simuler l'ESP32 avec Wokwi

le jumeau numérique de votre projet

Uri Shaked, Wokwi

Wokwi est un simulateur pour les systèmes embarqués et les appareils IdO. Il fournit un environnement en ligne où vous pouvez configurer, déboguer et partager des projets ESP32 sans avoir besoin de matériel. Le simulateur fonctionne simplement dans un navigateur web et peut simuler toutes les puces de la famille ESP32 : l'ESP32 classique, ainsi que les versions S2, S3, C3, C6 et H2. Il est également possible de simuler d'autres familles de microcontrôleurs.

Wokwi vous permet de créer un jumeau numérique de votre projet : vous pouvez simuler différents appareils d'entrée et de sortie [1], tels que des écrans LCD, des capteurs, des moteurs, des LED, des boutons-poussoirs, des haut-parleurs et des potentiomètres, et vous pouvez même créer vos propres modèles de simulation pour de nouveaux appareils. Avec le simulateur, vous pouvez modifier et développer votre micrologiciel plus rapidement même si vous ne disposez pas du matériel, vous pouvez utiliser des outils de débogage puissants, partager vos projets et collaborer facilement avec d'autres développeurs.

Commencez votre aventure Wokwi

Vous pouvez utiliser votre langage de programmation préféré avec Wokwi. Si vous êtes débutant, MicroPython ou Arduino Core est probablement un bon choix. Les professionnels et les utilisateurs expérimentés peuvent utiliser ESP-IDF, Rust et des RTOS embarqués tels que Zephyr ou NuttX.
Il est recommandé de consulter quelques exemples existants pour avoir une idée de ce que vous pouvez réaliser avec Wokwi :

- MicroPython [2]
- ESP32 avec Arduino Core [3]
- Rust [4]
- DeviceScript (TypeScript pour ESP32) [5]

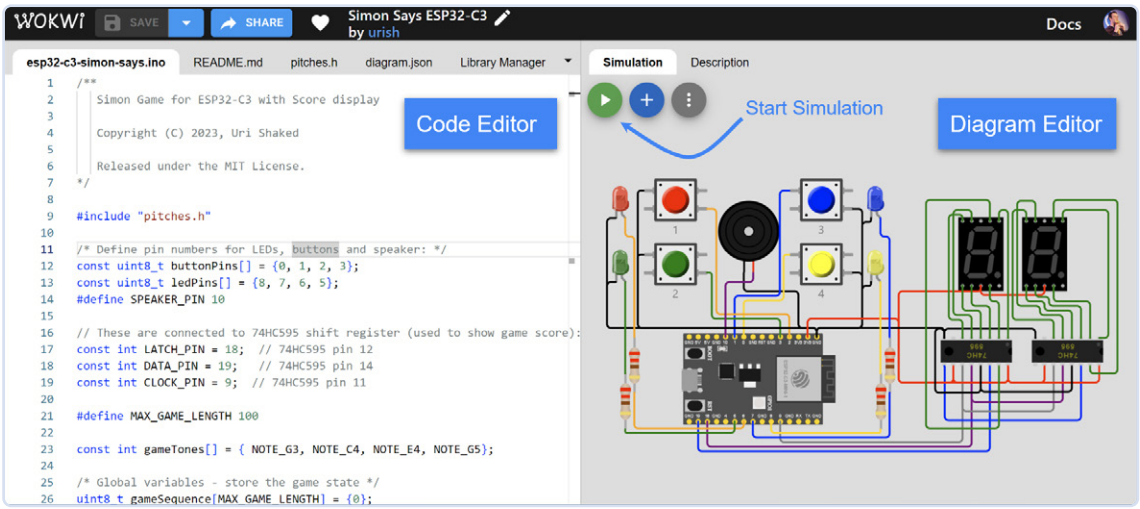

Figure 1. Le jeu « Jacques a dit » sur l'interface utilisateur de Wokwi.

