Simuleer ESP32 met Wokwi

de digitale tweeling van je project

Uri Shaked, Wokwi

Wokwi is een simulator voor embedded systemen en IoT-apparaten. Het biedt een online-omgeving waar je ESP32-projecten kunt opzetten, debuggen en delen zonder dat je tastbare hardware nodig hebt. De simulator draait gewoon in een webbrowser en kan alle chips uit de ESP32-familie simuleren: de klassieke ESP32, maar ook de S2-, S3-, C3-, C6- en H2-varianten. En ook andere microcontrollerfamilies kunnen worden gesimuleerd.

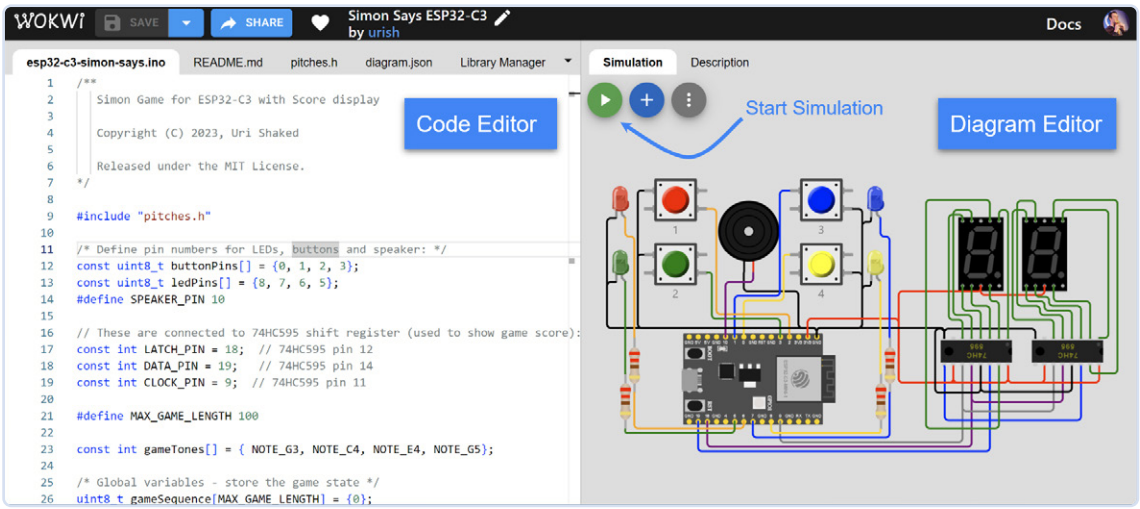
Met Wokwi kun je een digitale tweeling maken van je project: je kunt verschillende invoer- en uitvoerapparaten simuleren [1], zoals LC-displays, sensoren, motoren, LED's, drukknoppen, luidsprekers en potentiometers, en je kunt zelfs eigen simulatiemodellen maken voor nieuwe apparaten. Met de simulator kun je sneller itereren en aan je firmware werken; zelfs als de hardware niet beschikbaar is, kun je gebruik maken van krachtige debugtools, je projecten delen en gemakkelijk samenwerken met andere ontwikkelaars.

