : nous avons modifié le processus de téléchargement habituellement utilisé pour l'ESP32, qui obligeait traditionnellement les utilisateurs à passer au module USB natif pour utiliser l'outil ESP, un processus qui n'était pas intégré de manière optimale à notre IDE. Désormais, lors du téléchargement d'un sketch, une méthode standard OTS (*Off The Shelf*) est employée. Le programme est téléchargé via *Device Firmware Update* (DFU), ce qui le déplace vers la deuxième partition par le biais d'une mise à jour OTA (*Over-The-Air*). Le programme d'amorçage est ensuite chargé de tenter un redémarrage à partir de cette deuxième partition. En cas de succès, le nouveau sketch est chargé, et dans le cas contraire, le sketch d'origine est conservé. Cette implémentation est une amélioration significative, car Espressif avait déjà envisagé plusieurs cas d'utilisation. Nous avons adapté leur approche pour développer un système plus rapide et plus fiable avec un mode de récupération.
Nous avons intégré le double clic (vous pouvez entrer en mode *recovery* en appuyant deux fois sur le bouton de réinitialisation), ce qui n'existait pas sur la carte ESP. Ce sont quelques modifications qui ont été bien acceptées par la communauté en raison de l'effort fourni. Même si le noyau communautaire est soutenu par Espressif, il s'agit d'un effort de la communauté, donc tout le monde était content de cela.

Alessandro Ranellucci : il y a deux autres contributions que nous avons faites à l'écosystème dans son ensemble, donc pas spécifiquement au noyau. Une partie du travail que nous avons effectué était du débogage, et nous avons également ajouté la prise en charge de MicroPython pour l'ESP32.

Elektor : en ce qui concerne le débogage, est-ce sur cette carte qu'il y a un accès à l'ESP par des points de soudure, ou est-ce que c'est sur une autre carte ?
Martino Facchin : non, vous pouvez déboguer cette carte directement via USB, Espressif a eu la gentillesse de le proposer. Via l'USB, vous avez un port série, le CDC, les choses habituelles, et aussi une interface JTAG. Vous pouvez interagir avec l'interface JTAG en utilisant la liaison USB habituelle en même temps que la communication normale. Il n'est pas vraiment difficile de connecter un débogueur externe, mais ce n'est pas nécessaire. En fait, il est déjà intégré dans l'EDI . Il suffit de connecter la carte, de sélectionner « *Debug mode* (Hardware CDC) » dans le menu et d'appuyer sur le gros bouton *Debug*. En plus, un tas de choses impliquant OpenOCD et GDB seront exécutées, mais c'est transparent du point de vue de l'utilisateur.
Alessandro Ranellucci : notre contribution a été de concevoir une expérience utilisateur pour cela. Avec Arduino, c'est beaucoup plus facile de tester, de documenter, de sensibiliser, de diffuser auprès des utilisateurs et d'interagir ensuite avec Espressif que d'utiliser des outils de débogage professionnels complets.

Elektor : les détails de la mise en œuvre me rappellent des questions posées en ligne. Il y avait une confusion au sujet des ports lorsque l'on branchait la carte, car il y avait deux ports et les gens ne savaient pas quel port utiliser.
Martino Facchin : oui, ce n'est pas facile à expliquer, parce que la principale façon dont nous avons toujours dit aux gens de faire les choses a été de commencer par les bases. Par exemple, vous le faites de cette façon, et il y a un port série, et ainsi de suite, et ensuite vous passez à un sujet avancé. Les sujets plus complexes doivent être très bien expliqués. C'est pourquoi nous disposons d'une documentation sur la manière de procéder correctement au débogage. Nous ne nous contentons pas de laisser la fonctionnalité en place et de laisser les gens expérimenter. De plus, les gens ne lisent généralement pas la documentation, c'est pourquoi nous faisons de notre mieux pour éviter ces inconvénients et mettons tout à leur disposition. La documentation devrait toujours être lue, et tout y est très bien expliqué.
Alessandro Ranellucci : la version la plus récente de l'EDI Arduino, sortie il y a un mois, a apporté de nombreuses améliorations à la sélection des ports et de choix des cartes. La confusion concernant les ports multiples ou les ports qui changent après que vous avez effectué des opérations, etc., est gérée de manière plus conviviale.



