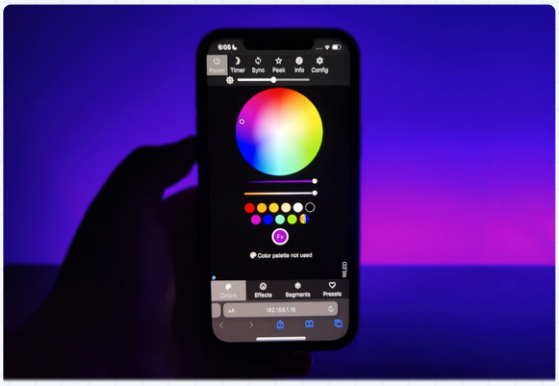
Bibliothèques ESP recommandées

Saad Imtiaz et Jean-François Simon (Elektor)

Vous cherchez des bibliothèques liées aux ESP ? Consultez ces recommandations de l'équipe d'ingénieurs d'Elektor.

WLED (github.com/Aircoookie)

WLED, de Christian Schwinne, est une implémentation rapide et riche en fonctionnalités d'un serveur web ESP8266/ESP32 pour contrôler les LED NeoPixels (WS2812B, WS2811, SK6812). Elle supporte également les puces SPI comme le WS2801, l'APA102 et bien d'autres. La bibliothèque propose plus de 100 effets spéciaux pour les NeoPixels, 50 palettes, des effets de bruit FastLED, et bien plus encore ! Elle dispose d'une interface utilisateur moderne avec des contrôles avancés. La configuration s'effectue via le réseau. Parmi les avantages notables, cette bibliothèque prend en charge jusqu'à 10 sorties LED par instance et peut fonctionner avec des bandes RGBW. Jusqu'à 250 préréglages utilisateur peuvent être utilisés pour sauvegarder et charger facilement des couleurs/effets, et permettent de les faire défiler. Les préréglages peuvent également être utilisés pour exécuter automatiquement les appels API. Il existe également une fonction « veilleuse », qui réduit progressivement l'intensité lumineuse des LED. En ce qui concerne les mises à jour, *Over the Air* (HTTP + ArduinoOTA) est pris en charge et peut être protégé par un mot de passe. Si vous cherchez à contrôler vos lumières RGB ou à donner un nouveau thème à votre pièce, vous allez adorer celle-ci ! Voir aussi Adafruit_NeoPixel ci-dessous.
https://github.com/Aircoookie/WLED

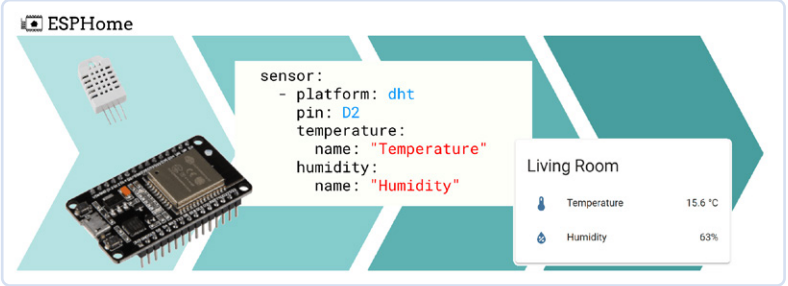


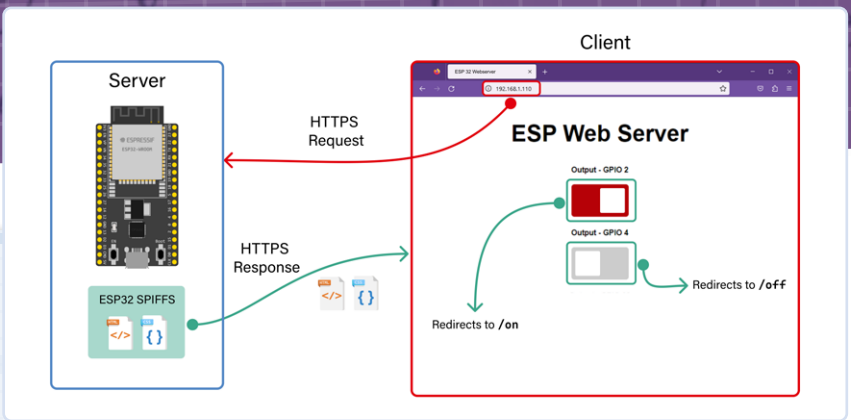
ExpressLRS (github.com/ExpressLRS)