Start je Wokwi-avontuur

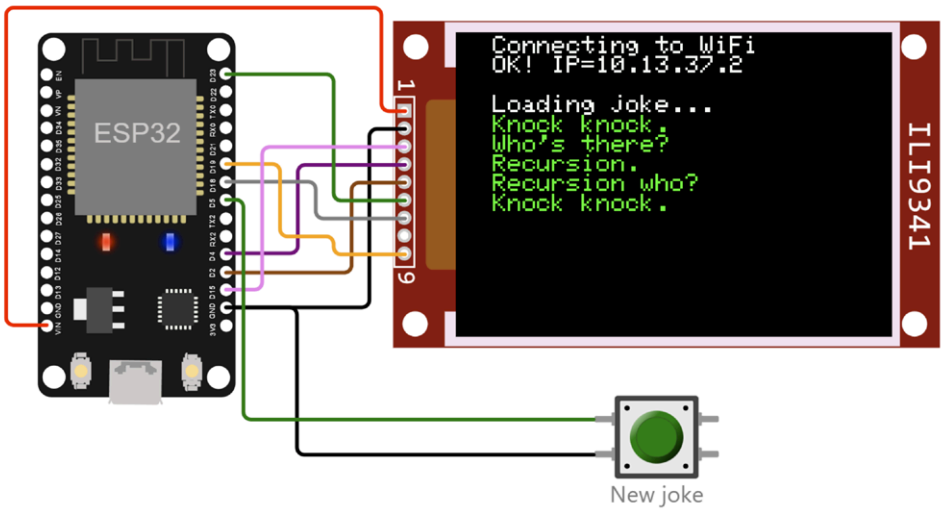
Je kunt je favoriete programmeertaal gebruiken met Wokwi. Als je een beginner bent, is MicroPython of Arduino Core waarschijnlijk een goede keus. Voor professionals en geavanceerde gebruikers zijn er de ESP-IDF, Rust en embedded RTOS zoals Zephyr of NuttX.

Het is een goed idee om enkele bestaande voorbeelden te bekijken om ideeën te krijgen van wat je zoal met Wokwi kunt bouwen:

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



Figuur 1. De gebruikersinterface van Wokwi met "Simon Says".



Figuur 2. WiFi-simulatie: client [8].

Open een voorbeeld en druk op de groene *play*-knop om de simulatie te starten en met het project aan de slag te gaan. Sommige projecten hebben ook een `README`-bestand met meer informatie over het project en hoe ermee te werken. Om een nieuw project aan te maken, ga je naar [6]. Kies dan een microcontroller en een programmeertaal en klik op *Start*.

De gebruikersinterface van Wokwi

De webversie van Wokwi bestaat uit twee delen: In het linker paneel kun je de broncode bewerken (de *Code Editor*), en rechts teken je het schema van het project door verschillende apparaten/componenten toe te voegen en ze met de microcontroller te verbinden (de Diagram Editor), zie **figuur 1**.

De Diagram Editor

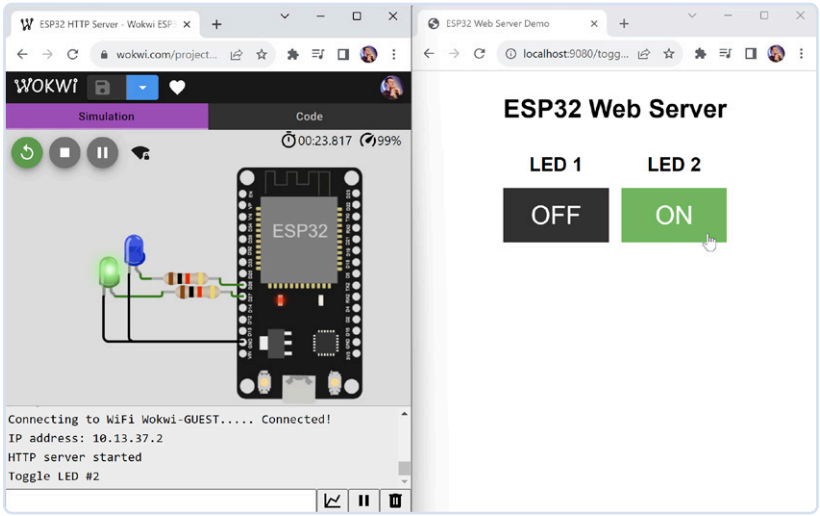
Voeg elementen toe aan het schema door op de blauwe + knop te klikken en het gewenste onderdeel uit de lijst te kiezen. Sleep de onderdelen naar de gewenste positie. Om een draad te tekenen, klikt je op de pin waar de draad begint en dan op de pin waar de draad naartoe moet gaan. Als je de draad een specifieke route wilt geven, klik je na het aanklikken van de begin-pin op de plaats waar je hem wilt hebben.
Voor een net resultaat kun je het raster inschakelen door op *G* te drukken. Dat lijnt alle componenten en draden uit op een raster van 0,1" (2,54 mm) – dat is de standaard pinafstand van een header. Enkele nuttige sneltoetsen: *D* dupliceert het geselecteerde onderdeel, *R* roteert het, de cijfers 0 t/m 9 veranderen de kleur van een draad, LED of drukknop en als je Shift indrukt kun je met de muis je meerdere componenten tegelijk selecteren. Zie voor meer sneltoetsen de handleiding van de Diagram Editor [7].

