

Rédaction invitée
pour ce numéro



ESPRESSIF

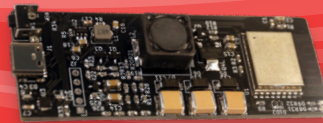
Édition bonus
déclassifiée !

domotique avec les puces Espressif

ADF, IDF et autres SDK



Try it with
ESP Launchpad



p. 4 un guide pour
utiliser **ChatGPT**
avec les SoC
Espressif

ESP32 et ChatGPT

p. 14 clé d'authentification
à deux facteurs
basée sur l'ESP32-C3



Articles pour les
professionnels,
les électroniciens
et les étudiants !



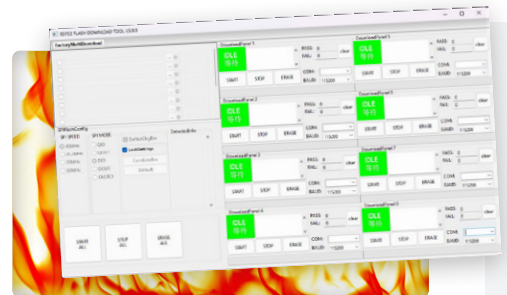
dekatron basé
sur l'ESP32-C3

p. 18



interview de Paulus Schoutsen,
le créateur de Home
Assistant

p. 24



programmez votre ESP32
de manière efficace et
sécurisée

p. 30



Design-In Expertise & Service

YOUR PARTNER FOR



ESPRESSIF



Leading Espressif franchise distribution partner



36 years of experience in the semiconductor industry



Inetek is always up to date regarding Espressif's innovations – Contact us!



Features are missing? We address your requests for next product generations



Dedicated and focused in-house support offering



Espressif & Inetek's FAE team are in close & regular contact to speed up your designs



We inform you on Espressif's products & wireless trends in trainings & webinars



Popular Espressif SoCs, modules and EVKs available from stock

CONTACT US FOR PARTS & SUPPORT



www.inetek.com

Inetek is the worldwide independent distributor with a **Passion for Innovation** and a **Commitment to Service**.

Founded in 1987, Inetek gained the trust of thousands industrial customers as a technical semiconductor and design-in service provider. We work with your team to ensure our mutual success by providing the highest level of technical and commercial services for your projects.

Start your career with us!

Apply now at personal@inetek.com



- Field Sales Engineer (m/f/x)
- Field Application Engineer (m/f/x)
- Line Management / Marketing (m/f/x)



Follow us!

INELTEK GmbH

Heidenheim, Wien, Castelfranco, London
Hamburg, München, Frankfurt, Dresden



Toute reproduction ou représentation intégrale ou partielle, par quelque procédé que ce soit, des pages publiées dans la présente publication, faite sans l'autorisation de l'éditeur est illicite et constitue une contrefaçon. Seules sont autorisées, d'une part, les reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective, et, d'autre part, les analyses et courtes citations justifiées par le caractère scientifique ou d'information de l'oeuvre dans laquelle elles sont incorporées (Loi du 11 mars 1957 - art. 40 et 41 et Code Pénal art. 425).

Certains circuits, dispositifs, composants, etc. décrits dans cette revue peuvent bénéficier de droits propres aux brevets; la Société éditrice n'accepte aucune responsabilité du fait de l'absence de mention à ce sujet. Conformément à l'art. 30 de la Loi sur les Brevets, les circuits et schémas publiés dans Elektor ne peuvent être réalisés que dans des buts privés ou scientifiques et non commerciaux. L'utilisation des schémas n'implique aucune responsabilité de la part de la Société éditrice. La Société éditrice n'est pas tenue de renvoyer des articles qui lui parviennent sans demande de sa part et qu'elle n'accepte pas pour publication. Si la Société éditrice accepte pour publication un article qui lui est envoyé, elle est en droit de l'amender et/ou de le faire amender à ses frais; la Société éditrice est de même en droit de traduire et/ou de faire traduire un article et de l'utiliser pour ses autres éditions et activités, contre la rémunération en usage chez elle.

Imprimé aux Pays-Bas par Senefelder Misset,
Mercuriusstraat 35, 7006 RK Doetinchem

Distribué en France par M.L.P. et en Belgique
par A.M.P.

Rédacteur en chef
Jens Nickel

Content Director
C. J. Abate

Publisher
Erik Jansen



Édition bonus déclassifiée !

Au cours des derniers mois, l'équipe éditoriale d'Elektor a travaillé en étroite collaboration avec Espressif pour rédiger des articles de fond sur une variété de sujets passionnants. Tout ce travail a abouti à une édition du magazine Elektor rédigée en collaboration avec Espressif, que nous avons publiée au début du mois de décembre 2023. Mais notre coopération se poursuit. Cette édition bonus du magazine Elektor est remplie de projets et de tutoriels qui vous inspireront pendant les mois à venir.

Comme le veut la tradition chez Elektor, cette édition bonus a quelque chose pour tous, des électroniciens professionnels intéressés par le

développement de produits IoT aux maker à la recherche de projets créatifs pour le week-end. Elektor et Espressif proposent des conseils sur l'utilisation de ChatGPT avec les SoC Espressif, un projet Dekatron amusant, des conseils pour la création d'applications IdO sans expertise

logicielle, et bien plus encore. Bonne lecture ! Lorsque vous commencerez votre prochain projet Espressif, n'hésitez pas à partager votre expérience avec notre communauté sur la plateforme en ligne Elektor Labs à l'adresse elektormagazine.fr/labs - nous avons hâte de découvrir vos créations !

C. J. Abate (Directeur du contenu, Elektor)



DANS CE NUMÉRO

4 libérer la puissance d'OpenAI et d'ESP-BOX

un guide pour utiliser ChatGPT avec les SoC Espressif



14 ESP-Unlock

clé d'authentification à deux facteurs basée sur l'ESP32-C3



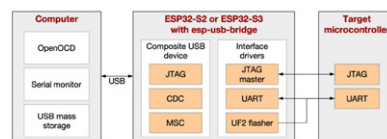
18 dekatron

une époque historique renaît !



24 la révolution de la maison intelligente

Paulus Schoutsen parle de Home Assistant, d'ESPHome, et d'autres choses encore

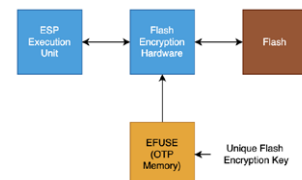


30 programmer le micrologiciel !

flashez votre ESP32

36 des fusées jusqu'aux violoncelles

applications pratiques et réflexions lors de la création de solutions sans fil



40 fabrication de dispositifs IdO sécurisés

Pourquoi ? Comment ?

44 une vie plus confortable et plus facile

46 comment construire des applications IdO sans expertise logicielle

48 un distributeur à valeur ajoutée de solutions IdO et plus encore

50 développement IoT rapide et facile avec M5Stack

52 concevoir une interface graphique sur ESP32

54 mettez la main sur le nouveau matériel Espressif



OpenAI

libérer la puissance d'OpenAI et d'ESP-BOX

un guide pour combiner ChatGPT avec les SoC Espressif

Ali Hassan Shah (Espressif)

Explorons le potentiel du ChatGPT avec ESP-BOX, une plateforme de développement dotée de cartes équipées de microphones et d'un vaste environnement logiciel pour la reconnaissance vocale et bien plus encore. Il s'agit d'une combinaison puissante qui peut faire passer les appareils IdO au prochain niveau !

Le monde est témoin d'une révolution technologique, et OpenAI est à l'avant-garde de ce changement. L'une de ses innovations les plus intéressantes est ChatGPT, qui utilise le traitement du langage naturel pour créer des expériences utilisateur plus attrayantes et plus intuitives. L'intégration des API d'OpenAI aux appareils IdO a ouvert un large éventail de possibilités.

Cet article contient trois parties principales, chacune couvrant des aspects essentiels du projet. La première section présente la plateforme de développement ESP-BOX, en détaillant ses caractéristiques et ses fonctions. La deuxième section est une étude de cas qui décrit les étapes de la réalisation du projet à partir de zéro. La dernière section fournit une liste de sources d'information supplémentaires que les lecteurs peuvent explorer pour approfondir leurs connaissances et leur compréhension du projet.

ESP-BOX

L'ESP-BOX [1] est une plateforme de développement AIoT de nouvelle génération qui intègre les cartes de développement ESP32-S3-BOX et ESP32-S3-BOX-Lite. Ces cartes sont basées sur le SoC ESP32-S3 [2] Wi-Fi + Bluetooth 5 (LE) et fournissent une solution flexible et personnalisable pour le développement d'applications AIoT compatibles avec divers capteurs, contrôleurs et passerelles.



Figure 1. ESP32-S3-BOX.

L'ESP-BOX (figure 1) propose plusieurs fonctions qui en font une plateforme de développement idéale pour l'AIoT. Examinons de plus près quelques-unes de ses principales caractéristiques.

- > **Interaction vocale en champ lointain avec 2 microphones :** L'ESP-BOX prend en charge l'interaction vocale en champ lointain grâce à deux microphones, ce qui permet aux utilisateurs d'interagir avec leurs appareils à distance.
- > **Reconnaissance hors ligne des commandes vocales en chinois et en anglais avec un taux de reconnaissance élevé :** L'ESP-BOX offre une reconnaissance hors ligne des commandes vocales dans les langues chinoise et anglaise avec un taux de reconnaissance élevé, ce qui facilite la conception d'appareils à commande vocale.
- > **Plus de 200 commandes vocales reconfigurables en chinois et en anglais :** les développeurs peuvent facilement reconfigurer plus de 200 commandes vocales en chinois et en anglais en fonction de leurs besoins.
- > **Identification continue et interruption de veille :** L'ESP-BOX prend en charge l'identification continue et l'interruption de veille, ce qui garantit que les appareils sont toujours prêts à recevoir des commandes vocales.
- > **Cadre d'interface graphique polyvalent et réutilisable :** L'ESP-BOX est livré avec un cadre d'interface graphique flexible et réutilisable, permettant aux développeurs de créer des interfaces utilisateur personnalisées pour leurs applications.



Figure 2. Processus de l'exemple.

- > **Cadre de développement AIoT de bout en bout ESP-RainMaker :** L'ESP-BOX repose sur le cadre de développement AIoT/IoT d'Espressif, ESP-RainMaker, qui fournit aux développeurs tous les outils dont ils ont besoin pour créer des appareils puissants et intelligents.
- > **Les connecteurs compatibles Pmod permettent l'extension des modules périphériques :** L'ESP-BOX dispose de connecteurs compatibles Pmod, ce qui permet d'étendre facilement ses capacités avec une large gamme de modules périphériques.

Étude de cas

Cette étude de cas décrit le processus de développement d'un chatbot à commande vocale qui utilise ESP-BOX et l'API OpenAI. Le système est capable de recevoir des commandes vocales de la part des utilisateurs, de les afficher à l'écran et de les traiter via les API OpenAI pour générer une réponse. La réponse est ensuite affichée à l'écran et diffusée par l'ESP-BOX. Le flux de travail étape par étape explique en détail comment intégrer ces technologies pour créer un chatbot à commande vocale performant et efficace (voir figure 2 et figure 3).

Configuration de l'environnement

La configuration d'un environnement adapté et l'installation de la bonne version sont essentielles pour éviter les erreurs. Dans cette démonstration, nous utiliserons la version 5.0 d'ESP-IDF (branche principale). Si vous avez besoin d'aide pour configurer ESP-IDF, veuillez vous référer au guide de programmation d'IDF [3] pour plus d'informations. Au moment de la rédaction de cet article, le *commit head* de l'IDF est *df9310ada2*.

Pour utiliser ChatGPT, un puissant modèle de langage basé sur l'architecture GPT-3.5, vous devez d'abord obtenir une clé API secrète. Pour cela, il faut créer un compte sur la plateforme OpenAI [4] et obtenir des tokens par création ou par achat. Avec une clé API, vous accédez à un large éventail de fonctions et de capacités personnalisables pour

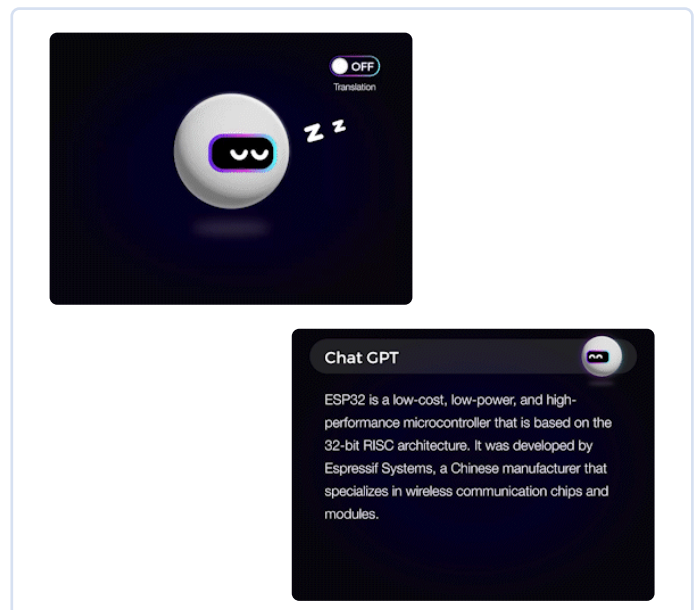


Figure 3. Démo d'ESP-BOX comme chatbot.

répondre à vos besoins, telles que le traitement et la génération de langage naturel, la complétion de texte et la personnalisation de la conversation. Consultez le lien de référence officiel de l'API [5]. Il va sans dire que le maintien de la confidentialité et de la sécurité de la clé API est essentiel pour empêcher tout accès non autorisé au compte et aux données de l'utilisateur.

Ajout de la reconnaissance vocale hors ligne

Espressif Systems a développé un cadre de reconnaissance vocale innovant appelé ESP-SR [6]. Ce cadre est conçu pour permettre aux appareils de reconnaître les mots et les phrases prononcés sans dépendre de services externes basés sur le cloud, ce qui en fait une solution idéale pour les applications de reconnaissance vocale hors ligne.

Le cadre ESP-SR se compose de plusieurs modules, notamment l'Audio Front-end (AFE), le Wake Word Engine (WakeNet), le Speech Command Word Recognition (MultiNet) et le Speech Synthesis (qui ne prend actuellement en charge que la langue chinoise). Pour plus d'informations, consultez la documentation officielle [7].

Intégration de l'API OpenAI

L'API OpenAI fournit de nombreuses fonctions que les développeurs peuvent exploiter pour améliorer leurs applications. Dans notre projet, nous avons utilisé les API Audio-to-Text et Completion et les avons implémentées avec un code en langage C basé sur ESP-IDF. La partie suivante donne un bref aperçu du code que nous avons utilisé.

Audio to Text

Pour extraire du texte à partir de l'audio, nous utilisons HTTPS et l'interface OpenAI Audio API. Le code suivant est utilisé pour cette tâche.

Ce code correspond à une fonction nommée `create_whisper_from_record()`, qui reçoit un pointeur vers un tampon contenant les données audio et un entier `audio_len` qui représente la longueur des données audio. Cette fonction envoie une requête POST au point d'accès de l'API OpenAI pour transcrire les données audio fournies. La fonction commence par initialiser l'URL de l'API OpenAI et par définir les en-têtes d'autorisation avec le jeton porteur `OPENAI_API_KEY`. Ensuite, un client HTTP est configuré et initialisé avec la configuration fournie, y compris l'URL, la méthode HTTP, le gestionnaire d'événements, la taille de la mémoire tampon, le délai d'attente et le certificat SSL.

Ensuite, le type de contenu et la chaîne de délimitation de la demande multipart form-data sont définis en tant qu'en-têtes pour le client HTTP. Les données du fichier et sa taille sont également définies, et une requête multipart/form-data est construite. La mémoire tampon de `form_data` est allouée par la fonction `malloc()`, et les informations nécessaires y sont ajoutées. Celles-ci comprennent le nom de fichier et le Content-Type du fichier audio, le contenu du fichier et le nom du modèle qui sera utilisé pour la transcription.

Une fois que `form_data` est construit, il est défini comme champ post dans le client HTTP, et le client envoie la requête POST au point de terminaison de l'API OpenAI. En cas d'erreur lors de la requête, la fonction enregistre un message d'erreur. Enfin, le client HTTP est nettoyé et les ressources allouées à `form_data` sont libérées.

La fonction renvoie un code d'erreur `esp_err_t`, qui indique si la requête HTTP a abouti ou non.

Chat Completion

L'API OpenAI Chat Completion [8] est utilisée pour envoyer des requêtes HTTPS pour l'achèvement du chat. Ce processus implique l'utilisation



Listing 1: Extracting Text from Audio.

```
esp_err_t create_whisper_request_from_record(uint8_t *audio, int audio_len)
{
    // Set the authorization headers
    char url[128] = "https://api.openai.com/v1/audio/transcriptions";
    char headers[256];
    snprintf(headers, sizeof(headers), "Bearer %s", OPENAI_API_KEY);
    // Configure the HTTP client
    esp_http_client_config_t config = {
        .url = url,
        .method = HTTP_METHOD_POST,
        .event_handler = response_handler,
        .buffer_size = MAX_HTTP_RECV_BUFFER,
        .timeout_ms = 60000,
        .crt_bundle_attach = esp_crt_bundle_attach,
    };
    // Initialize the HTTP client
    esp_http_client_handle_t client = esp_http_client_init(&config);
```

```

// Set the headers
esp_http_client_set_header(client, "Authorization", headers);
// Set the content type and the boundary string
char boundary[] = "boundary1234567890";
char content_type[64];
snprintf(content_type, sizeof(content_type), "multipart/form-data; boundary=%s", boundary);
esp_http_client_set_header(client, "Content-Type", content_type);
// Set the file data and size
char *file_data = NULL;
size_t file_size;
file_data = (char *)audio;
file_size = audio_len;
// Build the multipart/form-data request
char *form_data = (char *)malloc(MAX_HTTP_RECV_BUFFER);
assert(form_data);
ESP_LOGI(TAG, "Size of form_data buffer: %zu bytes", sizeof(*form_data) * MAX_HTTP_RECV_BUFFER);
int form_data_len = 0;
form_data_len += snprintf(form_data + form_data_len, MAX_HTTP_RECV_BUFFER - form_data_len,
                          "--%s\r\n"
                          "Content-Disposition: form-data; name=\"file\"; filename=\"%s\"\\r\n"
                          "Content-Type: application/octet-stream\r\n"
                          "\\r\n", boundary, get_file_format(file_type));
ESP_LOGI(TAG, "form_data_len %d", form_data_len);
ESP_LOGI(TAG, "form_data %s\\n", form_data);
// Append the audio file contents
memcpy(form_data + form_data_len, file_data, file_size);
form_data_len += file_size;
ESP_LOGI(TAG, "Size of form_data: %zu", form_data_len);
// Append the rest of the form-data
form_data_len += snprintf(form_data + form_data_len, MAX_HTTP_RECV_BUFFER - form_data_len,
                          "\\r\n"
                          "--%s\r\n"
                          "Content-Disposition: form-data; name=\"model\"\\r\n"
                          "\\r\n"
                          "whisper-1\\r\n"
                          "--%s--\\r\n", boundary, boundary);
// Set the headers and post field
esp_http_client_set_post_field(client, form_data, form_data_len);
// Send the request
esp_err_t err = esp_http_client_perform(client);
if (err != ESP_OK) {
    ESP_LOGW(TAG, "HTTP POST request failed: %s\\n", esp_err_to_name(err));
}
// Clean up client
esp_http_client_cleanup(client);
// Return error code
return err;
}

```



L'intégration du ChatGPT d'OpenAI avec l'ESP-BOX d'Espressif a ouvert de nouvelles possibilités pour créer des appareils IdO puissants et intelligents.