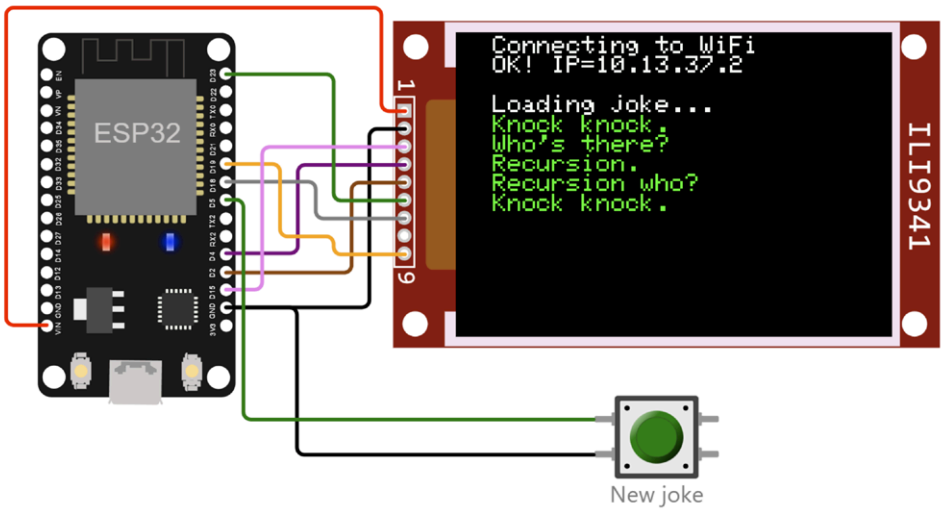

Figure 2. Exemple de simulation Wifi : client [8].

Ouvrez un exemple et appuyez sur le bouton vert *play* pour lancer la simulation et commencer à utiliser le projet. Certains projets disposent également d'un fichier `README` contenant plus d'informations sur le projet et sur la manière de l'utiliser. Pour créer un nouveau projet, rendez-vous sur [6]. Choisissez ensuite un microcontrôleur et un langage de programmation et cliquez sur *Start*.

Interface utilisateur de Wokwi

La version web de Wokwi se compose de deux parties : sur le panneau de gauche, vous pouvez éditer le code source (l'éditeur de code), et à droite, vous pouvez dessiner le schéma en ajoutant différents appareils/composants et en les connectant au microcontrôleur (*Diagram Editor*), voir la **figure 1**.

Diagram Editor

Ajoutez des éléments au diagramme en cliquant sur le bouton bleu + et en choisissant le composant souhaité dans la liste. Faites glisser les composants jusqu'à la position souhaitée. Pour dessiner un fil, cliquez sur la broche où le fil commence, puis sur la broche cible. Si vous voulez donner au fil un tracé spécifique, après avoir cliqué sur la broche de départ, cliquez sur l'endroit où vous voulez le connecter.
Pour des résultats plus précis, vous pouvez activer la grille en appuyant sur *G*. Cela aligne tous les composants et les fils sur une grille de 0,1 pouce (2,54 mm) - qui est l'espacement standard des broches d'un connecteur. Quelques raccourcis utiles : *D* duplique le composant sélectionné, *R* le fait pivoter, les chiffres *0* à *9* changent la couleur d'un fil, d'une LED ou d'un bouton-poussoir et si vous appuyez sur *Shift*, vous pouvez utiliser la souris pour sélectionner plusieurs composants en même temps. Pour plus de raccourcis, voir le manuel de l'éditeur de schémas [7].

Gestionnaire de bibliothèques

Utilisez le gestionnaire de bibliothèques (Library Manager) pour ajouter rapidement des bibliothèques Arduino standard à un projet. Les utilisateurs payants peuvent également télécharger leurs propres bibliothèques Arduino et les inclure dans leur projet.
Pour les autres langages de programmation, vous pouvez inclure des bibliothèques en modifiant les fichiers de configuration du projet (par exemple *Cargo.toml* pour Rust), ou en incluant le code source de la bibliothèque directement dans votre projet (par exemple pour MicroPython).