De Library Manager

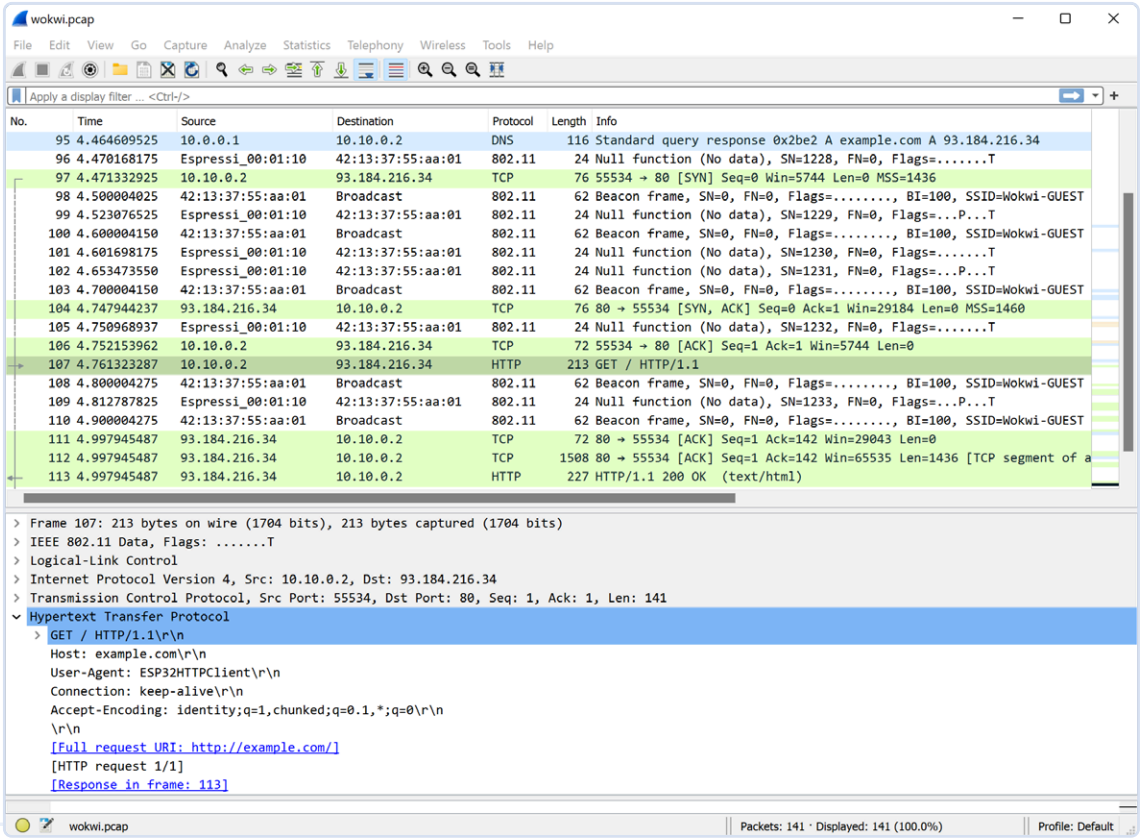
Gebruik de Library Manager om snel standaard Arduino-bibliotheken toe te voegen aan een project. Betalende gebruikers kunnen ook eigen Arduinobibliotheken uploaden en meenemen in hun project. Voor andere programmeertalen kun je bibliotheken meenemen door de project-configuratiebestanden aan te passen (bijvoorbeeld *Cargo.toml* voor Rust), of door de broncode van de bibliotheek rechtstreeks op te nemen in je project (bijvoorbeeld voor MicroPython).