Listing 2: HTTPS request for chat completion.

```
esp_err_t create_chatgpt_request(const char *content)
{
    char url[128] = "https://api.openai.com/v1/chat/completions";
    char model[16] = "gpt-3.5-turbo";
    char headers[256];
    snprintf(headers, sizeof(headers), "Bearer %s", OPENAI_API_KEY);
    esp_http_client_config_t config = {
        .url = url,
        .method = HTTP_METHOD_POST,
        .event_handler = response_handler,
        .buffer_size = MAX_HTTP_RECV_BUFFER,
        .timeout_ms = 30000,
        .cert_pem = esp_crt_bundle_attach,
    };
    // Set the headers
    esp_http_client_handle_t client = esp_http_client_init(&config);
    esp_http_client_set_header(client, "Content-Type", "application/json");
    esp_http_client_set_header(client, "Authorization", headers);
    // Create JSON payload with model, max tokens, and content
    snprintf(json_payload, sizeof(json_payload), json_fmt, model, MAX_RESPONSE_TOKEN, content);
    esp_http_client_set_post_field(client, json_payload, strlen(json_payload));
    // Send the request
    esp_err_t err = esp_http_client_perform(client);
    if (err != ESP_OK) {
        ESP_LOGW(TAG, "HTTP POST request failed: %s\n", esp_err_to_name(err));
    }
    // Clean up client
    esp_http_client_cleanup(client);
    // Return error code
    return err;
}
```

de la fonction `create_chatgpt_request()`, qui reçoit un paramètre de contenu représentant le texte d'entrée du modèle GPT-3.5.

La fonction commence par définir l'URL, le modèle et les en-têtes nécessaires à la requête HTTP POST, puis crée une charge utile JSON avec le modèle, les jetons max et le contenu. Ensuite, la fonction définit les en-têtes de la requête HTTP et définit la charge utile JSON comme champ post de la requête. La requête HTTP POST est alors envoyée avec `esp_http_client_perform()`, et si la requête échoue, un message d'erreur est enregistré. Enfin, le client HTTP est nettoyé et le code d'erreur est renvoyé.

Gestion de la réponse

La fonction de rappel `response_handler` est utilisée par la bibliothèque client HTTP ESP-IDF pour gérer les événements qui surviennent lors d'un échange requête/réponse HTTP.

Si l'événement `HTTP_EVENT_ON_DATA` survient, la fonction alloue de la mémoire pour les données entrantes, copie les données dans la mémoire tampon et incrémente la variable `data_len` en conséquence. Cela permet de rassembler les données de la réponse.

Si l'événement `HTTP_EVENT_ON_FINISH` survient, la fonction affiche un message indiquant que l'échange HTTP est terminé, puis appelle la fonction `parsing_data()` pour traiter les données accumulées/

brutes. Elle libère ensuite la mémoire et remet à zéro les variables `data` et `data_len`. Enfin, la fonction renvoie `ESP_OK`, indiquant que l'opération a réussi.

Analyse des données brutes

Le composant JSON `parser` [9] est utilisé pour analyser les réponses brutes obtenues de l'API ChatGPT et l'API Whisper AI via HTTP. Pour effectuer cette tâche, on utilise une fonction qui fait appel au composant d'analyseur syntaxique. De plus amples détails sur cet outil sont disponibles sur GitHub [10].

Intégration de l'API TTS

Pour l'instant, OpenAI n'offre pas d'accès public à son API de synthèse vocale (TTS). Cependant, il existe plusieurs autres API TTS, notamment Voicerss [11], TTSMaker [12] et TalkingGenie [13]. Ces API peuvent générer de la parole à partir de texte, et vous pouvez trouver plus d'informations à leur sujet sur leurs sites web respectifs.

Dans le cadre de ce tutoriel, nous utiliserons l'interface de l'API TalkingGenie, qui est l'une des meilleures options disponibles pour générer une parole de haute qualité et à consonance naturelle à la fois en anglais et en chinois. L'une des caractéristiques uniques de TalkingGenie est sa capacité à traduire des textes en langues mixtes, comme le chinois et l'anglais, en discours de manière transparente. Il s'agit d'un outil précieux pour créer du contenu destiné à un public international.



Listing 3: Handle events during an HTTP request/response exchange.

```
esp_err_t response_handler(esp_http_client_event_t *evt)
{
    static char *data = NULL; // Initialize data to NULL
    static int data_len = 0; // Initialize data to NULL
    switch (evt->event_id) {
    case HTTP_EVENT_ERROR:
        ESP_LOGI(TAG, "HTTP_EVENT_ERROR");
        break;
    case HTTP_EVENT_ON_CONNECTED:
        ESP_LOGI(TAG, "HTTP_EVENT_ON_CONNECTED");
        break;
    case HTTP_EVENT_HEADER_SENT:
        ESP_LOGI(TAG, "HTTP_EVENT_HEADER_SENT");
        break;
    case HTTP_EVENT_ON_HEADER:
        if (evt->data_len) {
            ESP_LOGI(TAG, "HTTP_EVENT_ON_HEADER");
            ESP_LOGI(TAG, "%.s", evt->data_len, (char *)evt->data);
        }
        break;
    case HTTP_EVENT_ON_DATA:
        ESP_LOGI(TAG, "HTTP_EVENT_ON_DATA (%d +)%d\n", data_len, evt->data_len);
        ESP_LOGI(TAG, "Raw Response: data length: (%d +)%d: %.s\n", data_len,
            evt->data_len, evt->data_len, (char *)evt->data);

        // Allocate memory for the incoming data
        data = heap_caps_realloc(data, data_len + evt->data_len + 1,
            MALLOC_CAP_SPIRAM | MALLOC_CAP_8BIT);

        if (data == NULL) {
            ESP_LOGE(TAG, "data realloc failed");
            free(data);
            data = NULL;
            break;
        }
        memcpy(data + data_len, (char *)evt->data, evt->data_len);
        data_len += evt->data_len;
        data[data_len] = '\0';
        break;
    case HTTP_EVENT_ON_FINISH:
        ESP_LOGI(TAG, "HTTP_EVENT_ON_FINISH");
        if (data != NULL) {
            // Process the raw data
            parsing_data(data, strlen(data));
            // Free memory
            free(data);
            data = NULL;
            data_len = 0;
        }
        break;
    case HTTP_EVENT_DISCONNECTED:
        ESP_LOGI(TAG, "HTTP_EVENT_DISCONNECTED");
        break;
    default:
        break;
    }
    return ESP_OK;
}
```



Listing 4: Parsing the raw response obtained from ChatGPT API and Whisper AI API.

```
void parse_response (const char *data, int len)
{
    jparse_ctx_t jctx;
    int ret = json_parse_start(&jctx, data, len);
    if (ret != OS_SUCCESS) {
        ESP_LOGE(TAG, "Parser failed");
        return;
    }
    printf("\n");
    int num_choices;
    /* Parsing Chat GPT response*/
    if (json_obj_get_array(&jctx, "choices", &num_choices) == OS_SUCCESS) {
        for (int i = 0; i < num_choices; i++) {
            if (json_arr_get_object(&jctx, i) == OS_SUCCESS &&
                json_obj_get_object(&jctx, "message") == OS_SUCCESS &&
                json_obj_get_string(&jctx, "content", message_content,
                sizeof(message_content)) == OS_SUCCESS) {
                ESP_LOGI(TAG, "ChatGPT message_content: %s\n", message_content);
            }
            json_arr_leave_object(&jctx);
        }
        json_obj_leave_array(&jctx);
    }
    /* Parsing Whisper AI response*/
    else if (json_obj_get_string(&jctx, "text", message_content,
        sizeof(message_content)) == OS_SUCCESS) {
        ESP_LOGI(TAG, "Whisper message_content: %s\n", message_content);
    } else if (json_obj_get_object(&jctx, "error") == OS_SUCCESS) {
        if (json_obj_get_string(&jctx, "type", message_content,
            sizeof(message_content)) == OS_SUCCESS) {
            ESP_LOGE(TAG, "API returns an error: %s", message_content);
        }
    }
}
}
```

Le code suivant envoie une réponse textuelle générée par ChatGPT à l'API TalkingGenie en utilisant HTTPS, puis lit le discours résultant grâce à un ESP-BOX.

La fonction `text_to_speech()` reçoit une chaîne de message et un paramètre `AUDIO_CODECS_FORMAT`. La chaîne de message est le texte qui sera synthétisé en parole, tandis que le paramètre `AUDIO_CODECS_FORMAT` spécifie si la parole doit être encodée au format MP3 ou WAV. La fonction commence par encoder la chaîne de messages avec la fonction `url_encode()` qui remplace certains caractères non valides par leur code ASCII, puis convertit ce code en une représentation hexadécimale à deux chiffres. Ensuite, elle alloue de la mémoire pour la chaîne codée résultante. Elle vérifie ensuite le paramètre `AUDIO_CODECS_FORMAT` et définit la chaîne de format de codec appropriée à utiliser dans l'`url`.

Ensuite, la fonction détermine la taille du tampon `url` nécessaire pour faire une requête `GET` à l'interface de l'API TalkingGenie, et alloue de la mémoire pour le tampon `url` en conséquence. Elle formate ensuite la chaîne `url` avec les paramètres appropriés, y compris `voiceId` (qui spécifie la voix à utiliser), le texte encodé, la vitesse et le volume de la parole, et le type d'audio (soit MP3 ou WAV).

La fonction met ensuite en place une structure `esp_http_client_config_t` avec l'`url` et d'autres paramètres de configuration, initialise un `esp_http_client_handle_t` avec la structure, et effectue une requête `GET` auprès de l'API TalkingGenie en utilisant `esp_http_client_perform()`. Si la requête est réussie, la fonction renvoie `ESP_OK`, sinon elle renvoie un code d'erreur. Enfin, la fonction libère la mémoire allouée au tampon `url` et au message encodé, nettoie l'`esp_http_client_handle_t` et renvoie le code d'erreur.

Gestion de la réponse TTS

De même, la fonction de rappel `http_event_handler()` est définie pour gérer les événements qui se produisent au cours d'un échange requête/réponse HTTP.

`HTTP_EVENT_ON_DATA` est utilisé pour gérer les données audio reçues du serveur. Les données audio sont stockées dans une mémoire tampon appelée `record_audio_buffer` et la longueur totale des données audio reçues est stockée dans une variable appelée `file_total_len`. Si la longueur totale des données audio reçues est inférieure à la valeur prédéfinie `MAX_FILE_SIZE`, les données sont copiées dans `record_audio_buffer`.



Listing 5: Text to Speech.

```
esp_err_t text_to_speech_request(const char *message, AUDIO_CODECS_FORMAT code_format)
{
    int j = 0;
    size_t message_len = strlen(message);
    char *encoded_message;
    char *language_format_str, *voice_format_str, *codec_format_str;
    // Encode the message for URL transmission
    encoded_message = heap_caps_malloc((3 * message_len + 1), MALLOC_CAP_SPIRAM | MALLOC_CAP_8BIT);
    url_encode(message, encoded_message);
    // Determine the audio codec format
    if (AUDIO_CODECS_MP3 == code_format) {
        codec_format_str = "MP3";
    } else {
        codec_format_str = "WAV";
    }
    // Determine the required size of the URL buffer
    int url_size = snprintf(NULL, 0,
        "https://dds.dui.ai/runtime/v1/synthesize?voiceId=%s&text=%s&speed=1&volume=%d&audiotype=%s", \
            VOICE_ID, \
            encoded_message, \
            VOLUME, \
            codec_format_str);
    // Allocate memory for the URL buffer
    char *url = heap_caps_malloc((url_size + 1), MALLOC_CAP_SPIRAM | MALLOC_CAP_8BIT);
    if (url == NULL) {
        ESP_LOGE(TAG, "Failed to allocate memory for URL");
        return ESP_ERR_NO_MEM;
    }
    // Format the URL string
    snprintf(url, url_size + 1,
        "https://dds.dui.ai/runtime/v1/synthesize?voiceId=%s&text=%s&speed=1&volume=%d&audiotype=%s", \
            VOICE_ID, \
            encoded_message, \
            VOLUME, \
            codec_format_str);
    // Configure the HTTP client
    esp_http_client_config_t config = {
        .url = url,
        .method = HTTP_METHOD_GET,
        .event_handler = http_event_handler,
        .buffer_size = MAX_FILE_SIZE,
        .buffer_size_tx = 4000,
        .timeout_ms = 30000,
        .crt_bundle_attach = esp_crt_bundle_attach,
    };
    // Initialize and perform the HTTP request
    esp_http_client_handle_t client = esp_http_client_init(&config);
    esp_err_t err = esp_http_client_perform(client);
    if (err != ESP_OK) {
        ESP_LOGE(TAG, "HTTP GET request failed: %s", esp_err_to_name(err));
    }
    // Free allocated memory and clean up the HTTP client
    heap_caps_free(url);
    heap_caps_free(encoded_message);
    esp_http_client_cleanup(client);
    // Return the result of the function call
    return err;
}
```

Enfin, l'événement `HTTP_EVENT_ON_FINISH` est utilisé pour gérer la fin de la réponse HTTP. Dans ce cas, `record_audio_buffer` est envoyé à une fonction appelée `audio_player_play()` qui joue l'audio.

Affichage

Pour l'affichage, nous utilisons LVGL, une bibliothèque graphique embarquée open-source qui gagne en popularité grâce à ses fonctions puissantes et visuellement attrayantes et à sa faible consommation de mémoire. LVGL a également publié un éditeur d'interface utilisateur par glisser-déposer appelé SquareLine Studio [14]. C'est un outil puissant qui facilite la création d'interfaces graphiques de qualité pour vos applications.

Pour intégrer LVGL à votre projet, Espressif Systems fournit un outil officiel de gestion de paquets [15]. Cet outil vous permet d'ajouter directement LVGL et les composants de portage associés à votre projet, vous faisant gagner du temps et de l'effort. Pour plus d'informations, suivez les blogs officiels [16] et les documentations [17].

Créer des appareils IdO intelligents

L'intégration de ChatGPT d'OpenAI avec l'ESP-BOX d'Espressif a ouvert de nouvelles possibilités pour créer des appareils IdO puissants et intelligents. L'ESP-BOX fournit une plateforme de développement AIoT flexible et personnalisable avec des fonctions telles que l'interaction vocale en champ lointain, la reconnaissance de commandes vocales hors ligne et un cadre d'interface graphique réutilisable. En combinant ces capacités avec l'API OpenAI, les développeurs peuvent créer des chatbots vocaux et améliorer l'expérience des utilisateurs des applications IdO.

N'oubliez pas de consulter le dépôt GitHub [19] d'Espressif Systems [18] pour d'autres démonstrations open-source sur ESP-IoT-Solution [20], ESP-SR et ESP-BOX. Le code source de ce projet est disponible sur GitHub [21]. Dans le cadre de nos projets futurs, nous visons à introduire un composant pour l'API OpenAI qui offrira des fonctions conviviales. ◀

230462-04

Questions ou commentaires ?

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



À propos de l'auteur

Ali Hassan Shah est ingénieur en logiciel embarqué, passionné par les technologies IdO. En tant que membre estimé de l'équipe d'ingénierie d'application d'Espressif Systems, il met son expertise au service de la mission consistant à rendre la technologie accessible sans effort et conviviale pour tous, guidée par la devise « simplifier la technologie pour tous ».



Produits

> **ESP32-S3-BOX-3**
www.elektor.fr/20627



Listing 6: Handling TTS Response.

```
static esp_err_t http_event_handler(esp_http_client_event_t *evt)
{
    switch (evt->event_id) {
        // Handle errors that occur during the HTTP request
        case HTTP_EVENT_ERROR:
            ESP_LOGE(TAG, "HTTP_EVENT_ERROR");
            break;
        // Handle when the HTTP client is connected
        case HTTP_EVENT_ON_CONNECTED:
            ESP_LOGI(TAG, "HTTP_EVENT_ON_CONNECTED");
            break;
        // Handle when the header of the HTTP request is sent
        case HTTP_EVENT_HEADER_SENT:
            ESP_LOGI(TAG, "HTTP_EVENT_HEADER_SENT");
            break;
    }
}
```

```

// Handle when the header of the HTTP response is received
case HTTP_EVENT_ON_HEADER:
    ESP_LOGI(TAG, "HTTP_EVENT_ON_HEADER");
    file_total_len = 0;
    break;
// Handle when data is received in the HTTP response
case HTTP_EVENT_ON_DATA:
    ESP_LOGI(TAG, "HTTP_EVENT_ON_DATA, len=%d", evt->data_len);
    if ((file_total_len + evt->data_len) < MAX_FILE_SIZE) {
        memcpy(record_audio_buffer + file_total_len, (char *)evt->data, evt->data_len);
        file_total_len += evt->data_len;
    }
    break;
// Handle when the HTTP request finishes
case HTTP_EVENT_ON_FINISH:
    ESP_LOGI(TAG, "HTTP_EVENT_ON_FINISH:%d, %d K", file_total_len, file_total_len / 1024);
    audio_player_play(record_audio_buffer, file_total_len);
    break;
// Handle when the HTTP client is disconnected
case HTTP_EVENT_DISCONNECTED:
    ESP_LOGI(TAG, "HTTP_EVENT_DISCONNECTED");
    break;
// Handle when a redirection occurs in the HTTP request
case HTTP_EVENT_REDIRECT:
    ESP_LOGI(TAG, "HTTP_EVENT_REDIRECT");
    break;
}
return ESP_OK;
}

```

LIENS

- [1] ESP-Box : <https://github.com/espressif/esp-box>
- [2] ESP32-S3 product selector : <https://tinyurl.com/esp32s3prodsel>
- [3] Guide de programmation IDF : <https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32/index.html>
- [4] Plate-forme OpenAI : <https://openai.com>
- [5] Lien de référence officiel de l'API : <https://platform.openai.com/docs/api-reference>
- [6] ESP-SR : <https://github.com/espressif/esp-sr>
- [7] Guide de l'utilisateur de l'ESP-SR : <https://docs.espressif.com/projects/esp-sr/en/latest/esp32/index.html>
- [8] Chat Completion API : <https://platform.openai.com/docs/api-reference/chat/create>
- [9] JSON parser : https://components.espressif.com/components/espressif/json_parser
- [10] GitHub parser : https://github.com/espressif/json_parser
- [11] Voicerss : <https://voicerss.org/api>
- [12] TTSMaker : <https://ttsmaker.com/zh-cn>
- [13] TalkingGenie : <https://talkinggenie.com>
- [14] SquareLine Studio : <https://squareline.io>
- [15] Outil officiel de gestion des paquets pour LVGL : <https://components.espressif.com/components/lvgl/lvgl>
- [16] Blog about LVGL : <https://tinyurl.com/esp32fancyui>
- [17] Documentation de LVGL : <https://docs.lvgl.io/master/index.html>
- [18] Espressif Systems : <https://espressif.com/>
- [19] Dépôt GitHub d'Espressif Systems : <https://github.com/orgs/espressif/repositories>
- [20] ESP-IoT-Solution : <https://github.com/espressif/esp-iot-solution>
- [21] Code source de ce projet : <https://github.com/espressif/esp-box/tree/master/examples>



ESP-Unlock

clé d'authentification à deux facteurs basée sur l'ESP32-C3

Jakob Hasse, *Espressif*

De nombreux services en ligne proposent aujourd'hui une authentification à deux facteurs. L'application d'authentification pour smartphone est probablement le choix le plus populaire, cependant un smartphone peut être volé, et souvent les données privées et professionnelles sont mélangées. C'est là que les jetons matériels en forme de clé USB sont utiles. Dans ce projet, nous avons créé un tel jeton basé sur une carte ESP32-C3 peu coûteuse.

Imaginez que vous êtes un attaquant cyber et que vous avez obtenu le nom d'utilisateur et le mot de passe de votre victime pour un service en ligne quelconque. Vous essayez maintenant de vous connecter. Cela fonctionne ! Mais le service en ligne vous demande alors un « code de vérification ». Pourquoi ? Parce que ce service en ligne propose une authentification à deux facteurs et que votre victime l'a activée. L'authentification à deux facteurs - dont les acronymes courants sont 2FA, TFA ou MFA - signifie essentiellement que vous avez besoin d'une étape supplémentaire pour vous authentifier auprès d'un service en ligne. Vous vous authentifiez non seulement par des données que vous connaissez, mais aussi par quelque chose que vous possédez. Dans la pratique, cette étape supplémentaire implique souvent un dispositif matériel auquel vous avez accès. Un bon exemple est le retrait d'argent à un distributeur automatique de billets, pour lequel vous avez besoin d'un code PIN (connaissance) et de votre carte de débit (possession). Il ne suffit pas de connaître le code PIN ou de posséder la carte pour accéder au compte bancaire d'une victime.