Simulation Wifi

L'ESP32 dispose d'une fonction Wifi intégrée, c'est pourquoi il est populaire dans les projets IdO. Wokwi simule la fonction Wifi de la puce. Le simulateur (**figure 2**) crée un point d'accès virtuel appelé *Wokwi-GUEST*. C'est un point d'accès ouvert (c'est-à-dire sans mot de passe). Le point d'accès est connecté à internet, vous pouvez donc connecter les projets simulés à des serveurs HTTP, des brokers MQTT et d'autres services cloud tels que Firebase, ThingsSpeak, Blynk et Azure IoT (**figure 3**).

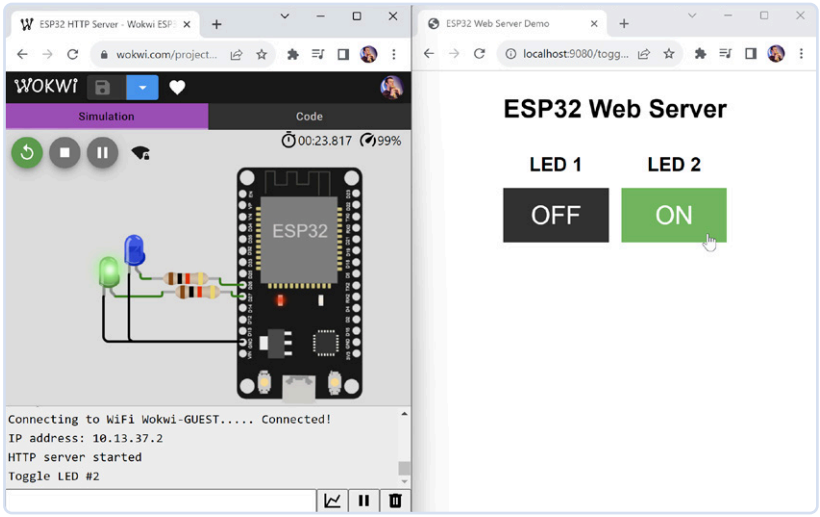

Figure 3. Exemple de simulation Wifi : serveur web [9].