WiFi-simulatie

De ESP32 heeft ingebouwde WiFi-functionaliteit, daarom is hij populair voor IoT-projecten. Wokwi simuleert de WiFi-functionaliteit van de chip. De simulator (**figuur 2**) maakt een virtueel access point met de naam *Wokwi-GUEST* aan. Het is een open access point (dus zonder wachtwoord). Het access point is verbonden met het internet, zodat je de gesimuleerde projecten met http-servers, MQTT-brokers en andere cloudd-services zoals Firebase, ThingsSpeak, Blynk en Azure IoT kunnen verbinden (**figuur 3**).

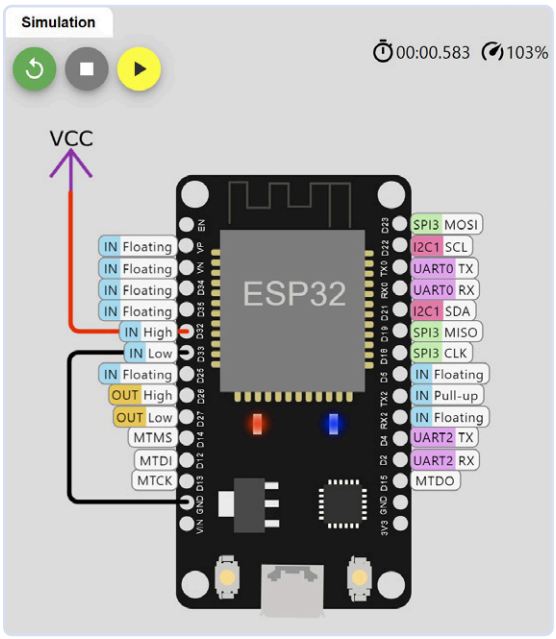


Figuur 3. WiFi-simulatie: webserver [9].



Figuur 4. Bekijk het WiFi-dataverkeer.

Betalende gebruikers kunnen ook een speciale IoT-Gateway [10] draaien op hun computer, waarmee ze verbinding kunnen maken met lokale servers vanuit de simulatie, en hun computer kunnen verbinden met een http-server (of een ander soort server) die draait in de simulator. Onder de motorkap simuleert Wokwi een complete netwerk-stack: dat gaat van de laagste 802.11 MAC-laag, via de IP- en TCP/UDP-laag, helemaal omhoog tot aan de protocollen zoals DNS, HTTP, MQTT, CoAP enzovoort. We kunnen de ruwe netwerkdata [11] bekijken in een netwerkprotocol-analyzer zoals Wireshark (**figuur 4**).



Figuur 5. Simulatie gepauzeerd: van elke pin is de toestand te zien.