ExpressLRS est une liaison radio open-source pour les applications RC (radiocommande). Elle est conçue pour offrir d'excellentes performances en utilisant la puce LoRa SX127x/SX1280 combinée à un microcontrôleur Espressif ou STM32. ExpressLRS permet aux utilisateurs de bénéficier d'une bonne portée et d'une très faible latence, grâce à la modulation LoRa et à une taille de paquets réduite. ExpressLRS prend en charge le matériel de nombreux fabricants : AxisFlying, BETAFPV, Flywoo, FrSky, HappyModel, HiYounger, HGLRC, ImmersionRC, iFlight, JHEMCU, Jumper, Matek, NamimnoRC, QuadKopters et SIYI. Il offre un débit de paquets de 1 kHz, la télémétrie, des mises à jour WiFi, deux fréquences pour la liaison RC (2,4 GHz ou 900 MHz) et bien d'autres choses encore. C'est sans aucun doute très intéressant pour de nombreux projets RC, avec différents matériels adaptés pour différents besoins.
https://github.com/ExpressLRS/ExpressLRS

ESPHome (github.com/esphome)

ESPHome est un système de développement open-source qui simplifie le processus de configuration et de gestion des appareils basés sur ESP8266 et ESP32. Il permet aux utilisateurs de créer des microprogrammes personnalisés pour ces appareils sans exiger de programmation avancée. Avec ESPHome, vous pouvez définir les fonctionnalités et les paramètres des appareils à l'aide d'un fichier de configuration YAML facile à utiliser, ce qui le rend accessible aux débutants comme aux développeurs expérimentés. Les principales caractéristiques d'ESPHome sont la prise en charge d'un large éventail de capteurs et de composants, la détection et l'intégration automatique avec les plateformes domotiques les plus répandues comme Home Assistant, et les mises à jour OTA pour des mises à niveau transparentes du micrologiciel. Il favorise la création de solutions domestiques intelligentes, permettant aux utilisateurs d'adapter leurs appareils à des besoins spécifiques, tels que la surveillance de la température, le contrôle de l'éclairage ou la sécurité à domicile. ESPHome a gagné en popularité dans la communauté de la maison intelligente. Ne manquez pas de le découvrir, si vous ne le connaissez pas déjà !
https://github.com/esphome/esphome



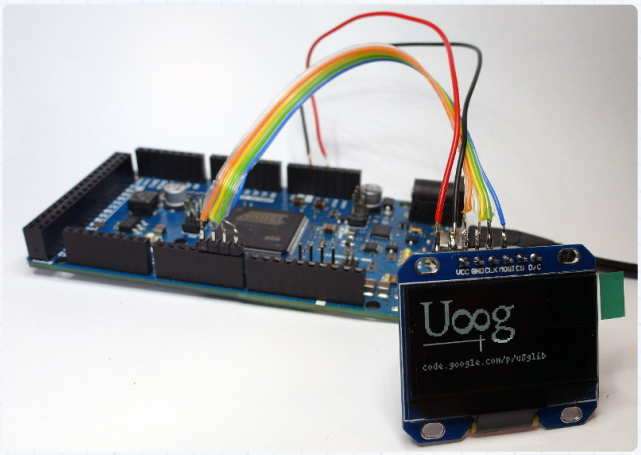


ESPAsyncWebServer (github.com/me-no-dev)

ESPAsyncWebServer est un serveur HTTP et WebSocket asynchrone pour ESP8266 utilisé dans l'environnement Arduino. Il peut être utilisé avec PlatformIO et l'EDI Arduino. L'utilisation d'un réseau asynchrone signifie que vous pouvez gérer plus d'une connexion en même temps. Un appel (*call*) est émis dès que la requête est prête et analysée. Pour envoyer la réponse, vous êtes immédiatement prêt à gérer d'autres connexions pendant que le serveur se charge d'envoyer la réponse en arrière-plan. La vitesse de traitement est très élevée ! ESPAsyncWebserver fournit une API facile à utiliser et est compatible avec les authentifications HTTP Basic et Digest MD5 (par défaut), ainsi qu'avec Chunked Response. Il existe plusieurs plugins offrant divers avantages, tels que : différents emplacements, envoi d'événements au navigateur, réécriture d'URL avancée, etc...
https://github.com/me-no-dev/ESPAsyncWebServer