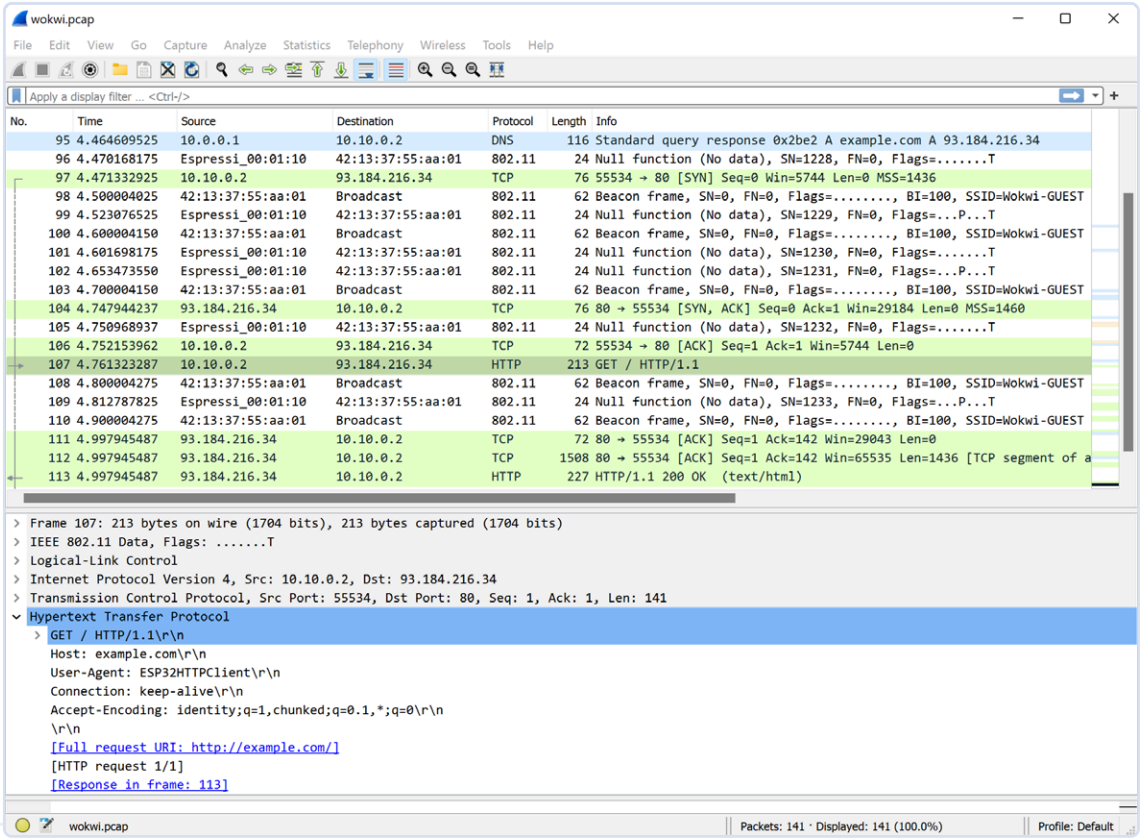

Figure 4. Visualisation du trafic de données Wifi.

Les utilisateurs payants peuvent également utiliser une passerelle IdO spéciale [10] sur leur ordinateur, qui leur permet de se connecter à des serveurs locaux depuis la simulation, et de connecter leur ordinateur à un serveur HTTP (ou tout autre type de serveur) fonctionnant dans le simulateur. Sous le capot, Wokwi simule une pile réseau complète : cela va de la couche MAC 802.11 la plus basse, en passant par les couches IP et TCP/UDP, jusqu'aux protocoles tels que DNS, HTTP, MQTT, CoAP, etc. Nous pouvons visualiser les données brutes du réseau [11] dans un analyseur de protocole réseau tel que Wireshark (**figure 4**).

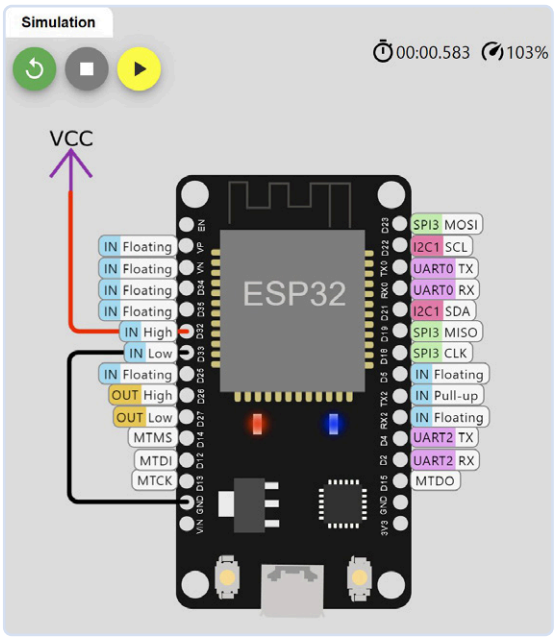

Figure 5. Simulation en pause : on peut voir l'état de chaque broche.

Pin Function Debugger

L'ESP32 offre une configuration flexible des broches : le logiciel peut choisir quel périphérique est connecté à chacune des broches en configurant IO MUX, GPIO Matrix et les périphériques basse consommation/RTC. Définir la configuration des broches au niveau logiciel peut être très pratique, mais cela complique la tâche de savoir quelle est la configuration réelle des broches dans le micrologiciel. Heureusement, dans Wokwi, nous pouvons simplement pauser la simulation et voir immédiatement la fonction et l'état actuels de chaque broche (**figure 5**).