Mais les banques ne sont pas les seules à utiliser l'authentification à deux facteurs. De nos jours, de nombreux services en ligne proposent également une authentification à deux facteurs. Contrairement aux banques, ils ne distribuent pas de cartes bancaires. Au lieu de cela, vous pouvez utiliser une application d'authentification pour smartphone ou un appareil électronique standard, que nous appellerons dorénavant «jeton matériel». Vous devrez toujours vous souvenir de vos données d'identification habituelles pour vous authentifier (connaissance), mais vous devrez également utiliser l'application d'authentification de votre smartphone ou le jeton matériel (possession).

L'application d'authentification pour smartphone est probablement le choix le plus courant, puisque pratiquement tout le monde possède un smartphone. Mais qu'en est-il du backup en cas de vol du téléphone ? Qu'en est-il de la séparation des comptes professionnels et du téléphone privé ? Ou s'il n'est tout simplement pas pratique pour vous de sortir votre smartphone de luxe de votre poche ? C'est là que les jetons matériels s'avèrent utiles.

Non seulement ils vous permettent de ne plus dépendre d'un smartphone, mais ils sont également petits et bon marché. Vous pouvez toujours emporter un petit jeton matériel dans diverses situations. Un jeton matériel peu coûteux permet à un plus grand nombre de personnes disposant d'un faible budget de l'utiliser, et il est possible de posséder plusieurs jetons matériels à des fins différentes ou pour un usage de secours sans trop coûter.

Fonctionnement des jetons matériels

Vous vous demandez peut-être comment le service en ligne peut distinguer votre jeton matériel de n'importe quel autre jeton ? La réponse est que le jeton matériel est en fait « intelligent » puisqu'il est doté d'un petit microprocesseur et d'une mémoire. Lorsque vous configurez le jeton matériel en tant qu'authentificateur pour le service en ligne, vous obtenez une clé de cryptage enregistrée dans la mémoire (plus de détails à ce sujet plus loin). La clé est ensuite utilisée pour prouver l'authenticité du jeton matériel. Vous pouvez l'imaginer comme un autre mot de passe mémorisé par le jeton matériel pour se connecter au service en ligne. Cependant, l'utilisation de ce type de jeton matériel est très similaire à celle d'une clé physique. Si vous utilisez plutôt une application d'authentification pour smartphone, celle-ci enregistrera la clé quelque part sur votre téléphone, mais les principes expliqués ici seront les mêmes. L'une des normes d'authentification à deux facteurs est la norme TOTP (*time-based one-time password*), largement utilisée [1]. Le TOTP est basé sur des mots de passe à usage unique (OTP), créés à partir d'une clé cryptographique et de l'heure actuelle de l'horloge, d'où le terme « basé sur le temps ». La clé est

partagée entre le service en ligne et le jeton matériel. Une fonction de hachage crée un OTP numérique à 6 chiffres à partir de la clé et de l'heure actuelle. Le service en ligne effectue le même calcul avec sa propre copie de la clé et de l'heure actuelle. L'OTP calculé par le jeton matériel est envoyé au service en ligne, qui le compare à l'OTP calculé à partir de sa propre copie de la clé. L'utilisateur n'est authentifié que si les deux OTP correspondent. Lors du calcul de l'OTP, une fonction de hachage est utilisée pour garantir que la clé originale n'est pas compromise lors de son transfert du jeton matériel au service en ligne. Pour plus d'informations sur la norme TOTP, voir [1].

Jeton matériel ESP-Unclock

Étant donné que les jetons matériels d'authentification à deux facteurs sont très pratiques et que la norme TOTP est largement utilisée et facile à mettre en œuvre, j'ai créé l'« ESP-Unclock » : un jeton matériel d'authentification à deux facteurs, open-source et compatible avec la norme TOTP.

J'ai choisi la puce ESP32-C3 assez peu coûteuse pour le jeton matériel puisque je suis familier avec la puce et l'Espressif IoT development framework (ESP-IDF) [2], qui permet de programmer un ESP32-C3. Pour être plus précis, j'ai choisi un module ESP32-C3-WROOM-02U qui intègre l'ESP32-C3 et d'autres composants utiles.

L'ESP32-C3 présente quelques caractéristiques très utiles :

- > Un convertisseur USB-série intégré qui permet la communication via USB, une interface disponible sur pratiquement tous les ordinateurs.
- > Un processeur avec des accélérateurs cryptographiques pour calculer l'OTP.
- > Du matériel permettant le cryptage de la mémoire flash pour chiffrer la clé secrète et sécuriser le démarrage pour empêcher l'exécution d'un code non autorisé sur le jeton matériel.
- > Le module ESP32-C3-WROOM-02U comprend une mémoire flash pour stocker le code du programme et la clé.

Le cadre de développement ESP-IDF permet le cryptage de la mémoire flash et l'amorçage sécurisé au niveau du matériel, en tant que fonction facilement accessible. Les seules pièces manquantes étaient un circuit imprimé pour assembler tout le matériel et un logiciel pour connecter les différents éléments.

Le projet ESP-Unclock combine tous les éléments : un jeton matériel TOTP open-source simple et bon marché et une application d'authentification pour un ordinateur hôte, dont nous discuterons plus loin. Nous allons maintenant aborder l'architecture du projet entier, le matériel et le logiciel, puis discuter de certaines considérations relatives à la sécurité et enfin voir comment utiliser l'ESP-Unclock.

Architecture du projet

Pour calculer et afficher l'OTP, vous avez besoin du jeton matériel lui-même, appelé ESP-Unclock, ainsi que d'un ordinateur hôte doté d'un port USB-A. L'ESP-Unclock stocke les clés permettant de générer des OTP pour différents services en ligne. L'hôte fournit l'heure actuelle via son horloge en temps réel. La communication entre les deux est assurée par une communication série sur USB, en utilisant le périphérique USB-Serial de l'ESP32-C3.

Le calcul d'un OTP nécessite quatre étapes (voir **figure 1**) :

1. L'ordinateur hôte obtient l'heure actuelle à partir de sa propre horloge en temps réel.
2. L'ordinateur hôte demande à l'ESP-Unclock de calculer l'OTP.
3. L'ESP-Unclock utilise la clé appropriée pour calculer l'OTP.
4. Le système ESP-Unclock renvoie l'OTP dans un message.

La communication (voir les étapes 2 et 4 de la figure 1) entre l'hôte et l'ESP-Unclock repose sur

un protocole requête-réponse assez simple, l'hôte agissant toujours en tant qu'initiateur, tandis que l'ESP-Unclock ne fait que répondre aux requêtes. Tous les messages sont envoyés en clair pour faciliter le débogage. Un nom de service dans la requête est nécessaire car le jeton matériel est utilisé avec plusieurs services en ligne, ce qui nécessite une clé par service. L'exemple de requête de la figure 1 est le suivant :

`TOTP:github,1690975870`

où **TOTP** : et le caractère de nouvelle ligne à la fin sont des délimiteurs pour l'analyse, `github` est le nom de service de la clé à choisir et `1690975870` est l'heure UNIX actuelle, formattée en nombre décimal. Le message de réponse correspondant dans la figure 1 est :

`TOTP:123456`

où **TOTP** : et la nouvelle ligne sont à nouveau des délimiteurs et `123456` est l'OTP. Pour plus d'informations sur ce protocole et sur les messages supplémentaires permettant d'énumérer les noms de services et d'ajouter de nouveaux groupes de clés et de noms de services, consultez le dépôt du projet [3].

Matériel

La taille réduite étant une exigence primordiale pour le matériel, le circuit imprimé de l'ESP-Unclock ne contient que les composants nécessaires : un module ESP32-C3-WROOM (sans antenne), le circuit d'amorçage

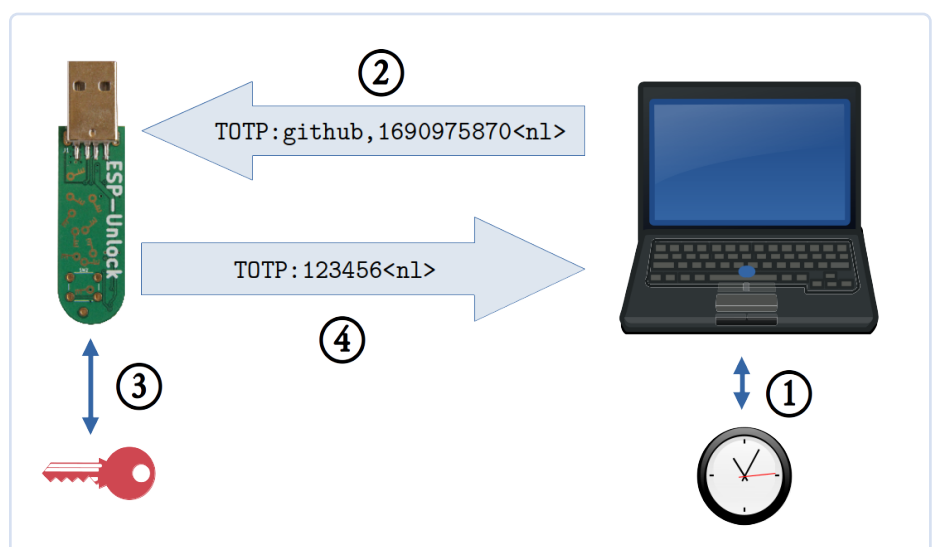


Figure 1. Communication entre un jeton matériel ESP-Unclock et l'ordinateur hôte.

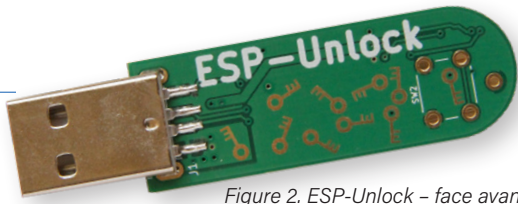


Figure 2. ESP-Unlock - face avant.

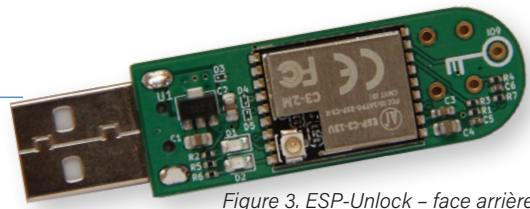


Figure 3. ESP-Unlock - face arrière.

correspondant, une alimentation (3,3 V fournis par l'alimentation USB 5 V), deux LED et un connecteur USB-A pour la connexion à l'ordinateur. Il y a également une empreinte pour un bouton optionnel qu'on peut utiliser dans les versions futures du logiciel pour permettre le calcul de l'OTP seulement si le bouton est pressé, ce qui augmente la sécurité. En incluant le connecteur USB, le matériel obtenu mesure 6×1,6 cm, soit à peu près la taille d'une clé de porte normale (figure 2 et figure 3). Nous avons conçu le matériel avec KiCad, un logiciel libre qui peut être utilisé pour créer des circuits et concevoir des schémas de circuits imprimés. Le circuit imprimé de l'ESP-Unlock comporte deux couches (figure 4), car c'est généralement l'option la moins chère. Tous les composants, à l'exception des traversants, sont situés d'un seul côté, ce qui permet de souder facilement tous les CMS par refusion.

Logiciel

Deux applications logicielles sont utiles pour ce projet : un micrologiciel fonctionnant sur l'ESP-Unlock, le rendant « intelligent », et une application fonctionnant sur l'ordinateur hôte, appelée application d'authentification de l'ESP-Unlock.

Le micrologiciel exécuté sur l'ESP-Unlock est une application C++ utilisant ESP-IDF [2] et ESP-IDF-C++ [4]. Dès que le micrologiciel de l'ESP-Unlock démarre, l'application commence à écouter les requêtes d'OTP. Les deux LED clignotent une fois en alternance lorsque l'appareil est prêt à recevoir des requêtes. Une fois qu'une demande d'OTP a été reçue, le micrologiciel ESP-Unlock vérifie s'il dispose d'une clé correspondant au nom indiqué dans la demande. Dans l'affirmative, il recherche la clé correspondante. Il calcule ensuite l'OTP en utilisant la clé qui vient d'être lue et l'heure indiquée dans la demande d'OTP. Un message de réponse contenant l'OTP est renvoyé à l'hôte (voir figure 1). Il convient de noter que tout le calcul de l'OTP est effectué sur l'ESP-Unlock - la clé ne le quitte jamais. L'application d'authentification ESP-Unlock est chargée de fournir une interface utilisateur simple et de coordonner la communication avec l'ESP-Unlock. L'interface utilisateur répertorie tous les services pour lesquels des clés sont disponibles sur l'ESP-Unlock actuellement connecté, affiche les OTP pour les

services correspondants et permet d'ajouter de nouvelles clés. L'application est écrite pour Linux uniquement, mais le micrologiciel de l'ESP-Unlock ne se préoccupe pas du système d'exploitation de l'hôte, de sorte que l'application pourrait être réimplémentée ou portée sur n'importe quel autre système d'exploitation.

Considérations de sécurité

Veillez noter qu'il ne s'agit pas d'une analyse formelle de la sécurité, qui serait trop complexe pour le projet à son stade actuel. Il s'agit de signaler les dangers potentiels et de les expliquer au lecteur.

Les données critiques de l'ESP-Unlock sont les clés, qui doivent rester secrètes en permanence. Il est possible de bloquer l'accès direct aux clés en utilisant le cryptage flash. Mais le logiciel fonctionnant sur l'ESP-Unlock a également accès aux clés. Par conséquent, seul un logiciel autorisé doit être exécuté afin d'empêcher les logiciels non autorisés de divulguer les clés. Le démarrage sécurisé (Secure Boot) garantit que seul le code autorisé s'exécute sur l'ESP32-C3. Si *Secure Boot* et *flash encryption* sont activés, même un attaquant ayant un accès physique à l'ESP-Unlock ne peut pas lire les données de la clé.

Un autre facteur est que le logiciel actuel permet à toute personne ayant un accès physique à l'ESP-Unlock de générer et de lire des OTP. Si vous le perdez et que quelqu'un d'autre s'en empare, cette personne peut générer des OTP tout comme vous.

Le micrologiciel fonctionnant sur le système ESP-Unlock peut comporter des failles. Dans le pire des cas, un tel exploit permettrait à un attaquant d'exécuter son propre code sur l'ESP-Unlock. L'activation du protecteur de pile, configurable dans le micrologiciel, consti-

tue une amélioration qui permet de limiter certaines catégories d'exploits. Une autre amélioration consisterait en un examen du code axé sur la sécurité, ce qui n'a pas été fait étant donné que le code est un prototype précoce. Il convient de noter que l'amorçage sécurisé ne protège pas contre l'exploitation du code. Le démarrage sécurisé protège uniquement contre l'exécution d'un code non autorisé sur le jeton matériel. Il ne protège pas contre un code autorisé qui se comporte mal et permet une exécution aléatoire du code. L'exploitation du code est plus probable en cas d'accès physique à l'ESP-Unlock. Il faut également envisager des attaques utilisant un logiciel sur l'ordinateur hôte en tant que proxy, mais elles sont moins probables en raison des étapes supplémentaires.

Compte tenu de ces aspects, si un attaquant accède physiquement à un ESP-Unlock, le propriétaire doit créer un nouvel ESP-Unlock et modifier les clés secrètes de tous les services en ligne concernés dès que possible. Il convient toutefois de noter que tant que le hacker n'obtient pas les informations d'identification de l'utilisateur (nom d'utilisateur et mot de passe), il ne pourra toujours pas se connecter. Après tout, l'OTP n'est que le deuxième facteur, et vous devriez être aussi en sécurité que si vous n'aviez pas d'authentification à deux facteurs.

Utilisation d'ESP-Unlock

Avant que l'ESP-Unlock puisse générer des OTP, nous devons installer l'application d'authentification ESP-Unlock et ajouter la clé pour l'authentification à deux facteurs. Pour plus d'informations sur l'installation de l'application d'authentification ESP-Unlock, veuillez vous référer aux instructions disponibles

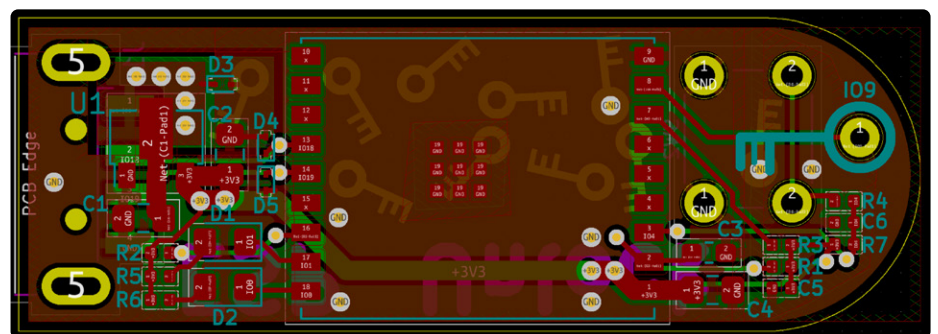


Figure 4. Couches supérieure, inférieure et sérigraphiée du circuit imprimé.

Can't scan the code?

To add the entry manually, provide the following details on your phone.

Account: gitlab.example.com:admin@example.com
Key: XYWH ZQDE XAMO BWLR 57AC EFTO LXE3 MN6N
 Time based: Yes

Current password

Your current password is required to register a two-factor authenticator app.

Enter verification code

Register with two-factor app

Figure 5. Clé codée en base32 dans GitLab (marquée en rouge).

esp-unlock

Device Found

ADD NEW 1

esp-unlock

gitlab 2

757ACEFT0 XE3MN6N 3

Add Entry 4

Figure 6. Ajouter une nouvelle clé.

esp-unlock

Device Found

github 1

782741 2

ADD NEW

Figure 7. Demander un OTP.

dans son dépôt [5]. Pour ajouter votre clé, vous devez d'abord la récupérer auprès du service en ligne compatible que vous souhaitez utiliser. La manière dont cela fonctionne dépend du service en ligne spécifique, mais normalement, vous serez guidé à travers le processus de configuration de votre dispositif ou application d'authentification à deux facteurs. Les services affichent généralement la clé sous la forme d'un code QR, ce qui est pratique lorsque vous utilisez une application d'authentification pour smartphone, mais pour l'ESP-Unlock, le format texte, formaté en base32, est nécessaire. Par exemple, GitLab fournit la clé encodée en base32 sur la page de configuration de l'authentification à deux facteurs, en plus du code QR (figure 5).

Une fois l'ESP-Unlock connecté à l'ordinateur hôte, il est possible de configurer l'authentification à deux facteurs pour un nouveau service dans l'application de l'authentificateur en suivant les étapes suivantes (voir figure 6) :

1. Cliquez sur **ADD NEW**.
2. Dans la fenêtre qui apparaît, choisissez un nom de service et saisissez-le.
3. Saisissez la clé elle-même, formatée en base32. Notez que vous devez supprimer les espaces, sinon la clé du système ESP-Unlock sera incorrecte.
4. Cliquez sur **Add Entry**.

Chaque fois que vous avez besoin d'un OTP, vous insérez l'ESP-Unlock dans le port USB de l'hôte et vous lancez l'application d'authentification ESP-Unlock, qui affichera les noms de toutes les clés enregistrées sur l'ESP-Unlock et, en cliquant sur un bouton (étape 1 dans la figure 7), demandera et affichera l'OTP correspondant (étape 2 dans la figure 7).