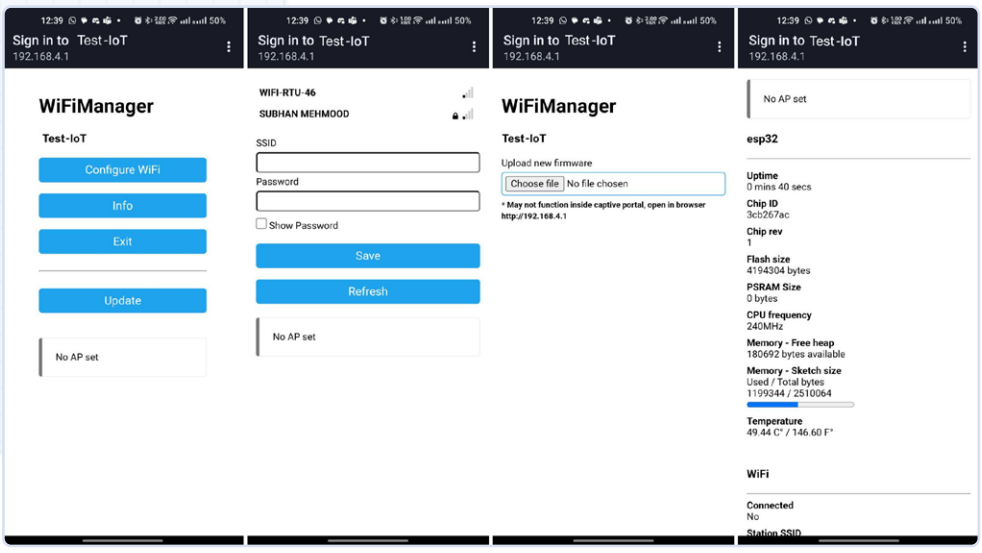
U8glib (github.com/olikraus)

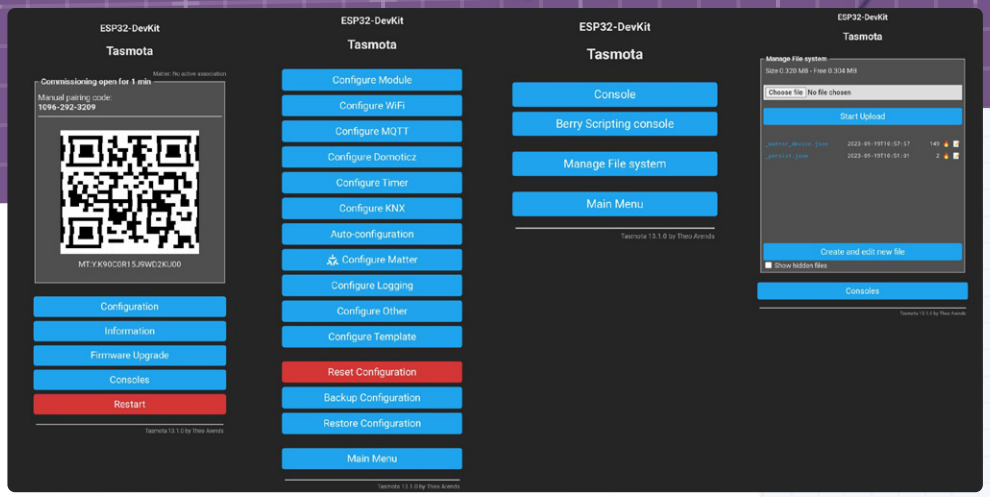
U8glib est une bibliothèque graphique qui prend en charge de nombreux écrans monochromes. La dernière version de *U8glib* pour Arduino est disponible dans le gestionnaire de bibliothèques et peut également être téléchargée depuis GitHub. Elle est compatible avec Arduino (ATmega et ARM), AVR, ARM (avec exemple pour LPC1114) et beaucoup d'autres modules tels que SSD1325, ST7565, ST7920, UC1608, UC1610, UC1701, PCD8544, PCF8812, KS0108 et plus encore. Cette bibliothèque prend en charge de nombreuses polices (monospaces et proportionnelles), le mode curseur de la souris, le mode paysage et portrait... En résumé, une bibliothèque très complète qui sera utile pour de nombreux projets
https://github.com/olikraus/u8glib



