tutoriel ESP-Launchpad

flashaer des micrologiciels en quelques minutes

Dhaval Gujar, Espressif

Vous travaillez avec des puces Espressif ? ESP-Launchpad est un outil web qui simplifie l'évaluation et le test des micrologiciels. Observons-le de plus près.

Cet article présente ESP-Launchpad, un outil web qui permet de vérifier et de tester les micrologiciels pour les puces Espressif en toute simplicité. Il simplifie la configuration de l'environnement de développement, exploite les navigateurs web pour le flashage des micrologiciels et est destiné aussi bien aux développeurs qu'aux autres utilisateurs.

Pourquoi choisir ESP-Launchpad ?

Lorsque vous créez un projet open-source, vous souhaitez offrir aux utilisateurs un moyen simple de l'étudier. Pour les développeurs embarqués, cet objectif implique la configuration de l'hôte de développement et des outils de programmation, le clonage du projet, la compilation et la programmation du matériel.

Comme l'hôte de développement et l'environnement logiciel peuvent être différents, il y a de fortes chances qu'une de ces étapes se passe mal. Et même si tout fonctionne bien, la tâche est trop compliquée pour les utilisateurs non-développeurs qui veulent simplement tester le projet.

Si vous développez avec l'une des puces Espressif, vous avez désormais une meilleure possibilité de développer facilement votre projet. ESP-Launchpad simplifie ce processus et permet d'évaluer et de tester rapidement et facilement les micrologiciels développés pour la plateforme ESP.

À vos marques, prêts, ESP-Launchpad !

ESP-Launchpad est un outil web qui utilise les fonctions de navigation les plus modernes pour flasher de manière simple et sans fioritures des micrologiciels préconçus sur des contrôleurs ESP. Les installations traditionnelles de logiciels dédiés pour la mise en place d'hôtes de développement ne sont plus nécessaires ; à la place, on profite de l'accessibilité d'une plateforme web : l'interface web série des navigateurs Chrome, Edge et Opera permet de communiquer avec la carte de développement, de programmer et d'accéder à la console série depuis le navigateur.

Voyons comment lancer facilement un micrologiciel avec ESP-Launchpad.

ESP-Launchpad est un outil web qui utilise les dernières fonctions du navigateur pour flasher facilement des micrologiciels pré-conçus sur des contrôleurs ESP.

Préparation du micrologiciel

Une fois que votre firmware est prêt, l'étape suivante consiste à créer des builds pour tous les systèmes cibles que vous souhaitez prendre en charge et à les convertir en fichiers binaires individuels. Ne vous inquiétez pas, esptool.py (l'utilitaire Python principal d'Espressif pour tout ce qui concerne le flashage) offre une option pratique `merge_bin` qui vous permet de créer facilement un tel fichier binaire. Vous trouverez de plus amples informations à ce sujet sur notre page de documentation [1]. Ces différents fichiers binaires doivent être téléchargés vers une URL accessible au public, que nous utiliserons plus tard.

Remarque : ce tutoriel n'aborde pas en détail la manière d'y parvenir.

Ce qu'il faut savoir : le secret derrière ESP-Launchpad. est un portage JavaScript d'`esptool`, appelé `esptool-js`, qui utilise en interne l'API WebSerial.

Comprendre les bases : le pouvoir de TOML

Avant de nous plonger dans la pratique, nous devons nous familiariser avec la structure TOML de la configuration d'ESP-Launchpad, un aspect fondamental d'ESP-Launchpad. Les fichiers TOML servent ici de simples modèles de configuration qui décrivent en détail les composants de votre micrologiciel, le matériel pris en charge et les applications complémentaires.