Intégration avec Visual Studio Code

L'utilisation de Wokwi dans un navigateur web est très pratique pour apprendre le prototypage rapide, et pour partager des projets. Mais pour les projets plus complexes, il est recommandé d'utiliser Wokwi en combinaison avec votre EDI et outils de développement habituels. Wokwi s'intègre parfaitement à Visual Studio Code. Il suffit d'installer l'extension *Wokwi For VS Code* [12] et de créer un simple fichier de configuration [13] qui indique à Wokwi où se trouve votre micrologiciel compilé.

Vous pouvez également intégrer Wokwi au débogueur intégré de VS Code en ajoutant quelques lignes à sa configuration [14]. Contrairement au matériel réel, Wokwi dispose d'un nombre illimité de points d'arrêt, ce qui permet un débogage efficace (**figure 6**).

Pour les projets IdO, vous pouvez utiliser le transfert de port TCP [15]. Cela vous permet de connecter votre ordinateur à un serveur web (ou tout autre type de serveur TCP) fonctionnant sur la puce ESP32 simulée.

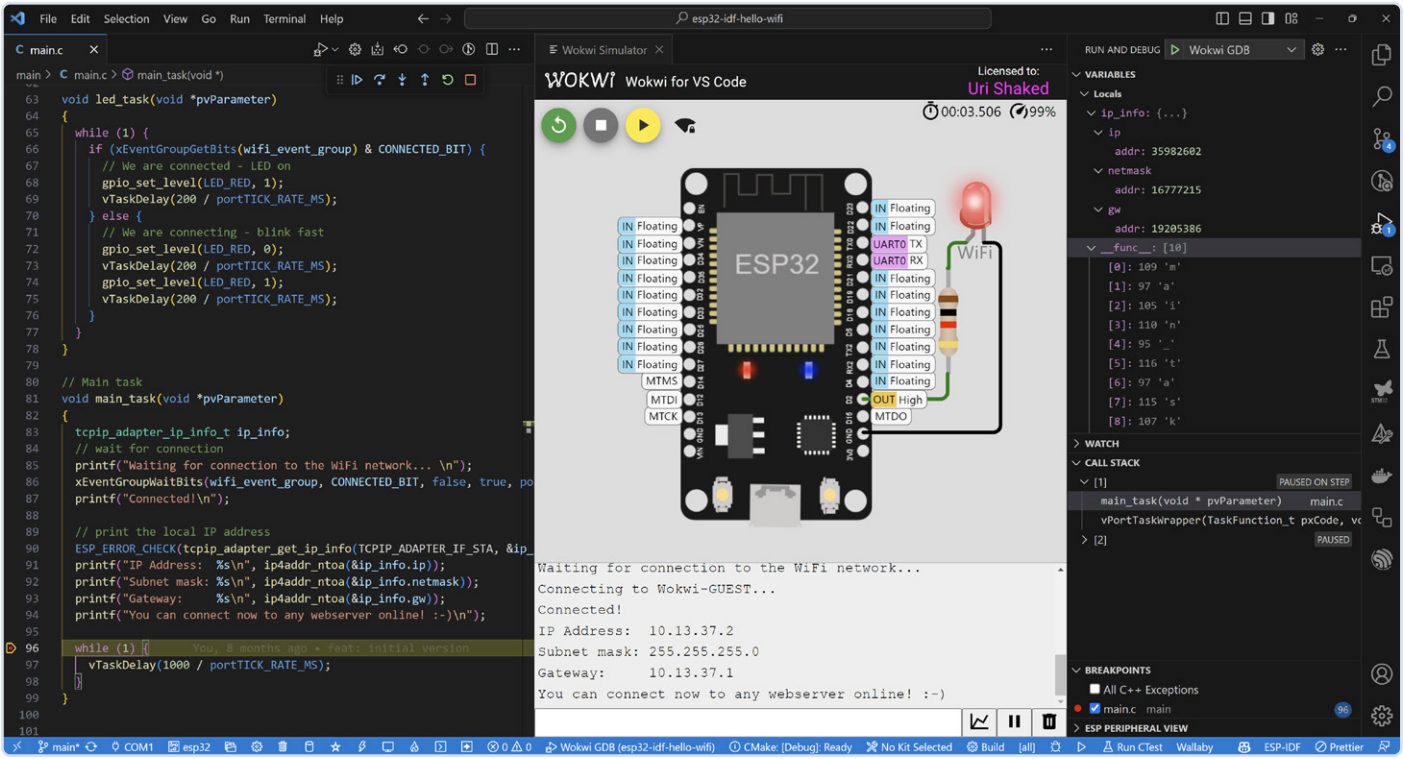

Figure 6. Débogage d'un projet ESP-IDF avec VS Code et Wokwi.

Intégration avec l'EDI Espressif

Les adeptes de l'EDI Espressif peuvent également créer une configuration de démarrage pour exécuter un projet dans le simulateur. La sortie (UART) du programme est alors envoyée vers la console de l'EDI. Voir Comment utiliser le simulateur Wokwi avec Espressif-IDE [16] pour plus de détails. [16].

Custom Chips

Custom Chips nous permettent d'étendre les fonctions de Wokwi avec de nouveaux composants. Nous pouvons simuler des capteurs, des écrans, des mémoires, des équipements de test (**figure 7**) et même du matériel personnalisé.

Pour créer notre propre circuit intégré, nous définissons son brochage dans un fichier JSON et nous écrivons du code pour implémenter la logique de la puce. Le code est généralement écrit en C, et l'API ressemble aux couches d'abstraction matérielle (HAL) courantes pour les puces intégrées, de sorte que les développeurs de micrologiciels seront en mesure de l'utiliser. D'autres langages tels que Rust, AssemblyScript et Verilog font également l'objet d'une prise en charge expérimentale. Pour plus d'informations et d'exemples, voir la documentation *API Chips* [17].

Wokwi pour les circuits intégrés

La simulation peut faire gagner beaucoup de temps lors du prototypage, mais elle peut aussi aider à la recherche de défauts pendant le développement. L'intégration continue (CI) est une approche moderne du développement logiciel : construire et tester le projet après chaque modification du code. Mais les tests sur du matériel réel prennent du temps et sont difficiles à automatiser. La situation est encore plus délicate si l'on souhaite une intégration complète du système. Pensez aux tests automatisés du Wifi ou à la capture de l'image sur un écran LCD. Il est très difficile d'y parvenir de manière stable et fiable.

Wokwi pour les CI résout tout cela. Si vous utilisez GitHub Actions, `wokwi-ci-action` [18] vous permet d'intégrer la simulation dans votre flux de travail existant avec seulement quelques lignes de code. Pour les autres environnements de CI, `wokwi-cli` [19] fournit une interface de ligne de commande simple pour exécuter et tester des micrologiciels par simulation. Le serveur de simulation de CI est disponible en tant que service cloud géré et peut également être déployé sur votre poste de travail.