WifiManager (github.com/tzapu)

WiFiManager est un gestionnaire de connexion pour les projets basés sur ESP. Il permet à l'utilisateur de configurer les identifiants WiFi et les paramètres personnalisés au moment de l'exécution, grâce à un portail captif. Comment cela fonctionne-t-il ? Lorsque votre ESP démarre, il se met en mode *Station* et tente de se connecter à un point d'accès précédemment enregistré. En cas d'échec, le micrologiciel fait passer l'ESP en mode *Point d'accès* et lance un DNS et un serveur Web. Ensuite, à l'aide de n'importe quel appareil WiFi doté d'un navigateur (ordinateur, téléphone, tablette), vous pouvez vous connecter au point d'accès nouvellement créé et configurer les informations d'identification. L'ESP se connectera alors au réseau de votre choix !
Il existe également des options permettant de changer cette manière de faire, ou de lancer manuellement le portail de configuration ainsi que le portail web indépendamment l'un de l'autre. Il est également possible de les faire fonctionner en mode *non bloquant*.
https://github.com/tzapu/WiFiManager





Tasmota (github.com/arendst)

Tasmota est un micrologiciel open-source conçu pour les appareils ESP, spécifiquement adaptés à la domotique et aux maisons intelligentes. Il prend en charge les microcontrôleurs ESP8266, ESP32, ESP32-S ou ESP32-C3. Theo Arends a initié ce projet en 2016 sous le nom de *Sonoff-MQTT-OTA*. Son objectif principal était d'équiper les appareils basés sur l'ESP8266 produits par l'ITEAD (Sonoff) en les rendant compatibles avec MQTT et les mises à jour *over-the-air* (OTA). Cela avait commencé comme une solution simple, pour modifier un Sonoff Basic (qui, à l'origine, requiert l'usage du cloud) en un appareil gérable localement. Le projet a fini par devenir un écosystème complet adapté à presque tous les apareils basés sur l'ESP. Tasmota est développé pour PlatformIO, ce qui facilite la configuration par le biais d'une interface web (webUI). Les mises à jour peuvent être effectuées par WIFI. Les utilisateurs peuvent automatiser les appareils à l'aide de minuteries ou de règles personnalisées. Le contrôle local complet et l'extensibilité sont possibles grâce aux protocoles MQTT, HTTP, série ou KNX
https://github.com/arendst/Tasmota

Esp32FOTA (github.com/chrisjoyce911)

esp32FOTA est une bibliothèque simple pour ajouter la compatibilité avec les mises à jour *over-the-air* (OTA) à votre projet ESP32. Cette bibliothèque va essayer d'accéder à un fichier JSON hébergé sur un serveur web, analyser son contenu pour déterminer si une nouvelle version du firmware est disponible, et si c'est le cas, la télécharger et l'installer. Pour que ce processus de mise à jour fonctionne, vous avez besoin d'un serveur web avec un fichier JSON valide (compressé en option avec zlib ou gzip). La liste complète des conditions requises (y compris les détails concernant HTTPS) est disponible sur GitHub. Cette bibliothèque est assez complète et inclut la prise en charge des firmwares compressés, des mises à jour de partitions SPIFFS/LittleFS, la compatibilité avec divers systèmes de fichiers pour le stockage des certificats et des signatures. En outre, esp32FOTA permet également des mises à jour basées sur le web (avec un serveur web), la synchronisation des firmwares par lots et la possibilité de forcer les mises à jour. Enfin, la bibliothèque dispose également de fonctionnalités avancées telles que la vérification des signatures pour les images de firmwares téléchargées, la vérification des signatures et la prise en charge du versionnage sémantique. En résumé, c'est une bibliothèque qui mérite d'être découverte si vous souhaitez utiliser les mises à jour OTA pour votre prochain projet.
https://github.com/chrisjoyce911/esp32FOTA