Si ce projet vous semble intéressant, n'hésitez pas à y jeter un coup d'œil ou à le tester en visitant le dépôt ESP-Unlock [3] et le dépôt correspondant de l'application d'authentification ESP-Unlock [5]. Les schémas et la disposition du circuit imprimé sont disponibles dans le dépôt du matériel [6], mais le matériel ESP-Unlock n'est pas nécessaire ! Vous pouvez en effet utiliser n'importe quelle carte de développement de la série ESP32, avec une configuration légèrement modifiée (dans ce cas, veuillez également vous référer au fichier *README.md* du dépôt ESP-Unlock [3]). Toutes les contributions au projet, telles que les suggestions, les astuces, les améliorations et les rapports de bogues, sont les bienvenues !

230559-04

Questions ou commentaires ?

Envoyez un courriel à l'auteur (jakob.hasse@mailbox.org) ou contactez Elektor (redaction@elektor.fr).



À propos de l'auteur

Jakob Hasse est diplômé en sciences de l'informatique et ses centres d'intérêt sont variés. Pendant ses études, il a travaillé au Centre aérospatial allemand dans le domaine de la robotique. Plus tard, il s'est passionné pour les systèmes embarqués et les logiciels libres. Après avoir débuté comme ingénieur en systèmes Linux embarqués, Jakob a rejoint Espressif pour travailler sur leur Espressif IoT Development Framework basé sur FreeRTOS. En outre, il s'intéresse à la sécurité informatique, qui est le cœur du projet ESP-Unlock.

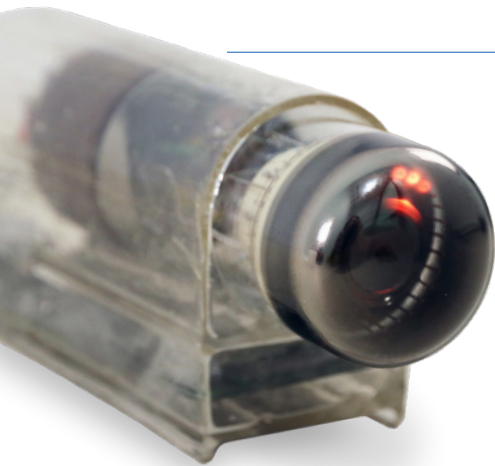


Produits

- > **ESP32-C3-WROOM-02**
www.elektor.fr/20695
- > **Peter Dalmaris, KiCad 6 Like A Pro (Bundle of two books)**
www.elektor.fr/20180

LIENS

- [1] Norme Time-based One Time Password (TOTP) : <https://ietf.org/rfc/rfc6238.txt>
- [2] Espressif IoT Development Framework (ESP-IDF) : <https://github.com/espressif/esp-idf>
- [3] Dépôt du projet : <https://github.com/0xjakob/esp-unlock>
- [4] ESP-IDF-C++ : <https://github.com/espressif/esp-idf-cxx>
- [5] Application d'authentification ESP-Unlock : <https://github.com/0xjakob/esp-unlock-host-gui>
- [6] Dépôt GitHub du matériel : <https://github.com/0xjakob/esp-unlock-hardware>



Dekatron

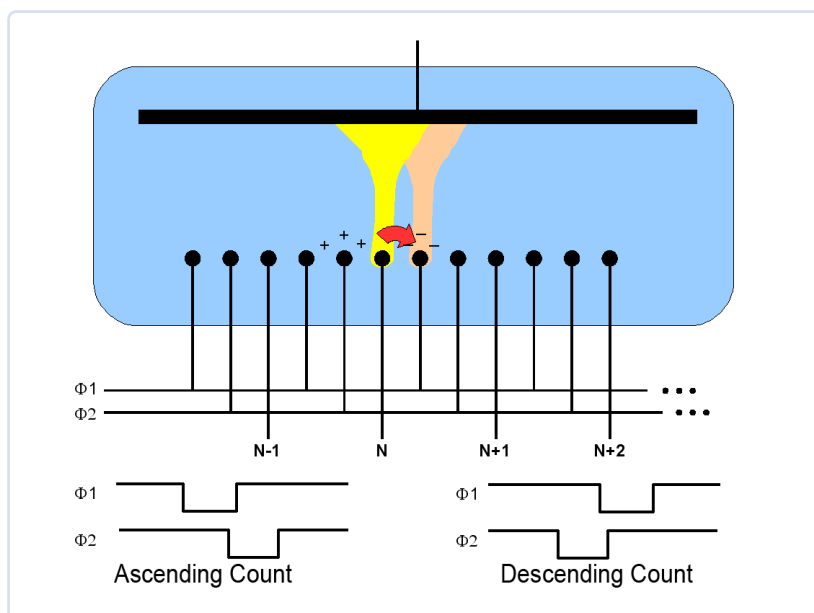
une époque historique renaît !

Jeroen Domburg, Espressif

En musique, le mélange de styles anciens et modernes donne souvent des résultats extraordinaires, en électronique il en va de même : cet article décrit comment le très moderne ESP32-C3 pilote avec brio un dekatron, un compteur décimal à tube des années 1950 qui reprend vie après 70 ans !

Contrairement à Elektor, vénérable institut, Espressif n'a pas vraiment un long passé. C'est sans doute mieux ainsi : en 1961, le marché des puces Wifi IoT était à l'évidence inexistant. En revanche, Espressif, créée en 2008, a sorti ses premières puces cinq ans plus tard : les ESP8089 et 8266. L'ESP8089 était conçue pour l'OEM afin de conférer une capacité Wifi par ex. aux tablettes Android, tandis que l'ESP8266 est une véritable puce Wifi IoT. De nos

Figure 1. Principe de base du fonctionnement d'un dekatron (source : Wikimedia Commons)



jours, des personnes lancent encore de nouveaux projets basés sur l'ESP8266. Cela montre à quel point ce produit n'a rien de "rétro" (Toutefois, si vous envisagez un projet avec un ESP8266, nous vous conseillons d'utiliser plutôt un ESP32-C3 ; en effet, une demi-décennie d'innovation en fait une puce bien plus agréable à un prix voisin).

Je me suis dit qu'il serait plus intéressant d'étudier si une interface entre l'ancien et le nouveau était réalisable. Pour cela, je vais marier un dekatron vieux de plus de 50 ans au presque nouveau venu qu'est l'ESP32-C3. Le tout servira d'indicateur numérique de la bande passante (BP) Internet que nous utilisons. Ce ne pouvait pas être un usage envisagé au moment où le dekatron a été conçu, mais cela reste un bon usage.

Le tube à vide Dekatron

Si vous n'êtes pas familiarisé avec la rétronique, sachez qu'un dekatron est un tube de verre rempli de gaz. (Notez que ce n'est pas le cas d'un tube à vide, qui est essentiellement "rempli" de vide donc sans gaz). En général, les tubes à gaz contiennent du néon ou un autre gaz inerte, mais l'hydrogène a aussi été utilisé. Si le terme "néon" vous évoque les voyants au néon des boutons marche/arrêt de vieux appareils, vous êtes sur la bonne voie : les principes sont similaires.

Un dekatron n'est qu'un compteur avec indication visuelle du comptage. Il fonctionne comme suit : le dekatron a une anode ronde au sommet. 30 cathodes formant des broches régulièrement espacées sont disposées autour comme sur le dessin explicatif de la **figure 1** et sur les clichés réels de la **figure 2**. Dix de ces cathodes sont dites de sortie, les vingt autres sont des cathodes de guidage, divisées en deux groupes : G1 et G2, respectivement $\Phi 1$ et $\Phi 2$ sur la **figure 1**. Chaque cathode de sortie a deux cathodes guides G1 d'un côté et G2 de l'autre. Les cathodes G1 sont toutes en parallèle, de même que les G2. Au démarrage, du 400 V limité en courant est appliqué entre l'anode et les cathodes de sortie. Cette tension étant supérieure à sa tension d'ionisation, le gaz s'ionise et s'allume à l'une de ces cathodes. Le courant qui passe ramène la tension d'anode sous la tension d'ionisation et aucune autre électrode ne s'allume. Cette lueur peut être "transférée" aux cathodes de sortie adjacentes en



mettant tour à tour G1 et G2 à la terre - par ex., G1 puis G2, l'ionisation se déplace dans le sens horaire ; G2 puis G1 fait se déplacer l'ionisation dans le sens antihoraire. À quoi servent toutes ces manigances d'ionisation ? Eh bien, cet assemblage de verre, de métal et de gaz est capable de compter les impulsions G1/G2 ; précisément, ils peuvent en compter jusqu'à **dix** soit un tour complet (d'où le nom de **dekatron**). En général, une partie ou la totalité des cathodes sont reliées à des broches à la base du dekatron, afin de détecter certains comptages. Le dekatron peut servir de compteur d'impulsions, de diviseur de fréquence, et même d'élément de mémoire : un célèbre ordinateur de 1951 en utilise à cette fin [2]. (Notez que la phrase précédente est au présent : l'ordinateur fonctionne toujours dans un musée).

Les progrès successifs de l'électronique vinrent à bout des dekatrons, d'abord les compteurs à transistors discrets. Ensuite, les CI exécutèrent des tâches plus complexes, les compteurs décimaux comme le bon vieux CD4017 prirent le relais. À l'heure des microcontrôleurs et des FPGA bon marché, la fonction du dekatron fut finalement confiée à quelques lignes de code ou HDL. Mais toutes ces nouvelles techniques de comptage ont un inconvénient. Contrairement au dekatron, elles n'ont pas de sortie visuelle propre. Bien sûr, vous pouvez relier des LED à votre CD4017 (je soupçonne d'ailleurs nombre de lecteurs de l'avoir fait autrefois), ou mettre un spinner sur le LCD relié à votre µcontrôleur dernier cri, mais la lueur du néon du dekatron a un charme que ni les pixels, ni les semi-conducteurs électroluminescents ne peuvent imiter.

Défis liés à l'alimentation

C'est là que se situe le problème : l'ESP32-C3 est tout à fait adapté à l'éclairage des pixels ou des LED, mais que dire de son interfacement avec une antique verrerie de 70 ans : malgré tout ce qu'il peut apporter au dekatron, avec 3,3 V en E/S, l'ESP32-C3 ne fait pas le poids. C.-à-d. qu'il faut trouver un moyen de générer 400 V et aussi commuter plusieurs tensions élevées.

Commençons par le 400 V. Le plus simple est d'utiliser un doubleur de tension sur la tension du secteur. Cependant, je ne voulais pas vraiment utiliser ici la tension du secteur ; les précautions de sécurité nécessaires à cet égard sont trop importantes. J'ai préféré opter pour un connecteur USB-C moderne ; entre les blocs secteur d'ordinateur portable, les chargeurs de téléphone mobile et ce chargeur lambda reçu avec un appareil électronique (mais vous ne savez plus lequel), il y a assez d'alimentations avec ce connecteur pour vous en sortir. De plus, l'USB-PD est désormais une norme commune qui permet de demander des tensions plus élevées que les 5 V normalement disponibles sur un port USB. Cela s'est avéré utile dans ce cas.

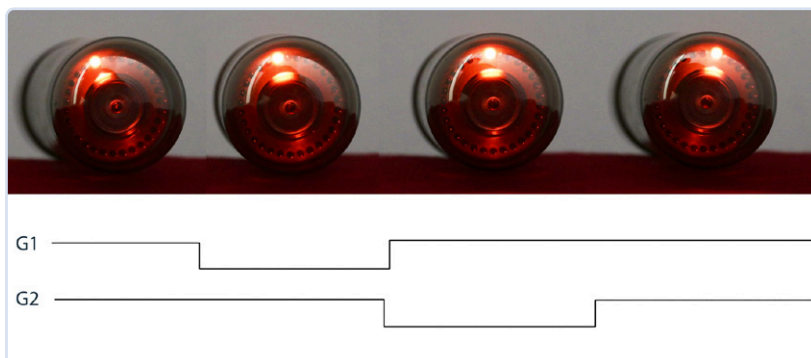
Alors comment arriver à 400 V ? Il y a pas mal de façons

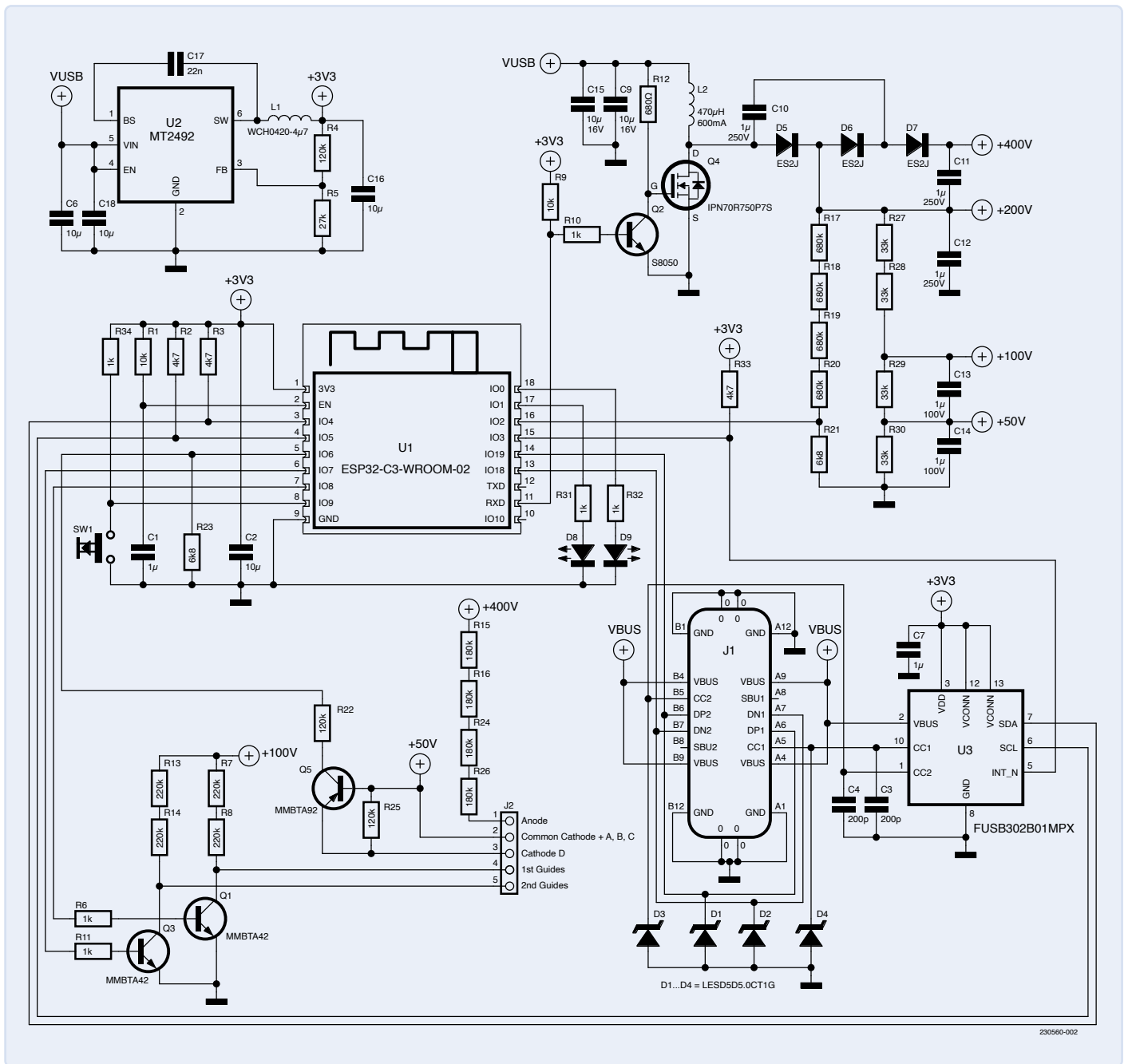
de procéder. La plus simple fait appel à des transformateurs : c.-à-d. convertir votre BT en courant alternatif, l'injecter dans l'enroulement BT d'un transformateur et obtenir une HT sur l'enroulement de sortie. On peut aussi l'utiliser comme convertisseur flyback : produire un champ magnétique en appliquant une tension au côté BT, couper la BT et faire en sorte que l'énergie du champ magnétique soit récupérée par le côté HT, et on aura notre sortie HT. Le problème avec ces deux solutions est que les transformateurs miniatures sont plutôt des produits de niche, et le risque qu'on ne puisse pas les réapprovisionner ultérieurement est élevé. Bien que je ne m'attende pas à ce que ce modèle soit fabriqué par millions, j'aimerais qu'il fonctionne et qu'il soit réparable longtemps, donc je ne veux pas de composants de niche.

La solution

J'ai donc opté pour une autre solution : un convertisseur élévateur. En général, un convertisseur boost est utilisé pour élever la valeur d'une tension. On peut, par ex., utiliser un faire fonctionner une logique de 5 V à partir de deux piles AA. L'inconvénient de ces convertisseurs est qu'il est assez difficile d'augmenter fortement la tension et aussi que l'élément commutant la BT doit également encaisser la HT de sortie, et la valeur maximale dont on augmente la tension d'entrée est limitée par des éléments comme la résistance en continu de l'inductance utilisée. Un doubleur de tension placé derrière le convertisseur est une astuce qui résout ce problème ; de cette façon, élever la tension à 200 V suffit. (Notez que cette astuce est tirée d'un circuit de convertisseur boost dekatron qui a fait ses preuves [2]) L'autre astuce consiste à partir d'une tension plus élevée : en général, les chargeurs USB-C compatibles PD peuvent au besoin fournir 5 V, mais aussi 9 V, 15 V ou 20 V. C.-à-d. que le convertisseur boost travaille moins dur pour augmenter la tension d'entrée. (On pourrait aussi utiliser un deuxième convertisseur élévateur pour augmenter un peu la tension d'alimentation avant de l'augmenter jusqu'à 400 V). J'ai choisi la voie USB-PD car je voulais de toute façon jouer avec.

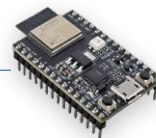
Figure 2. Le dekatron en comptage avant, et les signaux correspondants sur G1 et G2.





▲ Pour les détails de mise en œuvre, voir **figure 3**. La tension du port USB, J1, est filtrée par deux condensateurs céramique. L2 et Q4 forment la logique boost principale : lorsque Q4 conduit, il crée un champ magnétique, et lorsqu'il est ouvert, le courant qui s'oppose à l'effondrement du champ du champ charge C11 et C12 via le redressement et le doubleur formé par D5 à D7. Ainsi naissent deux tensions continues, l'une de 200 V et l'autre de 400 V. Q4 est spécial ; comme dit précédemment, il doit résister aux tensions élevées générées par L2. Par malheur, les MOSFETS capables de le faire sont incompatibles avec

un niveau logique. Q2 sert donc à convertir le signal MLI (Modulation en largeur d'impulsion, en anglais *MLI Pulse Width Modulation*) de 3,3 V entrant en un signal qui peut piloter Q4. Ce signal MLI est produit par l'ESP32-C3, et son rapport cyclique est régulé par la tension existant sur le rail de 200 V. Pour cela un diviseur la ramène à une valeur plus digeste pour l'ADC interne, en utilisant R17 à R21. Outre le 400 V, il nous faut aussi besoin de tensions plus basses, qui sont produites en divisant le 200 V à l'aide de quatre résistances. Notez que le schéma utilise plusieurs résistances en série là où une seule suffirait.



C'était pour utiliser des modèles 0603, qui ne résistent qu'à 75 V environ. En les mettant en série, la chute de tension est plus faible sur chacune d'elles, et aucune étincelle ne survient. Si l'arc électrique HT peut servir d'indicateur de vitesse de décharge de la self, il n'est pas reproductible et dégage une odeur envahissante. Côté dekatron, comme illustré par le schéma (figure 3), les 400 V sont injectés à l'anode via une résistance de 720 k Ω (4 x 180 k Ω). Elle limite le courant à 280 μ A pour rester en deçà des 310 μ A, valeur maximale selon la fiche technique [3]. Les signaux G1 et G2 sont décalés par deux transistors HT NPN et, via un décalage de niveau par transistor PNP, l'une des cathodes non partagées est reliée à l'ESP32-C3 afin qu'il sache si la lueur passe par cette cathode.