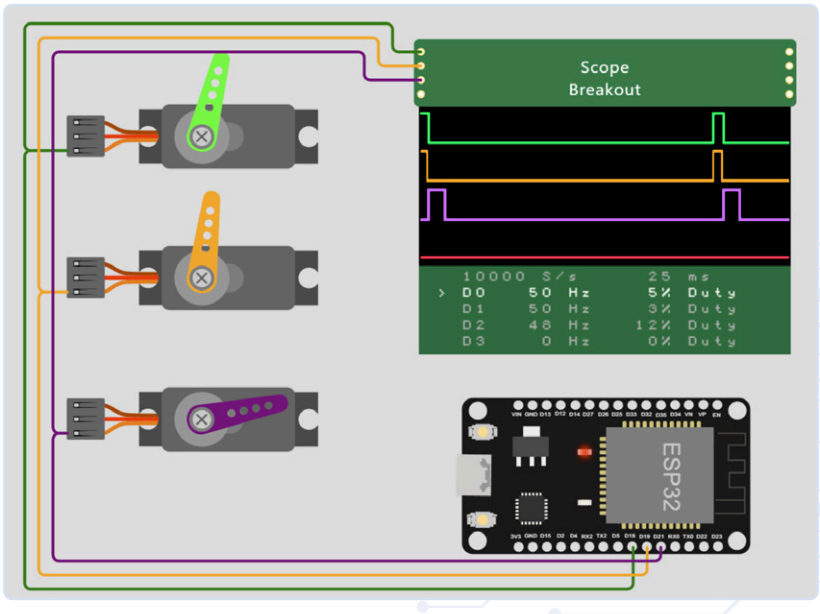

Figure 7. Un scope numérique créé par un utilisateur avec l'API Custom Chips.