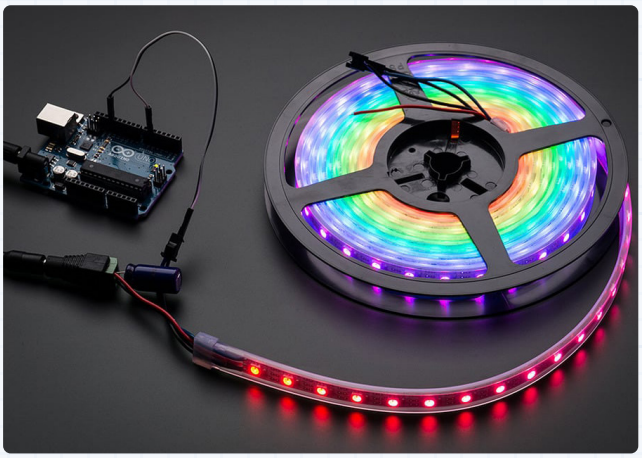
Willow (github.com/toverainc)

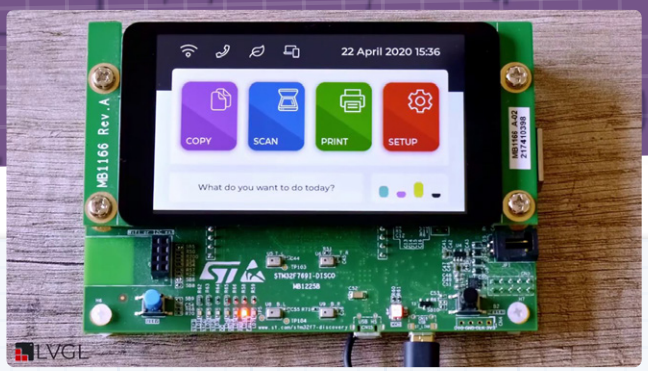
Willow est un un système de développement IdO d'Espressif (*Espressif IDF, IoT Development Framework*) utilisé pour transformer une ESP32-S3-BOX en un assistant vocal polyvalent doté de fonctionnalités diverses. Il peut être déclenché par des mots clés personnalisés (*wake words*) tels que « Hi ESP » ou « Alexa » et écoute les commandes vocales, détectant automatiquement quand commencer et arrêter d'écouter. Willow s'intègre bien aux plateformes domotiques les plus courantes, telles que *Home Assistant*, *openHAB* et les *API REST* génériques. Cet assistant vocal fonctionne parfaitement dans des environnements difficiles, reconnaissant les *wake words* et la parole à des distances d'environ sept m. Il garantit un son de la meilleure qualité possible grâce à des fonctions telles que le contrôle automatique du gain, l'annulation de l'écho, la réduction du bruit et la séparation des sources. Dans les environnements WiFi encombrés, Willow peut utiliser la compression audio pour optimiser l'utilisation du temps de transmission. En outre, Willow offre une reconnaissance vocale sur l'appareil, ce qui vous permet de configurer jusqu'à 400 commandes localement. Vous pouvez également utiliser le serveur d'inférence open-source Willow pour obtenir des capacités de transcription vocale plus étendues. En conclusion, c'est un projet très impressionnant !
https://github.com/toverainc/willow

Adafruit_NeoPixel (github.com/adafruit)

Il s'agit d'une bibliothèque Arduino pour contrôler les NeoPixels. Ce sont, dans le jargon d'Adafruit, des pixels et des bandes de LED RVB adressables individuellement, basés sur les WS2812, WS2811 et SK6812. Ces LED ont chacune un driver intégré et utilisent un protocole de contrôle à un seul fil. Elles ont tendance à être un peu difficiles à utiliser car les exigences de synchronisation sont strictes et à cause du protocole spécifique. Avec cette bibliothèque, tout devient plus faciles ! Adafruit la développe et la maintient gratuitement. En retour, les utilisateurs peuvent décider d'acheter certains de leurs produits. Les principaux objectifs d'Adafruit_NeoPixel sont d'être facile à utiliser et d'être flexible en termes de chipsets supportés. La bibliothèque supporte les AVR (ATmega et ATtiny), Teensy, Arduino Due, Arduino 101, ATSAMD21/51, Adafruit STM32 Feather, ESP8266, ESP32, Nordic nRF51/52 ainsi que certains ICs de la série XMC d'Infineon. Entre autres, les nombreuses fonctions bien connues telles que `setBrightness()`, `setPixelColor()` et 12 autres rendent cette bibliothèque très utile pour un prototypage rapide.