Le module ESP32-C3

L'ESP32-C3 lui-même apparaît au milieu du schéma (figure 3). Le module ESP32-C3-WROOM02 utilisé inclut la plupart du matériel nécessaire : la flash, les réseaux d'adaptation RF et l'antenne sont tous là. Seul matériel externe : deux LED, un poussoir et un réseau RC qui réinitialise le μ processeur à la mise sous tension. Si, comme moi, vous utilisez KiCad pour concevoir vos circuits imprimés, sachez qu'Espressif met à disposition gratuitement ses symboles et empreintes de tous ses modules et puces [4].

Enfin, les parties supérieure et droite du schéma, montrent la section d'alimentation. Le connecteur USB-C a deux fonctions : en y branchant un chargeur USB-PD tout sera alimenté, mais vous pouvez également le brancher à votre ordinateur. Dans ce cas, le dekatron ne s'allumera pas, car l'alimentation de 5 V disponible est insuffisante, mais cela permet de reprogrammer le code dans la flash ESP32-C3. Pour la négociation USB-PD, un contrôleur USB-C FUSB302 est utilisé. L'ESP32-C3 peut lui parler via I²C pour qu'il communique avec l'alimentation. La conversion de toute tension entrante en tension utilisable par l'ESP32-C3, est confiée à un convertisseur buck synchrone, le MT2492 d'Aerosemi. Il fonctionne jusqu'à 16 V, et comme il s'agit d'une puce à découpage, il reste bien frais en produisant le rail de 3,3 V sur lequel fonctionne l'ESP32-C3.

La carte imprimée

Une photo illustre l'agencement du circuit (v. figure 4). J'ai opté pour un format de même largeur que le dekatron, afin de le dissimuler dessous. Le routage de cette carte est un peu plus délicat qu'il n'y paraît : en HT, il faut maintenir un écart inter-pistes suffisant pour garantir qu'aucun arc ne se produise au risque de causer un incendie. Les composants HT sont aussi plus grands. J'ai tout de même pu tout faire tenir sur une carte double face et même avec un vaste plan de masse. J'ai décidé de faire faire le circuit imprimé avec un masque de soudure noir

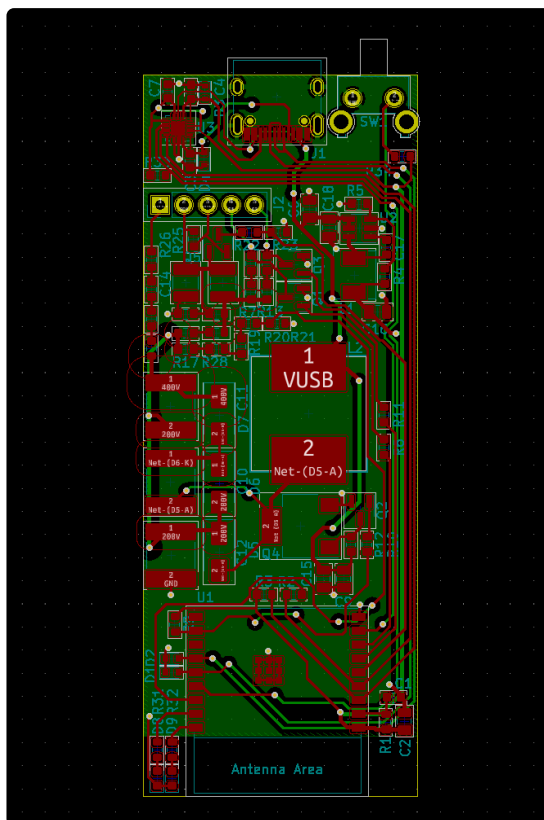


Figure 4. Agencement de la carte d'interface dekatron basée sur l'ESP32-C3.

pour qu'il ne ressorte pas et ne vole pas la vedette au dekatron, (v. figure 5).

Mise en œuvre du logiciel

Une fois tout construit, j'ai abordé le logiciel. Il doit faire plusieurs choses. Tout d'abord, il doit vérifier que le reste du matériel reçoit la tension voulue et, à cet effet implémenter une pile USB-PD pour communiquer avec l'alimentation connectée. J'ai trouvé une bibliothèque qui fonctionne bien [5]. Une plage de tension assez large peut alimenter le matériel lui-même. Le logiciel essaie d'abord de demander 12 V à l'alimentation, sinon, il acceptera

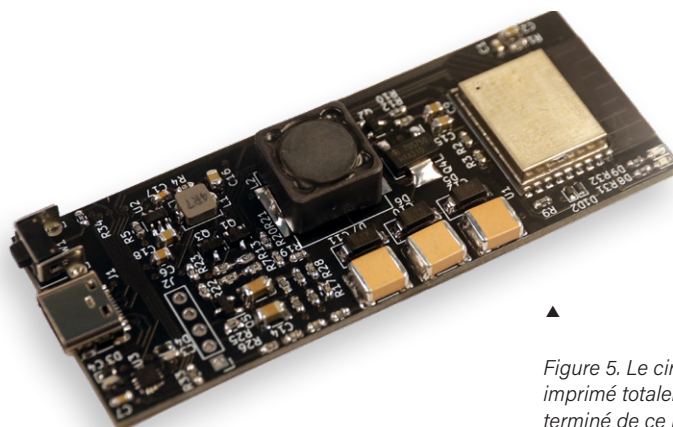


Figure 5. Le circuit imprimé totalement terminé de ce projet.

Figure 6. Sélection du point d'accès correct sur la page ESP32 Wi-Fi Manager.

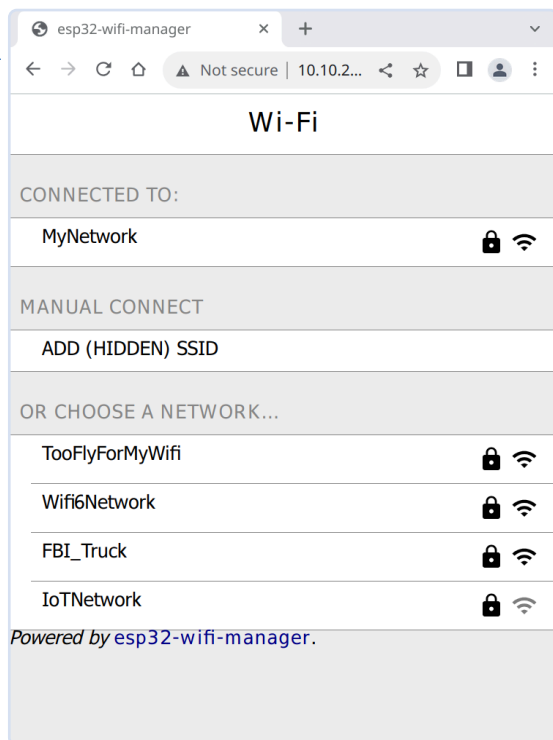


Figure 7. Saisie des paramètres de configuration de la connexion.

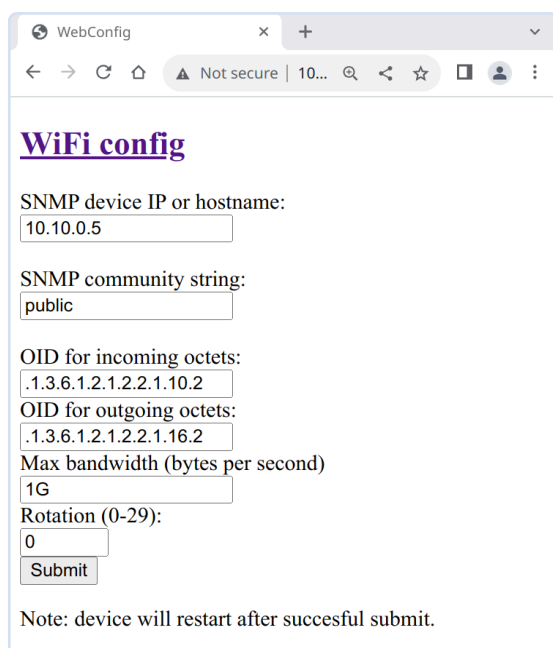


Figure 8. Boîtiers imprimés en 3D pour le projet : brut (à droite) et fini (à gauche) avec un vernis brillant approprié.



également 9 ou 15 V. Pour que le dekatron obtienne aussi sa HT, il mesure la tension instantanée sur le rail de 200 V et ajuste le cycle de travail MLI du convertisseur boost en conséquence.

En second lieu, il doit déterminer la BP Internet instantanée. Ma connexion internet est en fait distribuée par un commutateur Ethernet qui conserve des statistiques sur toutes les données qui y transitent. Un protocole appelé Simple Network Management Protocol (SNMP) obtient facilement ces données. Ce protocole sert à surveiller les infrastructures informatiques comme les routeurs, commutateurs, serveurs, etc. Une demande de statistiques d'un port réseau ne nécessite que quelques paquets UDP. Je pourrais donc le faire plusieurs fois par seconde pour avoir une idée en temps réel du trafic utilisé. Ces données servent à définir la vitesse des impulsions G1 et G2, et donc un téléchargement plus rapide entraîne une rotation plus rapide dans le sens horaire. Si à l'inverse, quelqu'un envoie des données, le dekatron tournera également, mais dans le sens antihoraire.

La dernière fonction que le firmware doit fournir est un genre d'interface de configuration. Je n'aime pas coder en dur la configuration dans mon micrologiciel, il me fallait donc un moyen d'entrer les identifiants Wifi ainsi que l'IP et les données SNMP du commutateur. La saisie des informations Wifi s'appelle le *provisioning*, et l'ESP-IDF dispose d'une solution adaptée, via une application smartphone qui rationalise le processus. Cependant, j'ai opté ici pour ce que l'on appelle une *gestionnaire Wifi*. S'il n'est pas configuré, il crée un point d'accès Wifi reconnaissable par un ordinateur portable ou un téléphone. Une fois connecté, il ouvre une page web permettant la sélection du point d'accès Wifi et l'entrée du mot de passe (figure 6). Le gestionnaire Wifi ayant besoin d'un serveur web, il est facile de l'étendre pour ajouter une page de configuration des paramètres SNMP (figure 7). Pour faciliter le débogage, elle accueille aussi quelques données, par ex., la tension réelle négociée avec l'alimentation.

Conception du boîtier

Logiciel et matériel étant terminés, il me restait une dernière chose à faire : un boîtier. Ici, un boîtier était nécessaire pour des raisons décoratives mais aussi car le circuit imprimé abrite des tensions qui, bien que non mortelles, piquent très fort. Je l'ai fait faire : j'ai demandé à l'un de nos designers industriels résidents s'il pouvait me concevoir un boîtier et l'imprimer sur notre imprimante 3D SLA. Il eu la gentillesse d'accepter (merci, Kaijie !) et imprima un boîtier en résine transparente, pour pouvoir toujours voir l'intérieur du dekatron.

Du moins, c'était l'idée. Il s'avère que le mode d'impression du boîtier le rend opaque plutôt que transparent. Il y a heureusement un moyen simple de résoudre ce problème : il suffit de recouvrir l'objet d'une fine couche de résine qui durcit aux UV. Le résultat de l'exposition aux



UV est présenté à côté d'un boîtier non traité (**figure 8**). J'aurais peut-être pu appliquer une couche plus uniforme, mais, honnêtement, l'esthétique "artisanale" de mon boîtier ne me dérange pas. Lorsque le dekatron est installé, (**figure 9**), il attire automatiquement l'attention sur l'avant de l'appareil. Il est évident que cela est encore mieux si l'appareil est allumé : même si on ne sait pas exactement ce qu'il affiche, le point lumineux qui tourne en rond attire tous les regards (**figure 10**).

Avant de partir

J'ai finalement réalisé quelque chose qui, je l'espère, est une rétronique partielle. Peut-être qu'un jour, dans le futur, l'ensemble du projet méritera cette appellation. Avant cela, Elektor et Espressif coopéreront peut-être à nouveau pour un article de magazine, et peut-être - juste peut-être - que je serai autorisé à sortir cet appareil de son étagère et à parler avec lyrisme de tout ce qui a conduit à son existence. En attendant, il me servira de point de repère lors de mes téléchargements importants. L'ensemble du projet est open-source. Vous pouvez pour votre usage, en réutiliser des éléments matériels ou logiciels et, si vous possédez un dekatron, reproduire cette réalisation, Vous trouverez le micrologiciel ainsi que le dessin du PCB et les fichiers de conception 3D du boîtier sur GitHub [6] ◀

230560-04



Figure 9. Le projet complet dans son boîtier définitif.

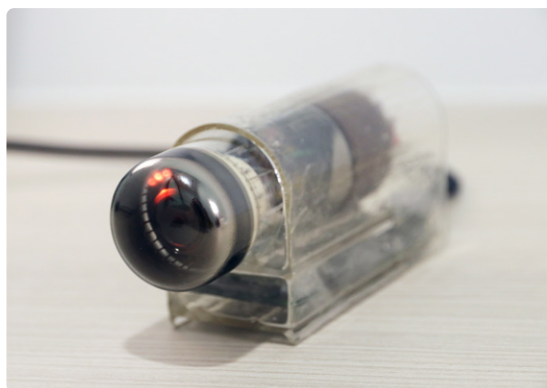


Figure 10. Le prototype au travail : irrésistiblement accrocheur !



Questions ou commentaires ?

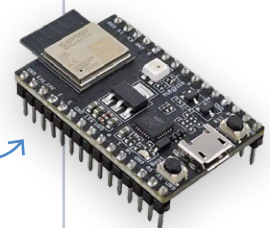
Envoyez un courriel à l'auteur (jeroen@spritesmods.com) ou contactez Elektor (redaction@elektor.fr).

À propos de l'auteur

Jeroen Domburg est Senior Software and Technical Marketing Manager chez Espressif Systems. Avec plus de 20 ans d'expérience dans le domaine de l'embarqué, il est participe au processus de conception logicielle et matérielle des SoC d'Espressif. Pendant son temps libre, il aime bricoler avec l'électronique pour réaliser des appareils qui ont une utilité pratique ou pas !

Produits

- > Anycubic Photon Mono X - Imprimante 3D SLA à résine UV
www.elektor.fr/19831
- > ESP32-C3-DevKitM-1
www.elektor.fr/20324
- > Dogan Ibrahim, *The Complete ESP32 Projects Guide* (Elektor, 2019)
www.elektor.fr/18860



LIENS

- [1] L'ordinateur Harwell Dekatron (WITCH) : <https://tnmoc.org/witch>
- [2] Circuit du convertisseur élévateur dekatron : <https://threeneurons.wordpress.com/dekatron-stuff/dekatron-do-hickie-kit>
- [3] Fiche technique GC10/4B : <http://r-type.org/pdfs/gc10-4b.pdf>
- [4] Bibliothèques KiCad pour les puces et modules Espressif : <https://github.com/espressif/kicad-libraries>
- [5] Jeu de pilotes USB-PD pour le FUSB302 : <https://github.com/Ralim/usb-pd>
- [6] Dépôt du projet : https://github.com/Spritetm/elek_dekatron

la révolution de la maison intelligente

Paulus Schoutsen parle de Home Assistant, d'ESPHome, et d'autres choses encore



Paulus Schoutsen. (Source: P. Schoutsen / Midjourney)

Questions de Saad Imtiaz (Elektor)

Paulus Schoutsen, le créateur de Home Assistant et ESPHome, parle de son parcours dans le monde de la domotique, l'IdO, et plus encore. Découvrez comment le micrologiciel d'ESPHome est facile à utiliser et comment il peut changer la donne pour les appareils intelligents à monter soi-même.

Elektor : commençons par votre parcours. Avez-vous toujours été intéressé par l'électronique et l'ingénierie ? Qu'est-ce qui vous a poussé à étudier l'informatique à l'Université de Twente ?

Schoutsen : j'ai étudié la Technologie de l'Information en Entreprise à l'Université de Twente. Le programme comprend des cours de programmation et de base de données mais il se focalise sur les systèmes d'information : quelle information circule au travers des processus en entreprise et comment vous pouvez la capturer, la traiter et l'analyser. Dans un sens, c'était un précurseur dans mon parcours dans la domotique. J'ai toujours été très intéressé par la programmation mais pas vraiment par l'électronique. C'est arrivé par accident. Lorsque Philips a sorti son ampoule connectée Hue en 2012, je l'ai tout de suite achetée. J'ai réalisé qu'elle disposait d'une API locale et j'ai écrit un script pour contrôler la lampe depuis mon ordinateur. Ce

script s'est développé et le 17 septembre 2013, j'ai envoyé la première version sur GitHub : Home Assistant était né. Cette plateforme open-source pour la domotique, centrée sur le contrôle local et la confidentialité, est devenue le deuxième projet open-source le plus actif sur GitHub [1].

Au fur et à mesure que Home Assistant s'est développé, j'ai joué avec l'électronique mais il y avait un gros problème : connecter l'électronique à l'internet était trop cher. L'ajout d'un shield Wi-Fi Arduino coûtait 70 \$. C'est alors qu'est arrivée la puce qui allait tout changer : l'ESP8266 d'Espressif.

L'ESP8266 était une puce à 3,50 \$ qui pouvait exécuter un code compatible avec Arduino et pouvait se connecter à un réseau sans fil. Grâce à cette puce abordable, il était enfin financièrement possible de bricoler des appareils électroniques pouvant s'intégrer à votre maison intelligente.

Pendant que l'ESP8266 s'imposait lentement dans le domaine du bricolage, j'ai fondé Nabu Casa. Nabu Casa existe pour rendre le développement de Home Assistant durable. Home Assistant est un grand projet open-source et la gestion d'un tel projet nécessite beaucoup d'administration, de processus, de structure et de maintenance – plus que ce que n'importe qui peut faire pendant son temps libre. Les employés à plein

À propos de Paulus Schoutsen

Paulus Schoutsen est le fondateur de Home Assistant, l'un des plus grands projets open-source sur GitHub, et de Nabu Casa, la société qui assure la viabilité à long terme de Home Assistant. Son travail tourne autour de la construction de Home Assistant : sa vision d'une maison intelligente qui offre confidentialité, choix et durabilité.

temps de Nabu Casa s'en chargent, financés par les abonnements aux services supplémentaires que nous offrons aux utilisateurs. Nabu Casa n'a pas d'investisseurs et n'existe que pour satisfaire ses utilisateurs. Au fur et à mesure que Home Assistant s'est développé, nous avons défini notre vision d'Open Home [2]. Il s'agit d'une vision de la maison intelligente basée sur trois valeurs : la confidentialité, le choix et la durabilité. À partir de cette vision, nous nous sommes rendu compte que si vous voulez que votre maison intelligente soit locale et privée, cela doit s'appliquer à la fois à la plateforme de la maison intelligente (c'est-à-dire Home Assistant) et aux appareils qui y sont connectés. L'un des principaux développeurs de Home Assistant est Otto Winter. Il s'intéressait à l'électronique mais il a constaté que les premiers projets ESP8266 étaient trop compliqués au niveau technique et nécessitaient une intégration plus facile avec Home Assistant. Il a décidé de résoudre ce problème et, le 21 janvier 2018, il a publié Esphomelib [3]. ESPHome est devenu de plus en plus populaire mais Otto n'avait plus le temps de le gérer. Comme ESPHome est une base essentielle pour les gens fabriquant des appareils correspondant à nos valeurs Open Home, nous voulions le soutenir. Le 21 mars 2018, Nabu Casa a acquis ESPHome [4]. Aujourd'hui, nous avons deux développeurs dédiés à plein temps qui travaillent sur ESPHome, financés entièrement par les personnes qui s'abonnent à Home Assistant Cloud de Nabu Casa.

Elektor : pouvez-vous décrire brièvement ESPHome ? En quoi se différencie-t-il des autres micrologiciels IdO et solutions domotiques ?

Schoutsen : ESPHome est un micrologiciel pour les cartes ESP8266, ESP32 et Raspberry Pi Pico qui permet aux utilisateurs de créer et de gérer facilement des appareils domestiques intelligents. ESPHome vous débarrasse de toute la tracasserie liée à l'écriture du logiciel et vous permet de vous concentrer sur la fabrication de votre matériel.