Pin Function Debugger

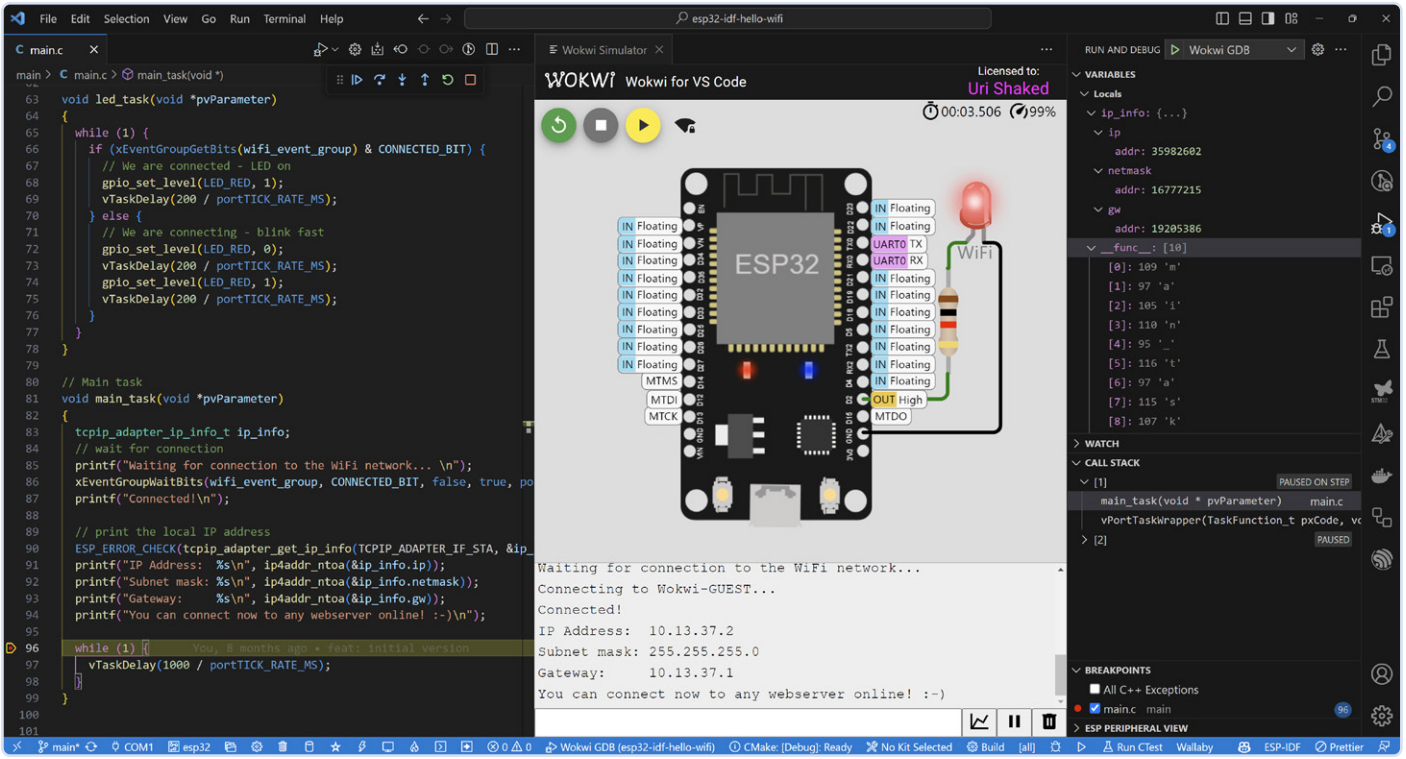
De ESP32 biedt een flexibele pinconfiguratie: de software kan kiezen welke periferie is verbonden met elk van de pinnen door de IO MUX, GPIO-Matrix en de low-power/RTC periferie te configureren. De pinconfiguratie softwarematig vastleggen kan heel handig zijn, maar het maakt het wel ingewikkelder om uit te vinden wat de werkelijke pinconfiguratie van de firmware is. Gelukkig kunnen we in Wokwi gewoon de simulatie pauzeren en direct de actuele functie en toestand van elke pin zien (**figuur 5**).

Integratie met Visual Studio Code

Het gebruik van Wokwi in de webbrowser is heel leerzaam en snel, en ook het delen van projecten is gemakkelijk. Maar voor ingewikkelde projecten is het toch aan te raden om Wokwi te gebruiken in combinatie met de gewone IDE en development tools. Wokwi integreert naadloos met Visual Studio Code. Je hoedt alleen de Wokwi for VS Code–extensie te installeren [12] en een eenvoudig configuratiebestand aan te maken [13] dat Wokwi vertelt waar je gecompileerde firmware staat.