Martino Facchin : nous avons également ajouté ce menu DFU, de sorte que nous disposons désormais d'une fonction de *pluggable discovery*. Cette détection à l'insertion est un concept très intéressant car vous pouvez découvrir des choses via le port série ou par WiFi via MDNS. Paul Stoffregen a ajouté *discovery* via HID à la version 1 de l'EDI Arduino parce que son programme d'amorçage était basé sur HID, mais il s'agissait toujours d'un correctif externe pour le programme d'installation Arduino. Nous avons décidé de rendre cela disponible pour tout le monde. Il a donc porté son *discoverer* sur la version 2 de l'EDI Arduino. Ensuite, nous avons eu un autre problème avec le Minima, car lorsque vous passez en mode *bootloader*, il ne montre pas de port série. C'était déroutant de ne pas le voir dans le menu. C'est pourquoi nous avons donc décidé d'ajouter notre *discoverer* au port DFU. L'ESP32 dispose d'un DFU pendant l'exécution, en raison de la façon dont vous téléchargez le code. Occasionnellement, il peut apparaître sur votre ordinateur, mais, comme je l'ai mentionné précédemment dans la documentation, tout est expliqué. Ce port est destiné à vos téléchargements et vous pouvez choisir l'un ou l'autre. Cela ne change rien au fait qu'il accepte les téléchargements.

Elektor : y a-t-il eu des changements de compatibilité en termes de bibliothèques de codes existants, lors de la transition vers la puce ESP32 pour le nouveau modèle Nano ?
Alessandro Ranellucci : oui c'est le cas, chaque fois que nous ajoutons une nouvelle architecture à la famille. Il y a les bibliothèques officielles de base qui doivent être portées, en particulier lorsqu'elles sont hautement optimisées, de sorte qu'elles fonctionnent avec un code de bas niveau. Dans ce cas, nous avons dû étendre certaines bibliothèques de base. Mais l'écosystème des bibliothèques pour l'ESP32 était beaucoup plus développé.

Elektor : pouvez-vous mettre en évidence les caractéristiques de sécurité ou les considérations qui vous ont amené à choisir l'ESP32 pour l'Arduino Nano ESP32, en particulier pour les applications IdO ?
Alessandro Ranellucci : je ne dirais pas que nous avons choisi l'ESP32 pour des raisons de sécurité. Bien sûr, l'ESP32 a quelques capacités, que nous utilisons partiellement pour le moment.

Martino Facchin : vraiment en partie, car nous n'utilisons pas de cryptage ou ce qu'ils appellent le « secure boot » parce que cela rend la vie de l'utilisateur extrêmement difficile. Vous devez signer chaque code binaire que vous produisez, et vous devez ensuite le modifier pour chaque carte, sinon cela n'a pas de sens. Nous ne l'appliquons donc pas, mais nous l'autorisons, bien sûr. Du point de vue de l'intégrateur, lorsque vous prenez ce produit et que vous voulez le rendre vraiment sûr, il a toutes les fonctionnalités pour le faire. Mais ce n'est pas la raison pour laquelle nous l'avons choisi.
Alessandro Ranellucci : habituellement, sur toutes nos cartes, nous avons un élément matériel sécurisé séparé. C'est la méthode que nous avons choisie pour tous nos produits. Dans le cas présent, nous ne l'avons pas mise en œuvre parce que le microcontrôleur principal, en théorie, possède ces capacités. Pour l'instant, nous avons décidé d'utiliser cette puce telle quelle, par souci de simplicité. Rien ne nous empêche de poursuivre le développement, bien entendu.
Martino Facchin : la même chose s'est produite avec le Portenta H7, par exemple, lorsque nous avons lancé le programme d'amorçage sécurisé (*secure bootloader*) et l'infrastructure de démarrage MCO pour fournir des mises à jour OTA et sécurisées. Il s'agissait d'une option à confirmer, et non de quelque chose que nous fournissons directement. À terme, je suis certain que nous fournirons également une sorte de documentation par le biais d'un menu.

Elektor : est-il prévu d'utiliser la puce de Espressif dans davantage de cartes Arduino, telles que la gamme PRO ?
Alessandro Ranellucci : je ne peux pas répondre pour la gamme PRO, mais en général, oui.
Martino Facchin : nous aimons le facteur de forme *u-blox* parce qu'il s'adapte vraiment bien aux Nano. Il est très petit. Nous avons utilisé leurs modules sur presque toutes les cartes Nano, soit comme puce d'accompagnement pour le WiFi, soit comme microcontrôleur principal pour le Nano BLE. Nous sommes donc très satisfaits de leur qualité en tant que fabricant. En même temps, bien sûr, sur la UNO R4, nous avions le module Espressif normal. Et oui, nous prévoyons de faire d'autres choses.