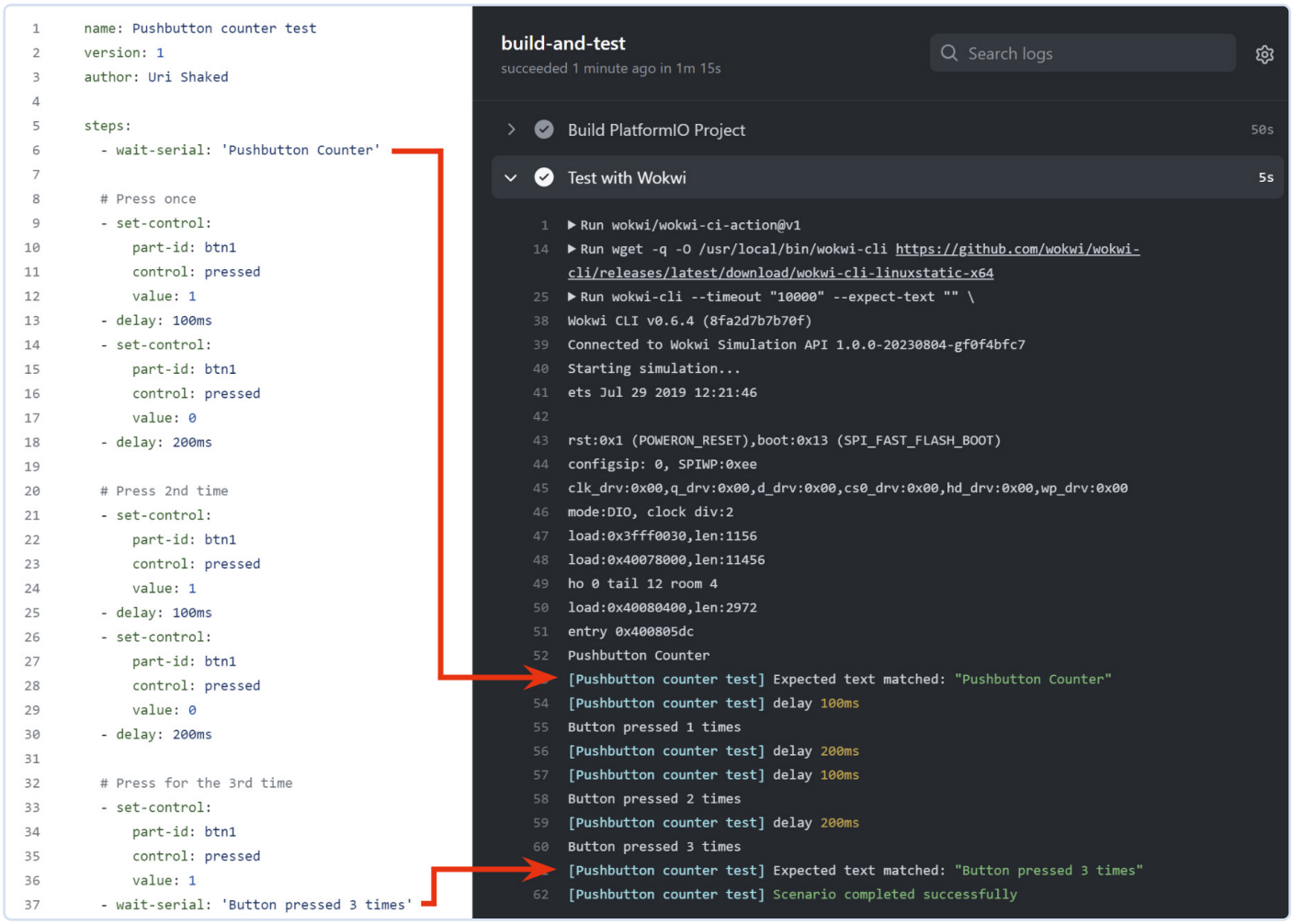

Figure 8. Le code d'exemple d'automatisation [20] (à gauche) avec la sortie de GitHub Action correspondante (à droite).

Scénarios d'automatisation

Les scénarios d'automatisation (**figure 8**) vous permettent d'effectuer des tests et des vérifications complexes sur le micrologiciel. Ils sont faciles à écrire : chaque scénario consiste en un fichier YAML contenant une série de commandes telles que l'appui sur un bouton, le réglage d'une valeur de capteur de température ou la recherche d'une chaîne de caractères dans la sortie série.

Limites

Wokwi offre de nombreuses fonctions, mais il a aussi quelques limites. L'objectif principal du simulateur est d'exécuter du code embarqué. Le simulateur est principalement numérique et la prise en charge de la simulation analogique est limitée. Wokwi ne vous avertira pas si vous oubliez une résistance de limitation de courant ; il ne dégage pas non plus de fumée magique (mais qui sait ce que l'avenir nous réserve...).

230523-04

Questions ou commentaires ?

Contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Uri Shaked est un maker chevronné. Il travaille actuellement sur Wokwi, une plateforme en ligne de simulation de systèmes IdO et embarqués, et sur Tiny Tapeout, qui rend la fabrication de vos propres ASIC abordable et accessible.

Produits

- **ESP32-C3-DevKitM-1**
  www.elektor.fr/20324
- **Koen Vervloesem,** *Getting Started with ESPHome* **(Elektor, 2021)**
  www.elektor.fr/19738

LIENS