Pour vous donner un exemple de la facilité avec laquelle il est possible de faire quelque chose avec ESPHome, il suffit de prendre un capteur de température, de le connecter à votre ESP32, d'indiquer à ESPHome dans un fichier de configuration à quelle broche le capteur est connecté et de cliquer sur *Install*. ESPHome va alors générer tout le firmware nécessaire et l'installer sur votre ESP32. L'appareil démarrera, se connectera à votre réseau Wi-Fi et votre température sera disponible dans Home Assistant. Le tour est joué. La première fois que vous installez ESPHome, vous devez connecter la carte à votre ordinateur. Les mises à jour ultérieures se font sans fil. Vous souhaitez connecter un deuxième capteur à la même carte ? Il suffit de mettre à jour le fichier de configuration et de cliquer sur *Install*. Quel que soit l'endroit de la maison où se trouve votre appareil, les mises à jour sont transparentes.

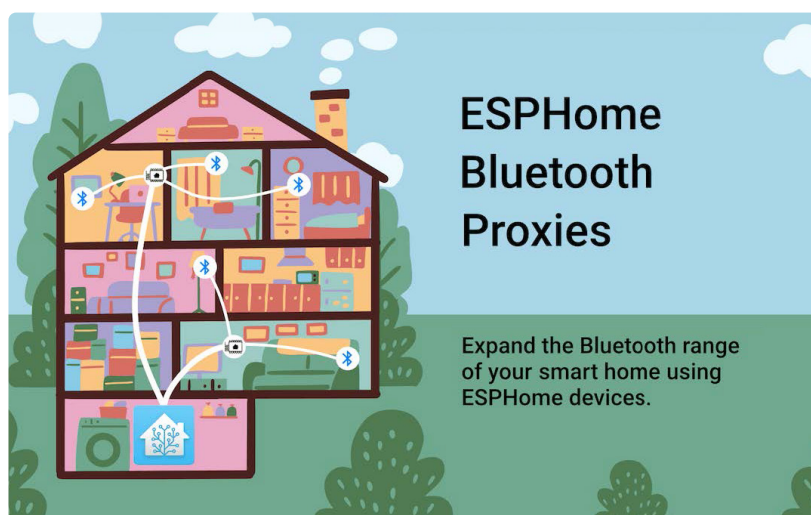
Une caractéristique supplémentaire de ESPHome est que les appareils peuvent également agir en tant que proxy Bluetooth pour Home Assistant. Cela permet à Home Assistant d'étendre sa portée Bluetooth en utilisant des ESP32s utilisant ESPHome pour écouter les paquets BLE et établir des connexions directes pour le contrôle. Pour rendre cela encore plus utile pour les bricoleurs, nous avons également introduit BTHome [5], un protocole BLE pour envoyer des données que ces proxys Bluetooth récupèrent (**figure 1**).

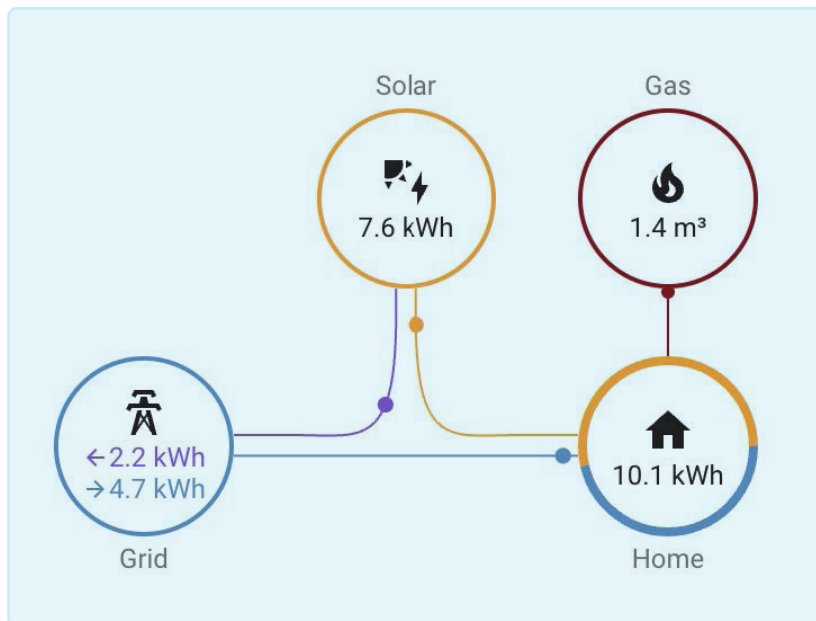
Elektor : ESPHome jouit d'une grande popularité au sein de la communauté des bricoleurs de maisons intelligentes. Qu'est-ce qui, selon vous, a contribué à sa popularité ?

Schoutsen : la facilité d'utilisation. Nous nous efforçons sans relâche de faciliter l'utilisation du matériel. Avec ESPHome, vous ne serez jamais confronté à des erreurs de compilation C++ ou à des sujets MQTT mal configurés. De plus, il est facile de partager vos fichiers de configuration avec d'autres utilisateurs, ce qui permet aux débutants de construire immédiatement des appareils fonctionnels.

Nous avons également développé un outil pour faciliter le partage des configurations : ESP Web Tools. Il s'agit d'un installateur web pour les cartes ESP8266 et ESP32 qui permet aux utilisateurs d'installer en un clic le micrologiciel sur leurs appareils directement à partir de leur navigateur. Désormais, les utilisateurs peuvent acheter des appareils prêts à l'emploi et y installer ESPHome. Par exemple, nous permettons aux utilisateurs de transformer leur kit de développement M5Stack Atom Echo en un assistant vocal pour Home Assistant directement à partir de leur navigateur [6]. ESP Web Tools a été d'une grande aide pour partager nos projets basés sur ESPHome, mais nous pensons que ce type de technologie devrait être partagé avec tout le monde. Ainsi, aujourd'hui, il alimente

Figure 1. Home Assistant peut étendre sa portée Bluetooth en utilisant des périphériques proxy Bluetooth alimentés par ESPHome. (Source : Paulus Schoutsen)





▲
 Figure 2. Tableau de bord d'énergie dans Home Assistant pour un accès facile à la consommation d'énergie. En outre, il fournit des indicateurs permettant d'évaluer la fiabilité du réseau. (Source : Paulus Schoutsen)

également les installateurs de Tasmota, WLED et beaucoup d'autres firmwares.

Elektor : comment ESPHome peut-il faciliter l'intégration de nombreux capteurs et appareils dans un environnement domotique, en particulier pour les personnes n'ayant pas une grande expérience du codage ?

Schoutsen : ce qui est formidable, c'est que pour utiliser ESPHome, il n'est pas nécessaire de savoir programmer. Le problème avec l'électronique, c'est qu'il faut apprendre trois choses en même temps :

Comment construire du matériel, interagir avec ce matériel et intégrer cette interaction dans un autre système ? Avec ESPHome, l'utilisateur peut se concentrer sur la première étape : expérimenter avec le matériel. Nous nous occupons du reste. Nous permettons aux utilisateurs de se concentrer sur la partie amusante de la construction de leur matériel.

Elektor : l'installation basée sur YAML est l'une des principales caractéristiques d'ESPHome. Pouvez-vous nous expliquer pourquoi vous avez choisi cette méthode et en quoi elle est bénéfique pour les utilisateurs ?

Schoutsen : lorsque ESPHome a été créé, il a imité l'architecture et le format de configuration de Home Assistant. Il correspond très bien au format d'ESPHome. Une intégration représente chaque capteur connecté. Vous pouvez configurer cette intégration pour régler les capteurs et elle sera incluse lors de la prochaine mise à jour de votre appareil.

Elektor : la communauté ESPHome a contribué à élargir la liste des appareils et composants compatibles. Dans quelle mesure l'implication de la communauté est-elle essentielle à la réussite du projet et comment gérez-vous concrètement cette collaboration ?

Schoutsen : pour la domotique open-source, la communauté est indispensable. Il faut du temps pour

comprendre le fonctionnement de chaque capteur, trouver les pilotes et les intégrer à ESPHome. Nos mainteneurs à plein temps travaillent avec la communauté pour examiner leurs contributions et les faire fusionner.

Cette année, les principaux sujets d'intérêt pour ESPHome ont été l'assistant vocal, le proxy Bluetooth et les écrans E-ink :

- L'assistant vocal permet de transformer un appareil ESPHome en un assistant vocal qui contrôle Home Assistant. ESPHome est chargé de capturer l'audio, tandis que Home Assistant traite la parole et agit en conséquence. Vous pouvez installer un assistant vocal à partir de votre navigateur [6].
- Le proxy Bluetooth permet de transformer n'importe quel appareil ESP32 en proxy Bluetooth pour Home Assistant. Home Assistant utilise le BLE de l'ESP32 pour écouter les paquets et se connecter aux appareils pour les contrôler. Vous pouvez installer un proxy Bluetooth à partir de votre navigateur [7].
- Les écrans E-ink ont le vent en poupe. Nous avons ajouté la prise en charge d'un plus grand nombre d'entre eux, de sorte que la création des tableaux de bord les plus fantastiques pour votre maison intelligente soit plus accessible.

Elektor : la sécurité de l'IdO et de la domotique est une préoccupation majeure. Quelles mesures de sécurité et de confidentialité ESPHome prend-il pour protéger les données des utilisateurs et des appareils connectés ?

Schoutsen : par défaut, chaque nouvel appareil ESPHome est crypté à l'aide du protocole Noise. Il s'agit d'un protocole rapide et efficace que Wireguard et WhatsApp utilisent également. Cela signifie que personne ne peut espionner les données d'ESPHome sur le réseau.

Si vous utilisez les appareils ESPHome avec Home Assistant, votre maison intelligente sera entièrement open source et fonctionnera de manière entièrement locale et cryptée. Aucune donnée ne sera partagée avec qui que ce soit et votre maison intelligente pourra même fonctionner si elle n'est pas connectée à Internet.

Lorsque ESPHome contient une mise à jour de sécurité, il peut recompiler votre fichier de configuration et mettre à jour votre appareil sans fil pour s'assurer qu'il utilise la dernière version. Pour vous aider à sécuriser votre maison intelligente, il est possible de mettre à jour vos appareils ESPHome depuis Home Assistant.

Elektor : pourriez-vous nous présenter des cas d'utilisation ou des projets concrets dans lesquels Home



ESPHome est un micrologiciel pour les cartes ESP8266, ESP32 et Raspberry Pi Pico qui permet aux utilisateurs de créer et de maintenir facilement des appareils domestiques intelligents.

Assistant et ESPHome ont fait une différence significative ou originale ?

Schoutsen : l'homme émet du CO₂ dans l'atmosphère, ce qui entraîne des changements climatiques et un réchauffement de la planète. L'une des façons dont nous pouvons tous contribuer est de veiller à ce que notre empreinte énergétique individuelle soit aussi faible que possible. Nos maisons représentent une part importante dans notre consommation d'énergie. En 2021, nous avons introduit la gestion de l'énergie domestique dans Home Assistant [8]. Il permet aux utilisateurs de surveiller leur consommation d'énergie, de passer à une énergie durable et d'économiser de l'argent (**figure 2**).

L'information la plus cruciale est de connaître la quantité d'énergie qui circule entre le réseau électrique et votre domicile. L'accès en temps réel à ces informations permet aux utilisateurs de décider quand ajuster leur consommation d'énergie en chargeant leurs vélos électriques, en faisant la lessive ou en réduisant le chauffage.

L'obtention de ces informations à partir des compteurs d'électricité est une tâche quelque peu standardisée mais l'adoption des différentes normes se fait généralement par pays ou par région. En collaboration avec notre communauté, nous avons construit des appareils open-source basés sur ESPHome pour prendre en charge différentes normes. Par exemple, le SlimmeLezer de Marcel Zuidwijk [9] se connecte directement aux ports P1 (utilisés aux Pays-Bas et dans certains autres pays de l'UE) pour accéder à la consommation d'énergie en temps réel et à la consommation cumulée d'énergie et de gaz. D'autres compteurs d'électricité sont équipés d'une diode électroluminescente clignotante au lieu d'un port. Chaque impulsion représente une unité d'énergie consommée. Avec l'assistant domestique Glow de Klaas Schoute [10], ces impulsions LED peuvent être comptées pour obtenir une consommation d'énergie précise et Home Assistant peut suivre votre consommation au fil du temps (**figure 3**).

Elektor : quels sont les progrès ou les améliorations que les utilisateurs d'ESPHome peuvent attendre de l'évolution de l'écosystème IdO ? Envisagez-vous d'associer ESPHome et Home Assistant ?

Schoutsen : l'une des valeurs d'Open Home est le choix. Ainsi, tout en veillant à ce que l'intégration entre Home Assistant et ESPHome soit la meilleure possible, nous ne voulons pas empêcher les gens de contrôler les appareils ESPHome via d'autres systèmes. Il existe d'autres possibilités d'intégration renforcée. Par exemple, lorsqu'un utilisateur achète un appareil équipé d'un micrologiciel ESPHome, il doit utiliser le tableau de bord ESPHome pour adopter l'appareil et pouvoir installer les mises à jour. Nous voulons rationaliser ce processus afin que les utilisateurs n'aient pas besoin d'apprendre tout cela pour rester à jour. Cela fait partie de nos efforts continus pour améliorer l'expérience des produits dans le cadre de notre programme « Works with ESPHome ».

Le programme « Works with ESPHome » permet aux créateurs de porter notre badge si leurs appareils remplissent un certain nombre d'exigences, par exemple en activant des fonctions qui facilitent l'intégration et en permettant aux utilisateurs de personnaliser la configuration de l'appareil.

Elektor : quels conseils donneriez-vous aux débutants désireux de s'initier à ESPHome et de se lancer dans leurs propres projets domotiques ? Avez-vous des ressources ou des bonnes pratiques à partager ?

Schoutsen : commencez par une simple carte de développement ESP32 et un capteur de température. Faites-le fonctionner, configurez-le dans Home Assistant et commencez à jouer avec la configuration de

Figure 3. L'assistant domestique Glow de Klaas Schoute. (Source : Paulus Schoutsen)



l'appareil ESPHome. Vous verrez les changements se refléter instantanément dans Home Assistant lorsque vous mettrez à jour votre appareil.

La façon la plus simple de commencer à utiliser ESPHome est de l'installer à partir du magasin de modules complémentaires de Home Assistant [11]. L'installation se fait en un clic. Cliquez sur *Ajouter un appareil* et un assistant vous guidera dans l'installation de votre premier appareil. Bonne automatisation !

Elektor : ESPHome est connu pour son approche conviviale mais quelles actions ou ressources recommanderiez-vous aux novices pour les aider à démarrer leur premier projet ESPHome ?

Schoutsen : j'irais sur YouTube. Il existe une grande variété de vidéos de démarrage.

Elektor : existe-t-il une idée ou une tentative continue de fournir une documentation plus détaillée ou des cours conçus principalement pour les débutants dans l'esprit de rendre la domotique plus accessible ?

Schoutsen : nous y réfléchissons pour 2024. Nous aimerions avoir un kit de démarrage facile pour ESPHome.

Elektor : pouvez-vous nous donner des informations sur les fonctionnalités ou les mises à jour prévues qui rendront ESPHome encore plus convivial ? Est-il prévu d'améliorer l'interface utilisateur ?

Schoutsen : lorsque vous créez un appareil ESPHome, vous intégrez votre réseau Wi-Fi et votre mot de passe dans le micrologiciel. Ainsi, l'appareil trouvera automatiquement votre réseau et s'y connectera. C'est très bien, sauf si vous voulez le partager avec vos amis ou vendre vos créations. Pour que cela fonctionne, nous avons développé Improv Wi-Fi [12]. Il s'agit d'une norme qui permet de configurer des appareils utilisant BLE ou Serial.

L'une des améliorations que nous prévoyons est d'ajouter la prise en charge de l'Improv Wi-Fi over BLE à Home Assistant. Ainsi, lorsque vous branchez un appareil acheté, Home Assistant peut vous guider tout au long de l'expérience d'accueil et faire en sorte que l'appareil soit connecté et configuré.

Elektor : ESPHome fonctionne déjà avec une grande variété de capteurs et d'appareils. Est-il prévu d'ajouter d'autres capteurs et composants à l'avenir et si oui,

quels types de capteurs les utilisateurs peuvent-ils s'attendre à voir ?

Schoutsen : nous ne nous arrêterons jamais ! Nous avons récemment constaté une augmentation du nombre de capteurs à ondes millimétriques. Ils offrent une nouvelle façon de détecter une présence dans une pièce qui fonctionne même lorsque vous restez assis sans bouger.

Nous souhaitons également étudier la possibilité d'utiliser des capteurs de qualité de l'air moins coûteux. Si l'air de votre chambre est mauvais, cela peut avoir un impact sur votre réflexion et vos performances. Il serait intéressant de débloquer des capteurs bon marché basés sur ESPHome pour aider les gens à vivre plus sainement.

Elektor : les utilisateurs peuvent rencontrer des difficultés lors de l'intégration de capteurs ou d'appareils personnalisés. Est-il prévu de simplifier encore ce processus et de permettre aux bricoleurs de connecter plus facilement leurs propres capteurs à ESPHome ?

Schoutsen : Toute configuration ESPHome génère un projet C++ qui est compilé et installé sur votre appareil. Si un développeur de capteurs personnalisés crée son propre protocole, il doit écrire du C++ pour l'intégrer dans ESPHome. Nous avons des protocoles comme BTHome [5] pour envoyer des données, mais il s'agit de données destinées à Home Assistant.

Elektor : est-il prévu que Home Assistant devienne également une solution basée sur le cloud où l'on peut utiliser un capteur ESP32 pour envoyer des données et les visualiser, sans avoir à configurer et à

exécuter Home Assistant sur le serveur local ?

Schoutsen : non. Votre maison intelligente doit fonctionner localement et conserver vos données à cet endroit. Nous avons lancé le Home Assistant Green [13] pour faciliter l'utilisation de Home Assistant. Il fonctionne avec Home Assistant dès sa sortie de l'emballage et constitue le moyen le plus simple pour les nouveaux utilisateurs de commencer à utiliser Home Assistant. Il est vendu au prix de 99 \$.

Elektor : ESPHome et Home Assistant sont-ils impliqués dans des partenariats ou des collaborations avec des fabricants de matériel ou dans d'autres efforts visant à améliorer la compatibilité des appareils et l'expérience des utilisateurs ?



Avec ESPHome, l'utilisateur peut se concentrer sur la première étape : l'expérimentation avec le matériel. Nous nous occupons du reste. Nous permettons aux utilisateurs de se concentrer sur la partie amusante de la construction de leur matériel.

Schoutsen : oui. Nous avons le programme *Made for ESPHome* [14] pour travailler avec les créateurs qui vendent des appareils basés sur ESPHome afin d'obtenir la meilleure expérience possible. Ce programme comporte un ensemble d'exigences visant à garantir la personnalisation et l'ouverture des appareils. Nous avons également le programme *Works with Home Assistant* [15]. Ce programme nous permet de tester la compatibilité des appareils et de collaborer avec le vendeur pour résoudre les problèmes éventuels.

Elektor : comment Home Assistant parvient-il à introduire de nouvelles fonctionnalités tout en soutenant les fonctionnalités existantes afin de garantir une expérience fiable et conviviale pour tous les utilisateurs au fur et à mesure de son évolution ?

Schoutsen : il faut beaucoup de monde. L'année dernière, Home Assistant était le deuxième projet open-source le plus actif sur GitHub. Nous avons reçu la contribution de 13 200 personnes. Toutes ces personnes s'efforcent de maintenir la compatibilité de Home Assistant avec l'évolution de l'espace domestique intelligent en prenant en charge de nouveaux appareils ou de nouvelles fonctionnalités. La puissance de Home Assistant réside dans la communauté qui le soutient. La communauté en fait la meilleure plateforme et nous offre les meilleures idées pour rendre nos maisons intelligentes.


Elektor : de nombreux nouveaux venus dans le domaine de la domotique et de l'IdO peuvent être préoccupés de la courbe d'apprentissage. Quels conseils donneriez-vous aux nouveaux utilisateurs pour les aider à surmonter les obstacles et les frustrations ?