We kunnen Wokwi ook integreren met de ingebouwde debugger van VS Code door een paar regels aan de configuratie toe te voegen [14]. In tegenstelling tot echte hardware heeft Wokwi een onbeperkt aantal breakpoints, zodat je efficiënt kunt debuggen (**figuur 6**).

Voor IoT-projecten kunt je TCP port-forwarding gebruiken [15]. Daarmee kun je je computer verbinden met een webserver (of een ander soort TCP-server) die draait in de gesimuleerde ESP32-chip.



Figuur 6. Debuggen van een ESP-IDF-project met VS Code en Wokwi.

Integratie met Espressif IDE

Liefhebbers van de Espressif IDE kunnen ook een startconfiguratie maken om een project in de simulator te draaien. De (UART-)uitvoer van het programma gaat dan naar het console van de IDE. Zie *How to Use Wokwi Simulator with Espressif-IDE* [16] voor alle details.

Eigen IC's

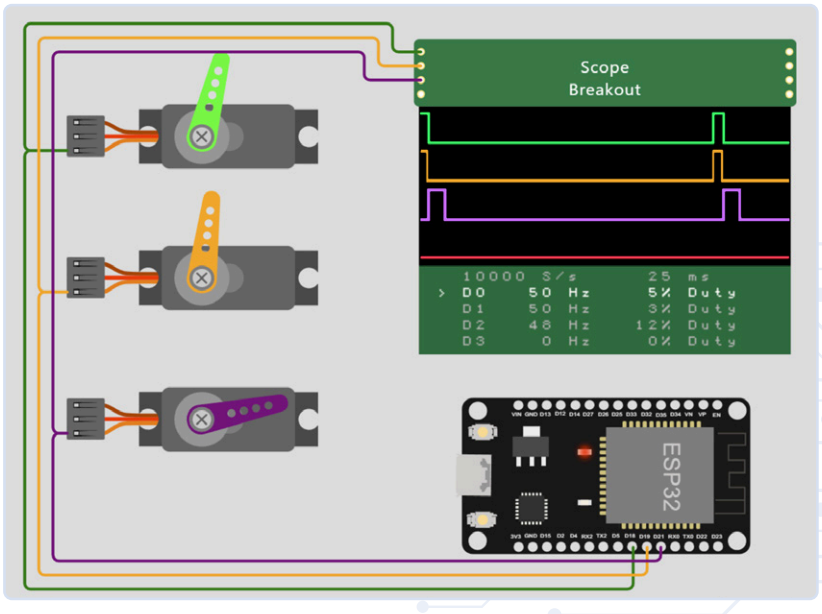
Met *Custom Chips* kunnen we de functionaliteit van Wokwi uitbreiden met nieuwe onderdelen. We kunnen sensoren, displays, geheugens, testinstrumenten (**figuur 7**) en zelfs eigen hardware simuleren.

Om een eigen IC te maken, leggen we de pinning ervan vast in een JSON-file en schrijven code om de logica van de chip te implementeren. De code wordt meestal geschreven in C, en de API lijkt op gebruikelijke hardware-abstractielagen (HAL) voor embedded chips, dus firmware-ontwikkelaars zullen daar mee uit de voeten kunnen. Er is ook experimentele ondersteuning voor andere talen zoals Rust, AssemblyScript en Verilog. Zie voor meer informatie en voorbeelden de Chips API-documentatie [17].