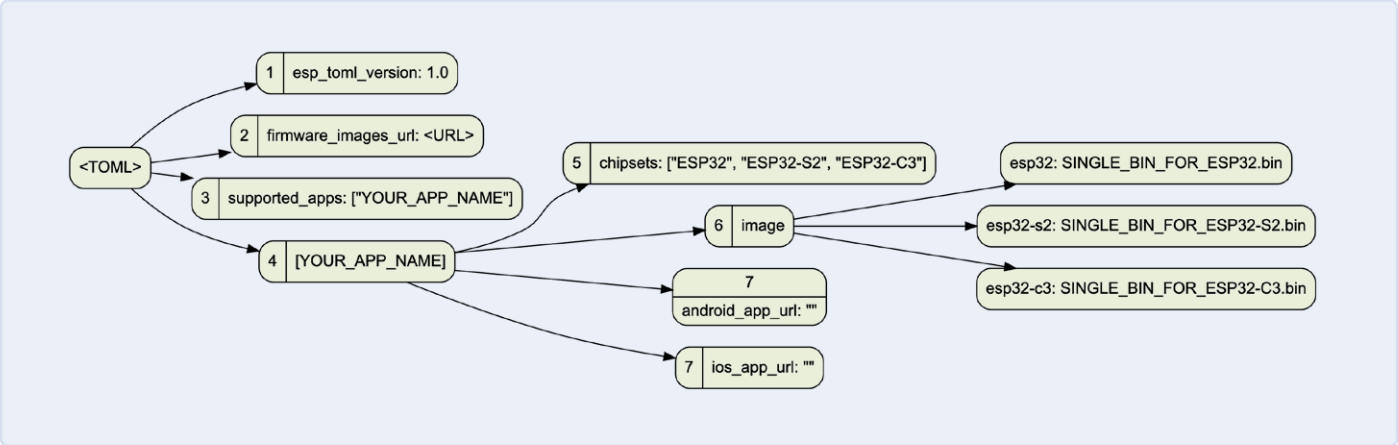

Figure 1. Diagramme en arbre de la structure TOML.

Voici un exemple :

```
esp_toml_version = 1.0
firmware_images_url = "<URL to your firmware images
directory>"
supported_apps = ["YOUR_APP_NAME"]

[YOUR_APP_NAME]
chipsets = ["ESP32", "ESP32-S2", "ESP32-C3"]
image.esp32 = "SINGLE_BIN_FOR_ESP32.bin"
image.esp32-s2 = "SINGLE_BIN_FOR_ESP32-S2.bin"
image.esp32-c3 = "SINGLE_BIN_FOR_ESP32-C3.bin"
android_app_url = ""
ios_app_url = ""
```

Pour mieux comprendre, jetez un coup d'œil à la **figure 1**.

- `esp_toml_version` indique la version du schéma TOML.
- `firmware_images_url` est une URL publique du serveur de fichiers sur lequel les fichiers binaires de votre firmware sont disponibles au téléchargement. Conseil de pro : nous hébergeons aussi bien nos fichiers TOML que nos fichiers binaires sur GitHub, en utilisant GitHub Pages !
- `supported_apps` est un tableau contenant la liste des applications supportées et pour lesquelles les fichiers binaires sont disponibles. Vous pouvez avoir plusieurs applications, qui apparaissent dans la liste déroulante des applications disponibles sur l'interface utilisateur d'ESP-Launchpad. Les trois valeurs que nous venons d'examiner étaient les paires clé-valeur du niveau supérieur.
- `[YOUR_APP_NAME]` — Il s'agit d'un tableau qui est pratiquement une collection de paires clé-valeur. Il contient
  - `chipsets` — Un tableau avec une liste de puces ESP pour lesquels vous allez fournir un micrologiciel prêt à l'emploi. Notez la convention de nommage ici, tout en majuscules, ESP32-C3.
  - `image.<chip-name>` — Ce sont des clés qui relient le nom de l'image binaire à chacune des destinations prises en charge. Celle-ci est ajoutée à `firmware_images_url` afin d'obtenir l'URL finale pour le fichier binaire. Respectez ici aussi la convention de nommage : tout en minuscules, esp32-c3.
  - `ios_app_url` et `android_app_url` sont des paires clé-valeu optionnelles mais spéciales sous la même table d'application, qui vous permettent d'afficher des liens sous forme de codes QR que les utilisateurs peuvent facilement scanner une fois que votre application a été flashée avec succès.

Voyons à quoi ressemble ESP-Launchpad (**figure 2**) lorsque nous fournissons l'exemple de TOML que nous venons de créer avec l'URL imaginaire suivante :
https://espressif.github.io/esp-launchpad/?flashConfigURL=https://some-nice-url/my_app.toml

OK, ça a l'air bien ! Et ensuite ?

Seulement trois étapes simples pour l'utilisateur !

- Connecter l'appareil : connectez l'ESP au port série USB de votre ordinateur.
- Effectuer la connexion : établissez une connexion avec l'appareil dans le menu de l'outil.
- Sélectionner et flasher : sélectionnez et flashez le firmware préconçu.