Elektor : pouvez-vous nous donner plus de détails sur la collaboration ou les contacts que vous avez eus avec l'équipe de développement de l'ESP32, pour garantir une intégration transparente dans l'écosystème Arduino ?
Alessandro Ranellucci : il y a une bonne équipe de développement chez Espressif. Elle fait un excellent travail en impliquant la communauté dans le processus de développement. Nous avons donc eu des réunions individuelles avec eux, ainsi que des appels récurrents. Nous avons également partagé un canal de communication sur Slack ou d'autres plateformes de communication, afin d'avoir un moyen direct et quotidien de se mettre à jour, de s'informer mutuellement des problèmes et d'arriver à un consensus avant de rendre le projet public sur GitHub. Il s'agissait d'une collaboration très étroite. Nous pouvons donc citer de nombreuses personnes d'Espressif qui nous ont aidés dans cette collaboration.
Alessandro Ranellucci : en fait, à un niveau plus élevé, Ivan Grokhotkov (vice-président de Software Platforms, Espressif) nous a aidés à faciliter toute la communication, et Pedro Minatel (Developer Advocate) a été d'une aide précieuse en nous aidant à parler à la bonne personne.

Elektor : à l'avenir, imaginez-vous l'évolution du partenariat entre Arduino et Espressif, et quelles sont les possibilités d'innovation et de collaboration que vous anticipez ?
Alessandro Ranellucci : Je dirais qu'au-delà des produits matériels, les deux équipes sont d'accord sur la nécessité de travailler étroitement à la normalisation de l'API. L'API est désormais plus interopérable entre les mondes Arduino et ESP32. C'est sur ce point que nous aimerions poursuivre notre collaboration.
Martino Facchin : de plus, Espressif s'est diversifié dans de nombreux domaines, tels que Rust. De nombreux développeurs travaillent sur Rust, qui n'est pas une priorité pour nous, mais nous pourrions obtenir des idées très intéressantes à partir du travail qu'ils font dans ce domaine. Il y a aussi le système d'exploitation Zephyr, dans lequel nous sommes tous deux partenaires. Au sein du comité de pilotage technique, nous pouvons prendre des décisions, tout comme Espressif. Tout le développement que nous entreprenons tous les deux est axé sur l'amélioration des prochains outils pour les générations futures. En fin de compte, nous travaillons dans le même but.

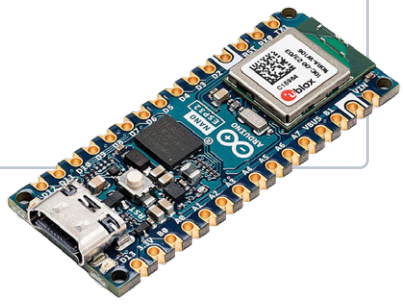
Elektor : lorsque vous comparez l'ESP32 à d'autres puces, pourriez-vous préciser les critères de référence, ou les mesures de performances, qui vous ont amené à conclure qu'il s'agissait du choix idéal pour l'Arduino Nano ESP32 ?
Martino Facchin : l'ESP32-S3 est une bonne puce. Sa consommation d'énergie n'est pas incroyablement faible, mais elle n'est pas mauvaise. Les caractéristiques de consommation ultra-faible existent si vous voulez vraiment les utiliser, mais nous n'allons pas dire à nos utilisateurs de tout mettre en veille profonde et d'utiliser la très faible consommation, même si c'est agréable de voir que ces choses existent. Je dirais que la performance n'est pas la principale raison d'être de la famille Nano. C'est plutôt la facilité d'utilisation. Nous avons eu l'occasion d'effectuer une comparaison avec le Portenta H7, par exemple, en ce qui concerne les capacités d'apprentissage automatique. Le Portenta H7 est environ sept fois plus rapide que l'ESP32, même à des vitesses d'horloge comparables. L'ESP32 a une vitesse d'horloge deux fois inférieure à celle de le Portenta H7. Le Nano existe pour la facilité d'utilisation, l'environnement, la disponibilité des bibliothèques, le facteur de forme, etc.

Elektor : compte tenu des capacités du système d'exploitation en temps réel de l'ESP32, comment cela est-il pris en compte dans la conception de l'ESP32 et dans son potentiel multitâche ?
Alessandro Ranellucci : je dirais qu'il s'agit d'un besoin croissant, avec comment utiliser plusieurs cœurs, le multitâche, etc. Ce besoin n'est pas actuellement très demandé par les *makers*, mais un écosystème devra être mis en place.
Martino Facchin : nous avons fait un effort de standardisation il y a un peu plus d'un an, avec ce que l'on appelle les threads Arduino. Si vous regardez sur GitHub, nous avons essayé de développer cette idée d'avoir des threads cachés derrière différents onglets dans votre EDI. Ainsi, vous avez un onglet avec `setup()` et `loop()`, comme d'habitude. Ensuite, vous avez un autre onglet avec un autre `setup()` et `loop()`, et cela représente votre deuxième thread. Ensuite, il y a un troisième, un quatrième, tout ce qui est nécessaire. Il reste le problème de la synchronisation des variables, où vous avez la restriction habituelle du *RTOS* (système d'exploitation en temps réel) qui vous oblige à utiliser des noms de variables différents entre deux threads. Nous l'avons résolu à un niveau supérieur. Nous avons donc caché cette complexité en utilisant *Mbed*, mais il ne s'agissait pas en fait d'un problème propre à *Mbed*. Nous voulions le porter sur plusieurs systèmes d'exploitation.
Si cela a réussi, la communauté des *makers* n'a pas du tout réagi à ce projet. Ce n'est donc probablement pas quelque chose dont les *makers* ont vraiment besoin. Le threading, c'est bien, mais cela complique aussi les choses. Et en même temps, bien sûr, vous pouvez utiliser FreeRTOS. Vous pouvez faire ce que vous voulez si vous êtes assez compétent, mais nous nous n'insistons pas pour cela, car nous avons toujours privilégié la facilité d'utilisation.