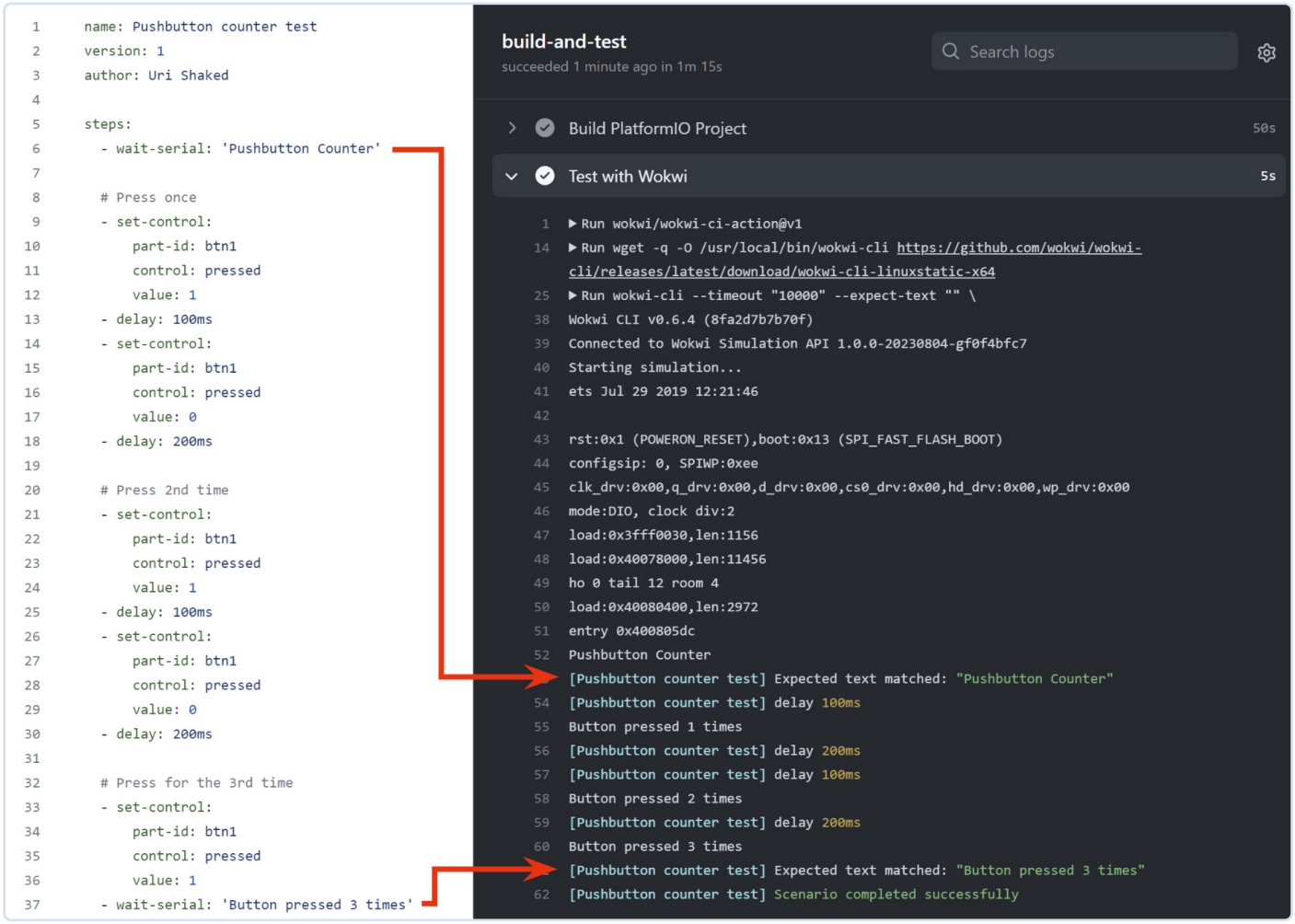
Wokwi voor CI

Simulatie kan veel tijd besparen bij het maken van prototypes, maar het kan ook helpen bij het foutzoeken tijdens de ontwikkeling. Continuous Integration (CI) is een moderne aanpak bij het ontwikkelen van software: bouw en test het project na elke aanpassing in de code. Maar testen met echte hardware is tijdrovend en moeilijk te automatiseren. Het wordt zelfs nog lastiger als je volledige systeemintegratie wenst. Denk maar eens aan geautomatiseerde WiFi-tests of het vastleggen van het beeld op een LC-display. Dat is heel lastig stabiel en betrouwbaar te realiseren.