Schoutsen : ne commencez pas par quelque chose de sophistiqué : restez dans le chemin standard pour l'instant. Il existe d'autres moyens d'installer Home Assistant, mais tenez-vous en au système d'exploitation Home Assistant. Il en va de même pour ESPHome : tenez-vous en aux cartes ESP32 standard au début et n'essayez pas d'être sophistiqué avec des cartes moins courantes comme la ESP32-S3. Cela ne fait que compliquer les choses.

Elektor : comment sont gérés les microprogrammes et les correctifs de bogues ? Est-il prévu d'automatiser ou de rationaliser ce processus afin que les utilisateurs disposent toujours des versions les plus récentes et les plus stables ?


Schoutsen : la sécurité est une priorité absolue dans notre vision d'Open Home ce qui en fait une priorité absolue pour nos projets tels que Home Assistant et ESPHome. Nous avons fait en sorte qu'il soit très facile de tout mettre à jour : Home Assistant permet aux utilisateurs de mettre à jour toutes les parties de Home Assistant d'un simple clic à partir de l'interface. Cela inclut la mise à jour des appareils ESPHome par voie sans fil lorsqu'une nouvelle version d'ESPHome est disponible. ◀

VF : Chris Elsass — 230594-04



Produits

- **Espressif ESP32-C3-DevKitM-1**
<https://www.elektor.fr/20324>
- **Espressif ESP32-DevKitC-32E**
<https://www.elektor.fr/20518>



LIENS

- [1] Principaux projets open-source : <https://octoverse.github.com/2022/state-of-open-source>
- [2] Open Home : <https://home-assistant.io/blog/2021/12/23/the-open-home>
- [3] Esphomelib : <https://community.home-assistant.io/t/esphomelib-library-to-greatly-simplify-home-assistant-integration-with-esp32/40245>
- [4] Acquisition d'ESPHome : <https://home-assistant.io/blog/2021/03/18/nabu-casa-has-acquired-esphome>
- [5] Protocole BTHome : <https://bthome.io>
- [6] Assistant vocal pour Home Assistant : https://home-assistant.io/voice_control/thirteen-usd-voice-remote
- [7] Installation du proxy Bluetooth : <https://esphome.github.io/bluetooth-proxies>
- [8] Gestion de l'énergie dans Home Assistant : <https://home-assistant.io/blog/2021/08/04/home-energy-management>
- [9] SlimmeLezer by Marcel Zuidwijk : <https://slimmelezer.nl>
- [10] Home Assistant Glow par Klaas Schoute : <https://github.com/klaasnicolaas/home-assistant-glow>
- [11] Magasin de modules complémentaires dans Home Assistant : https://my.home-assistant.io/redirect/_change/?redirect=supervisor_addon%2F%3Faddon%3D5c53de3b_esphome
- [12] Improv Wi-Fi : <https://improv-wifi.com>
- [13] Home Assistant Green : <https://home-assistant.io/green>
- [14] Fabriqué pour ESPHome : https://esphome.io/guides/made_for_esphome.html
- [15] Works avec Home Assistant : <https://partner.home-assistant.io>

programmer le micrologiciel !

flashez votre ESP32

Pedro Minatel, Espressif

Si vous êtes débutant en développement embarqué, vous vous familiariserez avec les termes « charger », « flasher » et « programmer » le micrologiciel sur un microcontrôleur ou une mémoire flash - il s'agit d'une étape fondamentale pour la programmation de l'application sur l'appareil. Dans cet article, nous examinons plus en détail comment cela peut être fait pour les contrôleurs ESP32 pendant la phase de développement et de production de manière efficace et sécurisée.

Le concept de « programmation du micrologiciel » était souvent utilisé pour les mémoires programmables une seule fois (OTP), telles que les mémoires mortes programmables (PROM), sur lesquelles on ne peut écrire des données qu'une seule fois, de sorte que le terme « burn » implique l'irréversibilité. Cela s'explique par le fait qu'une connexion physique est « brûlée », comme un fusible, entre deux points, interrompant le flux de courant et faisant passer ce « bit » de 1 à 0.

En revanche, une application ESP32 est stockée dans une mémoire flash externe, ce qui permet de flasher le micrologiciel ou de réécrire les données des milliers de fois.

Flasher l'ESP32

Sur l'ESP32, le processus de flashage est plus facile par rapport à la plupart des autres microcontrôleurs, principalement parce que l'ESP32 peut être flashé via UART. Il n'est pas nécessaire d'utiliser un programmeur spécial ou un JTAG (bien que le port JTAG soit présent sur l'ESP32 pour le débogage) - il suffit d'utiliser un adaptateur USB-série externe. Si vous utilisez une version récente de l'ESP32, vous disposerez à la fois d'un port USB série pour le flashage et d'un port USB JTAG pour le débogage interne du SoC par implémentation logicielle.

Mode chargement de l'ESP32

Pour flasher le micrologiciel d'un ESP32, vous devez utiliser le « mode chargement », géré par l'étape de boot de la ROM (bootloader de 1^e niveau) ; sinon, le processus de boot se poursuivra et tentera de lire le bootloader flash (bootloader de 2^e niveau), puis l'application. Le mode chargement est activé en tirant vers le bas la broche GPIO BOOT (généralement GPIO0 ou GPIO9), en la maintenant ainsi et en réinitialisant le SoC. Une fois le mode chargement activé, l'ESP32 peut recevoir le micrologiciel via l'UART et le stocker dans la mémoire flash.

USB série interne et JTAG

Dans certains des SoC les plus récents, deux nouvelles fonctionnalités extrêmement utiles ont été introduites : USB série et JTAG. Elles améliorent le développement et minimisent la nomenclature en supprimant la liaison USB-série externe, et permettent également de réduire le nombre de broches utilisées (en particulier pour JTAG).

Lorsque vous utilisez l'UART pour le flashage, vous devez utiliser deux broches GPIO pour l'UART (TX et RX) et une broche GPIO pour activer le mode chargement. Vous devrez également connecter la broche Reset au circuit de réinitialisation avec la broche de mode de chargement. D'autre part, en utilisant le port USB série interne, seuls GPIO D+ et D- sont nécessaires car le mode de chargement automatique est géré en interne.

Les SoC pris en charge sont :

- ESP32-C3
- ESP32-S3 (comprend également l'hôte et le périphérique USB)
- ESP32-C6
- ESP32-H2
- ESP32-P4

Pour utiliser les fonctions USB Serial et JTAG sur les SoC pris en charge, vous devez ajouter un connecteur USB aux broches D+ et D- ou un montage d'essai connecté directement à l'ordinateur via un câble USB. Veuillez consulter la fiche technique pour déterminer les broches USB. Cela signifie que sur chaque ESP32 vous avez, en interne (sur les SoC mentionnés ci-dessus), les ports USB série et JTAG pour le flashage et le débogage, sans coût supplémentaire ou matériel externe.

L'ESP32-S2 ne dispose pas d'un port USB série ou JTAG, mais d'un port USB-OTG (hôte et périphérique), ce qui permet de flasher avec le DFU (mise à jour du micrologiciel du périphérique) au lieu du port USB série.

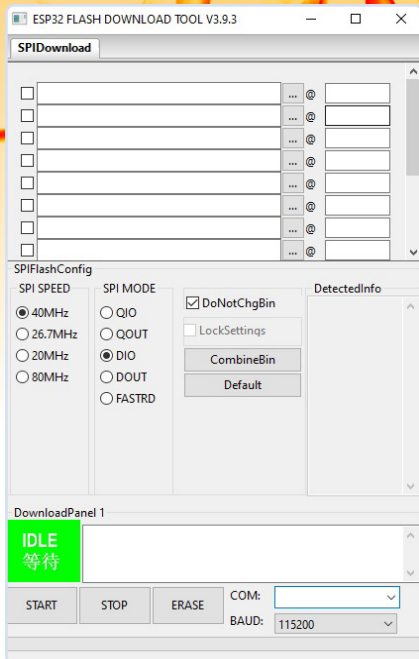


Figure 1. Flash Download Tools est l'application de programmation officielle d'Espressif, basée sur une interface graphique, que vous pouvez utiliser sans avoir besoin d'installer ESP-IDF ou d'exécuter ESPTool à partir du CLI.



Figure 2. Flash Download Tools en mode usine.

La sécurité est-elle assurée ?

Si vous vous souciez de la sécurité en livrant votre produit doté d'une interface série et JTAG intégrées, les deux fonctions peuvent être désactivées en utilisant les `eFuses DIS_USB_JTAG` et `DIS_USB_SERIAL_JTAG` pendant la phase de fabrication en série. Cette action n'est pas réversible, il faut donc y réfléchir pendant la phase de développement.

Un autre eFuse intéressant est `USB_EXCHG_PINS`. Cet eFuse échange les broches USB D+ et D- au cas où vous auriez accidentellement échangé les broches dans votre circuit.

Il est temps de flasher l'appareil !

Il existe différentes façons de flasher l'ESP32. Vous devez choisir celle qui vous convient le mieux, en fonction des spécificités de votre projet ou du stade de production. Voici un aperçu de toutes les possibilités disponibles pour flasher l'ESP32.

ESPTool

Le premier outil, et le plus répandu, est ESPTool [1]. ESPTool est un outil basé sur Python et l'outil principal pour flasher le micrologiciel. Il est intégré à l'ESP-IDF, c'est donc l'outil utilisé lorsque vous flashez le micrologiciel via l'interface en ligne de commande (CLI) ou via l'EDI.

Pour utiliser l'ESPTool directement depuis le CLI, vous devez définir certains paramètres, comme suit :

```
esptool.py -p -b --before default_reset --after hard_reset --chip write_flash --flash_mode --flash_size --flash_freq
```

Si vous êtes hésitant à propos de cette commande, bonne nouvelle : vous pouvez également flasher un micrologiciel construit avec ESP-IDF grâce à la commande `idf.py -p flash`.

Voici quelques-unes des fonctions d'ESPTool que vous pouvez utiliser en plus du flashage de micrologiciel :

- Effacement complet ou partiel de la mémoire flash
- Lecture et écriture sur la mémoire flash et la RAM
- Exécution du code d'application dans la mémoire flash
- Lecture des adresses MAC depuis de l'OTP ROM
- Lecture du Chip ID depuis l'OTP ROM
- Extraction des en-têtes d'un fichier binaire (bootloader ou application)
- Fusion de plusieurs fichiers binaires bruts en un seul fichier pour un flashage ultérieur
- Récupération de données relatives à la sécurité

Si vous devez flasher le micrologiciel pendant la production en série, vous pouvez utiliser ESPTool pour automatiser la procédure de flashage au moyen d'un script.

Flash Download Tools

Flash Download Tools est une application graphique pour Windows que vous pouvez utiliser pour le flashage sans avoir besoin d'installer ESP-IDF ou d'exécuter ESPTool sur la ligne de commande. L'interface graphique est assez simple et facile à utiliser et ne nécessite pas d'installation (figure 1).

Pour utiliser Flash Download Tools, vous devez sélectionner tous

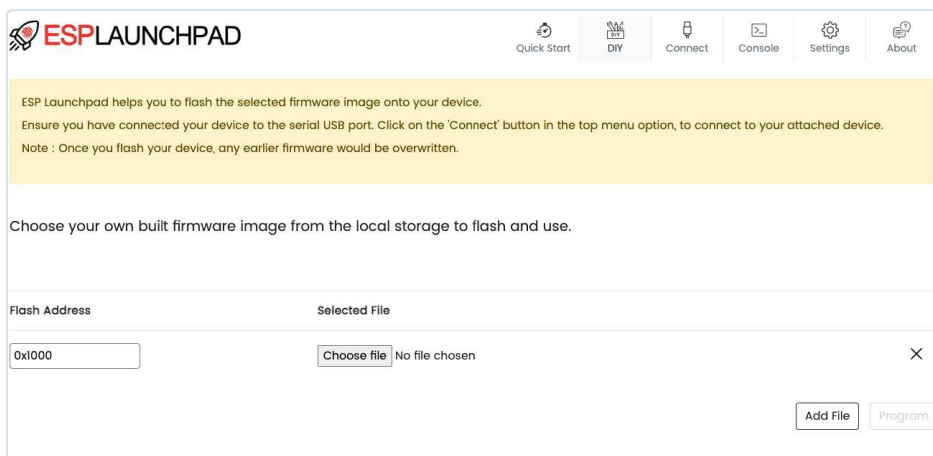


Figure 3. ESP Launchpad accédé directement depuis le navigateur Chrome.

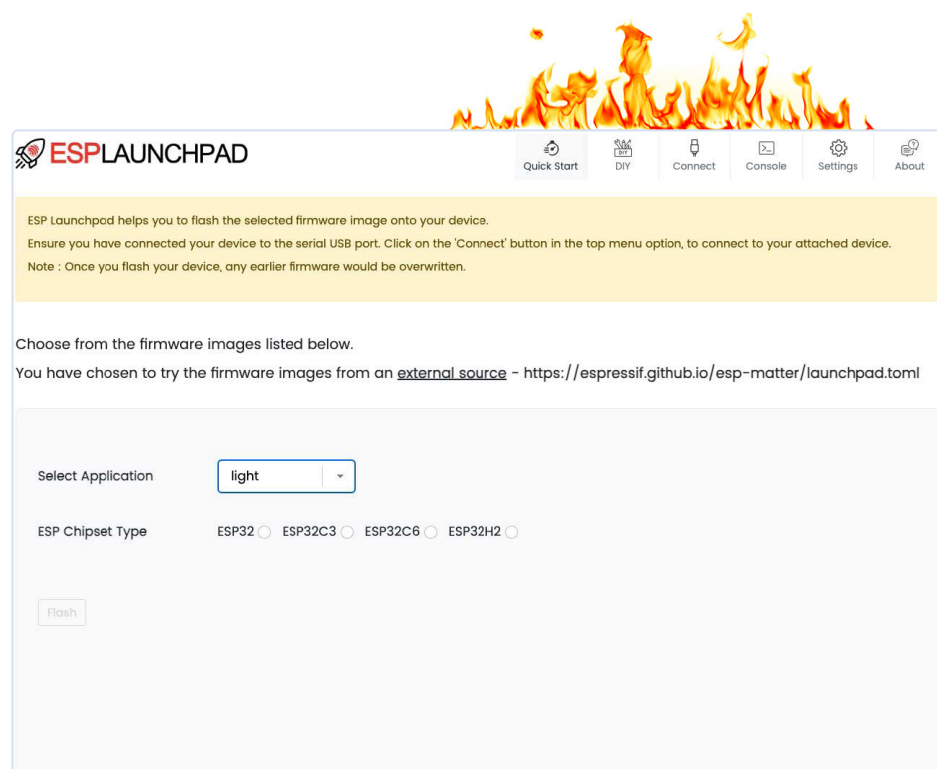


Figure 4. ESP Launchpad peut être personnalisé avec quelques fichiers de configuration supplémentaires.

les binaires que vous souhaitez flasher, ainsi que l'adresse de la mémoire flash pour chaque fichier.

Flash Download Tools fonctionne également en mode usine. Dans ce mode, vous pouvez sélectionner plusieurs interfaces série et flasher le micrologiciel simultanément en un seul clic (**figure 2**).

ESP Launchpad

Parfois, vous ne souhaitez pas installer ou exécuter une application sur votre ordinateur, et vous préférez utiliser uniquement des applications web. Dans ce cas, vous pouvez essayer ESP Launchpad directement depuis votre navigateur Chrome.

ESP Launchpad [2] est une application web basée sur *esptool-js*, qui permet de flasher le micrologiciel directement depuis le navigateur sans aucune installation, grâce à l'API WebUSB. Il est également possible de surveiller la sortie de la console et d'effacer la mémoire flash (effacer toutes les données de la flash).

ESP Launchpad est un logiciel open-source personnalisable (**figure 3**) avec quelques fichiers supplémentaires pour lui indiquer où est stocké le fichier du micrologiciel et les appareils cibles pris en charge par ce fichier (**figure 4**). Il vous suffit donc d'envoyer le lien à votre client et le processus de flashage sera simple et intuitif.



Exemple de fichier TOML

Dans le fichier de configuration TOML, vous pouvez ajouter l'URL des binaires et spécifier les cibles pour chaque binaire. Il est important de noter que le fichier binaire doit être un fichier unique, c'est-à-dire la version fusionnée comprenant tous les binaires requis. Vous pouvez les fusionner avec ESPTool ou Flash Download Tools. Voici un exemple de TOML :

```
esp_toml_version = 1.0
firmware_images_url = "https://espressif.github.io/esp-matter/"
supported_apps = ["light","light_switch"]
[light]
chipsets = ["ESP32","ESP32C3","ESP32C6","ESP32H2"]
image.esp32 = "esp32_light.bin"
image.esp32c3 = "esp32c3_light.bin"
image.esp32c6 = "esp32c6_light.bin"
image.esp32h2 = "esp32h2_light.bin"
ios_app_url = "https://apps.apple.com/app/esp-rainmaker/id1497491540"
android_app_url = ""

[light_switch]
chipsets = ["ESP32","ESP32C3","ESP32C6","ESP32H2"]
image.esp32 = "esp32_light_switch.bin"
image.esp32c3 = "esp32c3_light_switch.bin"
image.esp32c6 = "esp32c6_light_switch.bin"
image.esp32h2 = "esp32h2_light_switch.bin"
ios_app_url = "https://apps.apple.com/app/esp-rainmaker/id1497491540"
android_app_url = ""
```

Après avoir hébergé ce fichier sur un serveur public, vous pouvez utiliser l'URL combiné avec l'URL ESP Launchpad [3].

Le client final peut utiliser ESP Launchpad comme outil de mise à jour du micrologiciel, après la vente, lorsque la mise à jour à distance (OTA) n'est disponible.

ESP Serial Flasher

Imaginez que vous ayez besoin de flasher votre ESP32 à travers un autre appareil, ou un microcontrôleur hôte. Vous pouvez jeter un coup d'œil au projet *ESP Serial Flasher*. Dans ce projet, vous pouvez utiliser un microcontrôleur hôte pour flasher un ESP32 cible, ou vous pouvez utiliser un ESP32 pour programmer un autre ESP32 (**figure 5**).

Une application courante de cette fonction est lorsque l'ESP32 est utilisé comme coprocesseur radio et que vous souhaitez flasher une nouvelle version du micrologiciel via l'appareil hôte.

Actuellement, nous prenons en charge les contrôleurs d'hôte suivants :

- › ESP32
- › STM32
- › Raspberry Pi
- › Tout microcontrôleur fonctionnant sous Zephyr OS

Les microcontrôleurs cibles qui prennent en charge la mise à jour via UART sont les suivants :

- › ESP32
- › ESP8266
- › ESP32-S2
- › ESP32-S3
- › ESP32-C3
- › ESP32-C2
- › ESP32-H2
- › ESP32-C6
- › ESP32-P4

Mise à jour OTA

Enfin, la mise à jour à distance (over-the-air) est une autre solution intéressante pour la mise à jour des micrologiciels en pratique. Cette technique est souvent utilisée pour mettre à jour les micrologiciels via l'internet ou un réseau local, sans avoir besoin du matériel pour gérer les différentes mises à jour.

Lorsque vous rencontrez des problèmes avec la version actuelle de votre micrologiciel ou que vous devez y ajouter une nouvelle fonction, la mise à jour OTA est avantageuse. Vous pouvez la faire à distance pour plusieurs appareils en même temps. Vous économisez ainsi du temps et de l'argent.

Pour commencer à utiliser l'OTA, vous pouvez consulter les exemples du projet ESP-IDF sur GitHub ou dans votre environnement local.

Bonus : ESP USB Bridge

Il s'agit d'un autre projet ESP-IDF utile pour le développement, en particulier si vous utilisez un ESP32 sans USB série ou JTAG, comme l'ESP32, l'ESP32-C2, et l'ESP32-S2.

Vous pouvez transformer n'importe quel ESP32-S2 ou ESP32-S3 en un appareil USB Serial et JTAG et l'utiliser pour flasher et déboguer un autre ESP32. Il suffit de flasher n'importe quel ESP32-S2 ou ESP32-S3 avec ESP USB Bridge et de connecter les GPIO au microcontrôleur cible (**figure 6**).