VF : Laurent Rauber — 230524-04

Produit

› Arduino Nano ESP32
www.elektor.fr/20562



ARDUINO CLOUD

What Arduino Cloud is

Develop from anywhere

- **NO CODE**
  With ready-to-use templates
- **LOW-CODE**
  Automatically generated sketches
- **FULL ARDUINO EXPERIENCE**
  Either offline with the UDE2 or online with the Cloud Editor
- **STORE YOUR SKETCHES ONLINE**
  Use your code in your favourite Arduino development environment

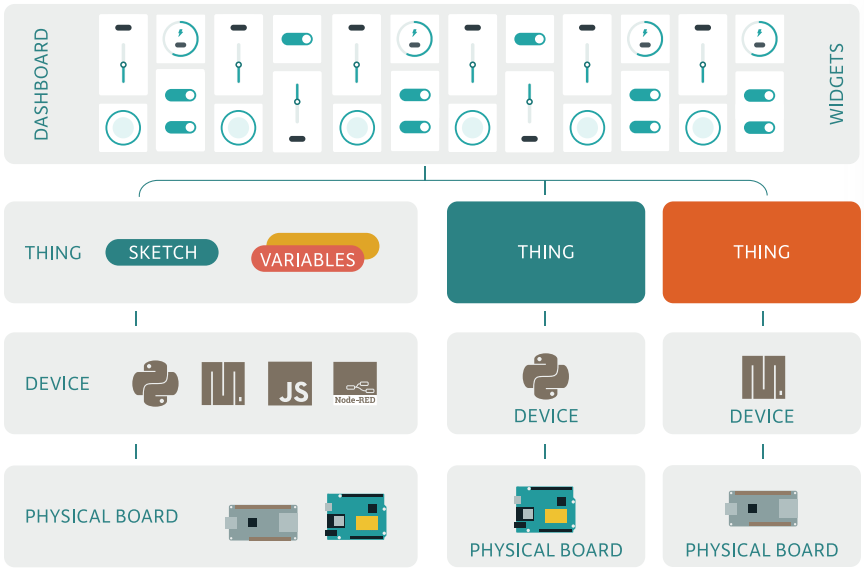
Program/Deploy

- **CABLE**
  Traditional USB programming
- **OVER-THE-AIR (OTA) UPDATES**
  Deploy your firmware wirelessly to your devices
- **MASS SCALE & AUTOMATION**
  With the Arduino Cloud CLI

Monitor & Control

- **CUSTOM DASHBOARDS**
  Using drag and drop widget
- **INSIGHTFUL WIDGETS**
  Interact with the devices and get real-time and historical data with dozens of widgets
- **MOBILE APP**
  Visualise your data in real-time from your phone with the IoT remote app

How does it work?



Compatible hardware

WITHIN ARDUINO DEVELOPMENT ENVIRONMENTS

ARDUINO
Cloud Applications can be developed using the Arduino Cloud Editor or Arduino IDE 2.

ESP32/ESP8266
+70% Of Arduino Cloud active users use ESP-based boards.

OUTSIDE ARDUINO DEVELOPMENT ENVIRONMENTS

Use your favourite programming environment and language to connect your devices to the Cloud.

Third party platform integration

TRIGGER ACTIONS ON THIRD PARTY PLATFORMS

Connect your Arduino Cloud devices to external platforms such as IFTTT, Zapier and Google Services using webhooks and unlock endless possibilities.

Seamlessly integrate your IoT devices with over 2 000 apps, enabling tasks like recieving phone notifications, automating social media updates, streamlining data logging to external files, creating calendar events, or sending e-mail alerts.

Get 30% off on the yearly Maker plan with code ELEKTOR30*

cloud.arduino.cc/elektor