[1] Wokwi Supported Hardware : https://docs.wokwi.com/getting-started/supported-hardware
[2] Wokwi Online Embedded Python Simulator : https://wokwi.com/micropython
[3] Wokwi Online ESP32 Simulator : https://wokwi.com/esp32
[4] Wokwi Online Embedded Rust Simulator : https://wokwi.com/rust
[5] Wokwi Online Devicescript Simulator: https://wokwi.com/devicescript
[6] Page de démarrage d'un nouveau projet : https://wokwi.com/projects/new
[7] Guide du Diagram Editor : https://docs.wokwi.com/guides/diagram-editor#keyboard-shortcuts
[8] Exemple de simulation Wifi : https://wokwi.com/projects/342032431249883731
[9] Exemple de serveur Web ESP32 : https://wokwi.com/projects/320964045035274834
[10] Application de passerelle privée : https://docs.wokwi.com/guides/esp32-wifi#the-private-gateway
[11] Visualisation du trafic de données Wifi avec Wireshark : https://tinyurl.com/wsviewtraffic
[12] Esxtension Wokwi for VS Code : https://tinyurl.com/wokwivsext
[13] Configuring Your Project: https://docs.wokwi.com/vscode/project-config
[14] Débogage du code : https://docs.wokwi.com/vscode/debugging
[15] TCP Port Forwarding : https://docs.wokwi.com/vscode/project-config#iot-gateway-esp32-wifi
[16] Comment utiliser le simulateur Wokwi avec Espressif-IDE : https://tinyurl.com/wokwiespide
[17] Démarrer avec Wokwi Custom Chips C API : https://docs.wokwi.com/chips-api/getting-started
[18] Wokwi CI Action : https://github.com/wokwi/wokwi-ci-action
[19] Wokwi CLI : https://github.com/wokwi/wokwi-cli
[20] Automation Scenario Code : https://tinyurl.com/pushcnttst