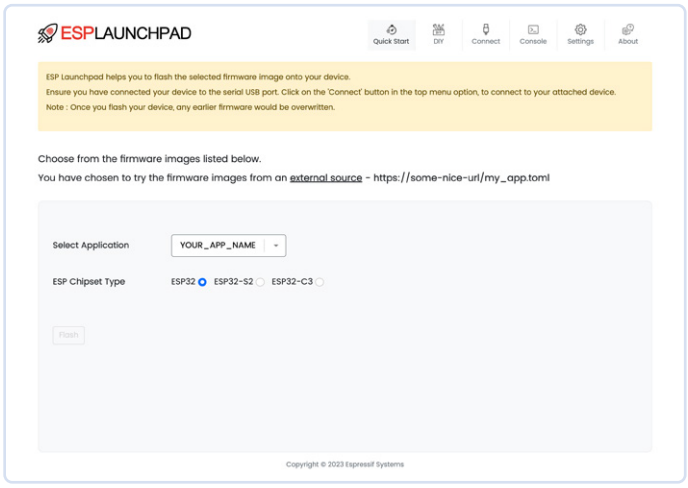

Figure 2. ESP-Launchpad : exemple de configuration chargé.

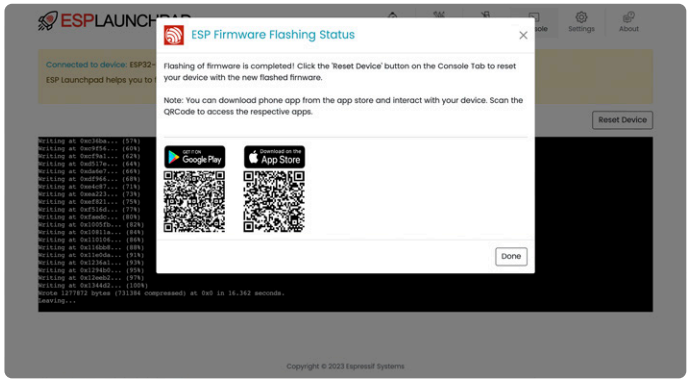

Figure 3. Écran final d'ESP-Launchpad

Après le flashage, l'utilisateur est accueilli par une console qui affiche le journal de l'appareil et un popup avec les codes QR des applications mobiles, si vous les avez ajoutées au TOML (**figure 3**).
Félicitations, vous venez de publier votre application avec ESP-Launchpad ! Conseil de pro : nous avons créé un badge (**figure 4**) que vous pouvez ajouter au fichier README ou sur la page web de votre projet et qui contient un lien hypertexte vers l'URL de configuration Flash de votre application ! Pour plus d'informations, consultez le fichier *README.md* du projet [2].

ESP-Launchpad : le firmware rencontre la communauté

La particularité d'ESP-Launchpad est qu'il offre aux utilisateurs la possibilité de mettre leurs applications de micrologiciel à la disposition d'autres personnes. Dans un simple fichier de configuration, les développeurs peuvent définir d'où proviennent les fichiers binaires prédéfinis de leur micrologiciel, quel matériel est pris en charge et même créer un lien vers des applications complémentaires. Cette approche globale garantit que le micrologiciel n'est pas seulement partagé en tant que fichier binaire, mais qu'il offre aussi une expérience visée par le développeur.

LIENS

[1] Commandes de base - fichiers binaires à flasher :
merge_bin : https://tinyurl.com/esp32mergebinaries
[2] Dépôt GitHub du projet :
https://github.com/espressif/esp-launchpad

Considérations

Le véritable potentiel de tout outil se révèle lorsqu'il est mis en pratique par les utilisateurs. Explorez ESP-Launchpad, intégrez-le dans vos processus de travail et partagez vos expériences ! Vos connaissances et vos contributions sont d'une valeur inestimable pour améliorer encore notre offre. Ensemble, redéfinissons l'expérience de l'évaluation open-source.

230596-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (dhaval.gujar@espressif.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Dhaval Gujar est ingénieur chez Espressif Systems, spécialisé dans les systèmes embarqués. Ce qui le motive, c'est la convergence dynamique de technologies telles que le cloud, l'IdO et plus encore. Dhaval est passionné par le paysage technologique en évolution et par les changements technologiques modernes.

Produits

- **ESP32-DevKitC-32E (SKU 20518)**
  www.elektor.fr/20518
- **Carte de développement LILYGO T-Display-S3 ESP32-S3 (avec connecteurs) (SKU 20299)**
  www.elektor.fr/20299