https://github.com/adafruit/Adafruit_NeoPixel





LVGL (github.com/lvgl)

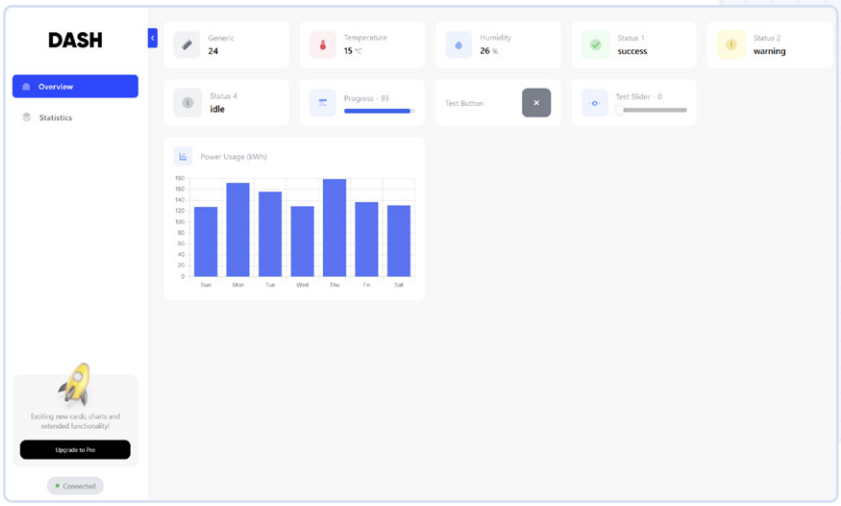
Toujours sur le sujet des tableaux de bord et des graphiques, LVGL (**figure 12**) est une bibliothèque graphique embarquée, permettant la création d'interfaces graphiques pour une large gamme de microcontrôleurs, y compris, bien sûr, ESP32 et Arduino (la liste complète est sur GitHub). Cette bibliothèque est écrite en C et est portable, c'est-à-dire qu'elle ne dépend d'aucune dépendance externe. Elle peut être utilisée avec ou sans système d'exploitation en temps réel (RTOS) et supporte un grand nombre d'afficheurs : monochrome, ePaper, OLED, etc. Elle nécessite 32 kB de RAM et 128 kB de mémoire Flash. LVGL dispose d'un large éventail de widgets intégrés, de styles, de présentations, et plus encore ; elle est très personnalisable. Animations, anticrénelage, contrôle de l'opacité, défilement fluide, ombres, transformation d'images… La liste est longue. Diverses méthodes de saisie comme la souris, le pavé tactile et le clavier sont prises en charge. Make et CMake sont tous deux supportés, ce qui vous permet de développer sur un PC et d'utiliser le même code d'interface utilisateur sur du matériel embarqué, ce qui peut être une fonctionnalité intéressante pour certains utilisateurs.

https://github.com/lvgl/lvgl

ESP-DASH (github.com/ayushsharma82)

ESP-DASH est une bibliothèque à haute vitesse conçue pour développer un tableau de bord fonctionnel et en temps réel adapté aux microcontrôleurs ESP8266 et ESP32. Elle est développée par l'utilisateur GitHub ayushsharma82 et d'autres contributeurs, et en est actuellement à la version 4. Cette bibliothèque comporte de nombreuses fonctionnalités, notamment des graphiques, des cartes d'affichage, des boutons interactifs et de nombreux autres composants, permettant de créer de beaux tableaux de bord. Ceux-ci seront accessibles localement et ne nécessiteront pas de connexion internet, toutes les données étant stockées sur la puce. ESP-DASH propose de générer automatiquement des pages web et de les mettre à jour en temps réel pour tous les clients connectés. Les utilisateurs n'ont pas besoin de se plonger dans le HTML, le CSS ou le JavaScript, car elle offre une interface C++ facile à utiliser. Elle fournit des composants préconfigurés pour la gestion de vos données et offre une grande souplesse, car vous pouvez ajouter ou supprimer des composants sans effort, directement à partir de la page web. De plus, elle est dotée d'une prise en charge intégrée des graphiques, ce qui améliore sa fonctionnalité.

https://github.com/ayushsharma82/ESP-DASH



VF : Laurent Rauber — 230588-04