Wokwi voor CI lost dat allemaal op. Als je GitHub Actions gebruikt, kun je met `wokwi-ci-action` [18] simulatie integreren in je bestaande workflow met slechts een paar regels code. Voor andere CI-omgevingen biedt `wokwi-cli` [19] een eenvoudige commandoregel-interface voor het uitvoeren en testen van firmware in simulatie. De CI Simulation Server is beschikbaar als een beheerde clouddienst, en kan ook op de werkplek worden ingezet.



Figuur 7. Een digitale 'scoop die een gebruiker heeft gemaakt met behulp van de Custom Chips API.



Figuur 8. De code voor het automatiseringsscenario [20] (links) met de corresponderende GitHub Action-uitvoer (rechts).

Automatiseringsscenario's

Met automatiseringsscenario's (**figuur 8**) kun je ingewikkelde tests en verificaties uitvoeren op firmware. Die zijn gemakkelijk te schrijven: elk scenario is een YAML-bestand met een reeks commando's zoals het indrukken van een knop, instellen van een temperatuursensorwaarde of zoeken naar een string in de seriële uitvoer.

Beperkingen

Wokwi biedt veel mogelijkheden, maar heeft ook enkele beperkingen. De belangrijkste focus van de simulator is het draaien van embedded code. De simulator is vooral digitaal en de ondersteuning voor analoge simulatie is beperkt. Wokwi zal je niet waarschuwen als je een stroombegrenzingsweerstand vergeet; er komt ook geen magische rook vrij (maar wie weet wat de toekomst nog brengt...).

vertaling: Evelien Snel — 230523-03

Vragen of opmerkingen?

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

Over de auteur

Uri Shaked is al heel lang maker. Op dit moment werkt hij aan Wokwi, een online IoT- en embedded system simulatieplatform, en aan Tiny Tapeout, dat het fabriceren van eigen ASIC's betaalbaar en toegankelijk maakt.

Gerelateerde producten

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

WEBLINKS

[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] Nieuw Arduino Uno Project startpagina: https://wokwi.com/projects/new
[7] Diagram Editor sneltoetsen: https://docs.wokwi.com/guides/diagram-editor#keyboard-shortcuts
[8] _esp32-jokes-api_ W-Fi-simulatievoorbeeld: https://wokwi.com/projects/342032431249883731
[9] ESP32 HTTP-server voorbeeld: https://wokwi.com/projects/320964045035274834
[10] ESP32 Wi-Fi Network: de Private Gateway-applicatie: https://docs.wokwi.com/guides/esp32-wifi#the-private-gateway
[11] ESP32 Wi-Fi Network: WiFi-dataverkeer bekijken met Wireshark: https://tinyurl.com/wsviewtraffic
[12] Wokwi Simulator — Visual Studio Code Extension: https://tinyurl.com/wokwivsext
[13] Projectconfiguratie (_wokwi.toml_): https://docs.wokwi.com/vscode/project-config
[14] Wokwi — code debuggen: https://docs.wokwi.com/vscode/debugging
[15] Projectconfiguratie — IoT Gateway (Port Forwarding): https://docs.wokwi.com/vscode/project-config#iot-gateway-esp32-wifi
[16] Gebruik van de Wokwi Simulator met Espressif-IDE: https://tinyurl.com/wokwiespide
[17] Aan de slag met de 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] _wowkwi-cli_: https://github.com/wokwi/wokwi-cli
[20] _platform-io-esp32-counter-ci_ — Automation Scenario Code: https://tinyurl.com/pushcnttst