Vous pouvez utiliser ce projet avec tous les outils de flashage, tels que ESPTool, Flash Download Tools et ESP Launchpad. Vous pouvez également utiliser l'USB-MSC (Mass Storage Class) en ajoutant le binaire au format UF2 sur la mémoire de masse USB qui apparaît lorsque vous connectez l'ESP-USB-Bridge à votre ordinateur.

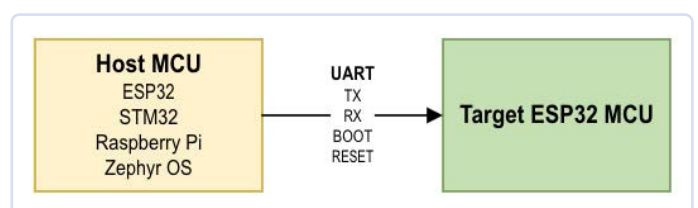


Figure 5. Avec le projet ESP Serial Flasher, vous pouvez utiliser un microcontrôleur hôte pour programmer un ESP32 cible, ou même utiliser un autre ESP32 comme hôte pour le flasher.

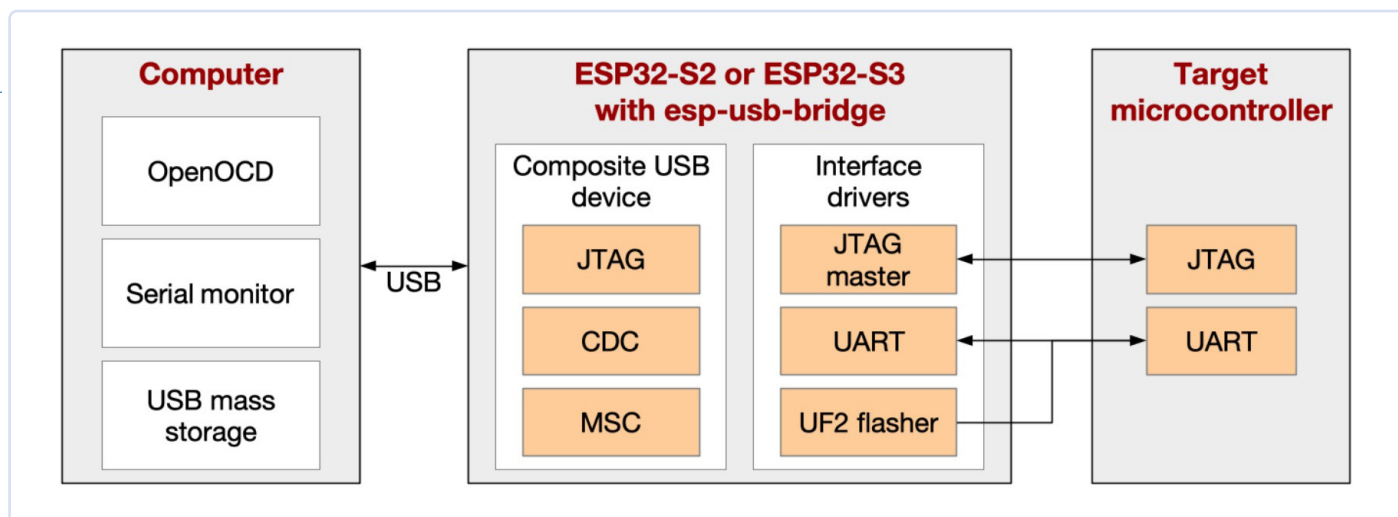


Figure 6. ESP USB Bridge est un projet ESP-IDF utilisant un ESP32-S2 ou un ESP32-S3 pour créer un lien entre un ordinateur (PC) et un microcontrôleur cible (MCU).

De nombreuses possibilités

Dans le monde des microcontrôleurs ESP32, il existe de nombreuses façons de charger un nouveau micrologiciel sur votre ESP32. Bien qu'ESPTool semble être la solution idéale, il est important de garder à l'esprit que certains exemples nécessitent des outils plus avancés dotés d'interfaces utilisateur graphiques (GUI) ou des processus de flashage simplifiés qui s'adressent aux utilisateurs de diverses plateformes sans nécessiter d'installation logicielle.

Cet article vise à donner des idées pour améliorer le développement et la production de produits. Il vise également à donner aux individus la possibilité de flasher ou de mettre à jour facilement les micrologiciels vers les versions les plus récentes, indépendamment de leur formation technique ou de la plateforme qu'ils utilisent. ◀

230625-04



À propos de l'auteur

Pedro Minatel est Developer Advocate chez Espressif. Il est titulaire d'un diplôme technique en électronique et d'un diplôme en technologie de l'information. Pedro a commencé à travailler pour Espressif en 2021, mais il est un membre actif de la communauté depuis 2014, publiant des articles et présentant des conférences sur l'IdO et la communauté des makers. Actuellement, il est responsable de la conférence annuelle des développeurs d'Espressif, connue sous le nom de DevCon.

Questions ou commentaires ?

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



Produits

- > ESP32-C3-DevKitM-1
www.elektor.fr/20324
- > Carte de développement LILYGO T-Display-S3 ESP32-S3 (avec connecteurs)
www.elektor.fr/20299



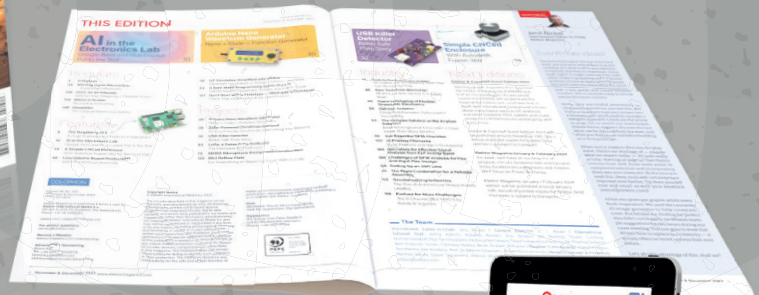
LIENS

- [1] ESPTool : <https://github.com/espressif/esptool>
- [2] ESP Launchpad : <https://espressif.github.io/esp-launchpad/>
- [3] ESP Launchpad URL : <https://espressif.github.io/esp-launchpad/?flashConfigURL=https://espressif.github.io/esp-matter/launchpad.toml>

20%
de réduction
sur la première année de
votre abonnement

Rejoignez la communauté Elektor

Devenez membre
maintenant !



- ✓ accès à l'archive numérique depuis 1978 !
- ✓ 8x magazine imprimé Elektor
- ✓ 8x magazine numérique (PDF)
- ✓ 10 % de remise dans l'e-shoppe et des offres exclusives pour les membres
- ✓ accès à plus de 5000 fichiers Gerber
- ✓ Livraison gratuite en France



www.elektormagazine.fr/gold-member

Utilisez le code promo :

ESPRESSIF20



ESPRESSIF



elektor



des fusées jusqu'aux violoncelles

applications pratiques et réflexions lors
de la création de solutions sans fil

Sorin Jayaweera, GXB Ventures

Qu'obtient-on lorsqu'on combine un bouton-poussoir et une antenne ? Un ordinateur de fusée, évidemment !

Le rêve

Dans cet article, nous tenterons de vous présenter les outils essentiels pour vous lancer dans des projets sans fil, en nous concentrant sur le matériel de base, les structures des réseaux de communication et les éléments essentiels à prendre en compte.

Nous avons fait nos premiers pas en essayant de construire un réseau de petits boutons d'aide, peu coûteux et sans fil, dans l'espoir de les répartir dans les maisons des membres de notre famille élargie, afin de permettre un contact facile avec le reste de la famille en cas d'urgence. Au niveau le plus élémentaire, chaque bouton nécessite la capacité d'enregistrer des données - si le bouton a été enfoncé - et la capacité de transmettre ces données d'une manière ou d'une autre à un système qui peut les utiliser. Nous devons également mettre au point une passerelle vers l'internet capable de surveiller le réseau local, tout en relayant les alarmes aux membres éloignés de la famille. Il s'agit d'une tâche apparemment simple, mais qui repose sur les mêmes bases que de nombreux autres projets intéressants dans le monde réel. Par exemple, en utilisant les mêmes systèmes que le bouton pour un réseau de communication à courte portée, mais en ajoutant des capteurs pour enregistrer des données environnementales,

on pourrait créer une station de surveillance météorologique locale efficace. L'utilisation domestique que nous avons créée répondrait à des questions telles que « quelle est la température actuelle et la température maxi/mini dans le grenier, le sous-sol et le poulailler à l'extérieur ? ». À partir de ces données, nous pourrions décider d'activer automatiquement des ventilateurs pour rafraîchir ces espaces ou de fermer les bouches d'aération pour les isoler. Plusieurs de ces stations de surveillance météorologique simples et peu coûteuses, fonctionnant en association dans une exploitation agricole, pourraient fournir des informations inestimables pour mieux prendre soin des cultures. En augmentant la portée sans fil, et en ajoutant des capteurs appropriés tels que des accéléromètres et des GPS, on obtient un suivi de modèles réduits de fusées et un enregistreur de données de vol pour les amateurs. En augmentant considérablement la portée, on peut obtenir un système de télémétrie et de suivi des ballons à haute altitude. Inversement, la réduction drastique de la latence à faible portée est utile pour la conception de systèmes de musique électronique (c'est-à-dire la fabrication de petites pédales de commande sans fil qui communiquent avec n'importe quel microcontrôleur au centre de la production musicale). En réalité, un grand nombre de choses intéressantes deviennent possibles avec seulement quelques techniques courantes de communication, d'enregistrement des données et de traitement des signaux. Quelles sont donc les exigences fondamentales au cœur de ces types de projets ?

Les aspects techniques

Pour répondre aux exigences mentionnées plus haut, les circuits doivent être compacts, légers et disposer d'une gestion efficace de l'énergie. En plus, chaque projet doit avoir la capacité de communiquer

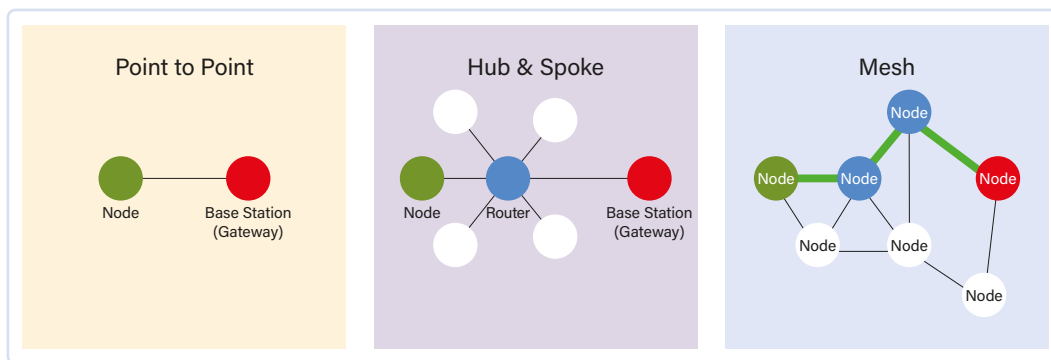


Figure 1. Topologies des réseaux.

sans fil via un réseau, des réseaux spécifiques « maillés » par exemple, ou des réseaux « en étoile » (nous verrons ce que c'est bientôt). Ces réseaux peuvent être de petite envergure, comme par exemple le bouton d'aide, ou à longue portée pour les fusées, mais la structure générale de ces réseaux est la même.

Pour les boutons d'aide et les projets ultérieurs, nous avons exclu les mini-ordinateurs puissants tels que le Raspberry Pi, qui ne supportent pas les coupures brutales et ont un besoin d'énergie relativement important. Au début, nous avons utilisé des contrôleurs Teensy, qui sont des microcontrôleurs compatibles Arduino très puissants pour le traitement des signaux et qui constituent une excellente solution pour de nombreux problèmes. Bien qu'ils soient très adaptables, ils nécessitent des périphériques supplémentaires pour tout ce qu'ils doivent faire, en particulier les communications sans fil, et cela devient problématique au fur et à mesure du développement des projets. Les communications sans fil à longue portée étant très importantes, nous avons commencé à chercher des solutions intégrées prédéfinies afin de minimiser la complexité de notre installation.





Nous avons essayé les cartes LoRa de Heltec, car elles intègrent le Wifi, le Bluetooth et LoRa. Bien qu'elles semblent prometteuses, nous avons rencontré de nombreuses difficultés pour les faire fonctionner, et nous n'avons trouvé aucune aide dans leurs forums et le support technique. Les exemples de code fournis ne se compilaient pas correc-

tement, sans que nous ayons trouvé comment éditer nous-mêmes les sources et les bibliothèques liées. Et même les antennes livrées n'étaient pas adaptées aux bandes de fréquences LoRa pour lesquelles elles étaient censées être optimisées. Nous avons continué à chercher et avons découvert les différentes puces Espressif et les cartes de développement qui les utilisent.

Les puces Espressif intègrent le Wifi et le Bluetooth, ce qui réduit la complexité générale de notre système. Pour de nombreux projets IdO de proximité à courte distance, l'utilisation des antennes intégrées des cartes fonctionne bien, car la portée peut être étendue en utilisant plusieurs cartes et en transférant les messages. Pour être honnête, toutes les cartes commerciales n'ont pas bien fonctionné. Un test de portée ESP-NOW (Wifi) utilisant une carte ESP-WROOM32, par rapport à une carte ESP32-C3, a donné des portées de 3 mètres et 80 mètres, respectivement. Mais après avoir limité le choix des possibilités, nous avons vraiment apprécié l'intégration de l'ESP32-C3 dans nos projets sans fil. Cela est dû en grande partie à la taille minuscule de la puce, à son assistance spécifique pour nos types de projets, à sa faible consommation d'énergie et à sa communication intégrée avec une antenne imprimée bien conçue.

Maintenant que nous avons notre « ordinateur » principal, il ne nous restait plus qu'à concevoir le circuit imprimé et les capteurs et E/S supplémentaires pour chaque projet. Pour les boutons d'appel à courte

Comparaison des besoins

Carte	Coût (USD)	Puissance (mW)	Pour	Contre
Arduino 	\$15+	~400	+ Facile à apprendre + Prototypage rapide	- Pas de multitâche - Pas de communication - (Assez) encombrant
Teensy 	\$15-40+	~800	+ Petit + Très rapide + Prise en charge étendue de l'audio + Communauté active	- Pas de communication intégrée - des besoins en énergie importants
Raspberry Pi 	\$15-80+	~1200	+ Un vrai ordinateur + Multitâche + n'importe quel langage de programmation + Sans fil intégré (Bluetooth, WiFi)	- (Assez) onéreux - Ne tolère pas les arrêts brusques - Nécessite plus d'énergie que les microcontrôleurs - Plus grand que les microcontrôleurs - Nécessite beaucoup d'énergie
ESP32 	\$2-\$5+	6-300	+ Compatible avec Arduino + Cartes de développements petites et modules encore plus petits + Sans-fil intégré (WiFi, Bluetooth 5 / BLE) + Bon marché + Optimisé pour les basses consommations	- Se connecte seulement au WiFi 2.4 GHz - Réseaux ESPNow & ESP-WiFi-Mesh propriétaires

portée, nous avons décidé de faire en sorte que chaque bouton ait son propre module C3, alimenté avec des piles bouton, ne fonctionnant que lorsque le bouton est pressé. Afin de recevoir les signaux d'aide, nous avons également une passerelle en écoute permanente, qui se connectait ensuite à l'internet global et nous envoyait un message texte. Il s'agit d'une petite version d'un réseau en étoile (**figure 1**), avec des passerelles « toujours à l'écoute », connectées à des capteurs qui ne s'activent et n'émettent qu'en cas de besoin. Comme chaque bouton ne doit se déclencher que lorsqu'il émet son propre signal, la durée de vie de la batterie s'en trouve considérablement optimisée. Nous avons mis en œuvre ce système en utilisant uniquement ESP-NOW entre chaque nœud final et la passerelle en écoute permanente, qui transmet les messages vers internet (via Ethernet).

Une autre topologie de réseau très utile est appelée réseau maillé. Dans un réseau maillé, chaque nœud est toujours actif. Les messages sont acheminés par le chemin le plus direct jusqu'à la destination finale. Les réseaux maillés ont plus de redondances. Si un seul nœud tombe en panne, le reste du système peut continuer à fonctionner en changeant l'itinéraire d'acheminement. La version à courte portée d'un réseau maillé est relativement facile à mettre en place, en utilisant la bibliothèque *painlessMesh* ou un réseau ESP-WIFI-MESH avec des cartes Espressif, dont les codes sources se trouvent facilement en ligne. Pour les projets à longue portée, pourtant, le Wifi n'est pas une solution viable. Notre approche préférée utilise LoRa, un protocole de communication à faible débit de données et à faible consommation d'énergie, mais avec la possibilité d'avoir une portée de plusieurs kilomètres. LoRa a un débit de données très faible et n'est que semi-duplex (la plupart des émetteurs ne peuvent pas envoyer et recevoir en même temps), ce qui rend les réseaux maillés difficiles à mettre en œuvre. Il existe plusieurs autres méthodes de communication possibles, qui sont détaillées dans l'encadré *Comparaison des communications sans-fil*. Veuillez noter que nous n'avons pas testé toutes les méthodes, car nous avons principalement obtenu des informations sur Internet avant de choisir les méthodes à utiliser. Nous avons compilé nos propres données pour LoRa, ESP-WIFI-MESH et ESP-NOW. D'autres solutions sont plus difficiles à utiliser, mais sont applicables à des projets plus formels.

Pour le système de suivi des fusées et d'enregistrement des données, nous avons utilisé la communication LoRa plutôt que la communication Wifi, car nous avons besoin de stabilité et d'une longue portée. Pour les périphériques LoRa, nous recommandons soit le Hope RFM95W ou le Reyax RYLR406. Le Reyax (basé sur UART) est très convivial, il est livré avec une bonne antenne et envoie des données avec des commandes AT relativement simples. Il ne dispose pas d'une structure de réseau préétablie, mais fonctionne parfaitement pour une communication point à point simple. Alternativement, la carte HopeRF (utilisant SPI) a beaucoup plus de support avec *radiolib* ou la bibliothèque radiohead, qui ont des implémentations pour les réseaux maillés ad hoc (légèrement instables).

Nous avons utilisé une topologie maillée pour les systèmes de fusées, car nous pouvions ainsi lancer plusieurs fusées et faire en sorte que

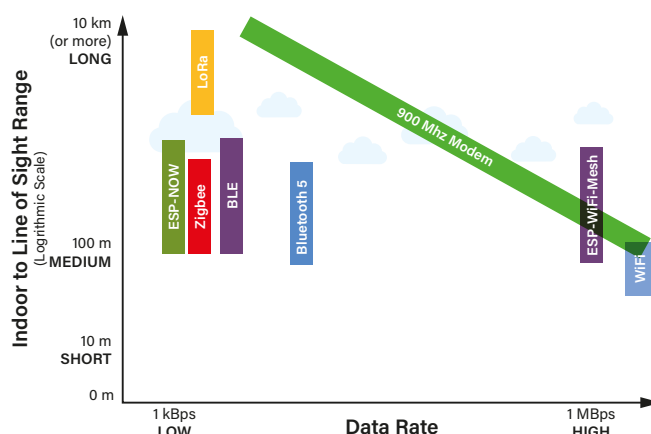
celles qui étaient en vol reçoivent des transmissions de celles qui étaient au sol, puis communiquer l'emplacement de chaque fusée à notre station de base. Comme nous ne lançons pas plus de trois fusées à la fois, la lenteur de LoRa et son incapacité à transmettre tout en écoutant n'étaient pas un désavantage pour nos objectifs. Cependant, pour les projets à plus grande échelle, il serait bon d'envisager l'utilisation de LoRaWAN ou d'autres grands réseaux en étoile qui prennent en charge la transmission en duplex intégral.

Nous avons essayé de rassembler le plus d'informations possibles pour les débutants dans ce bref article. Pour des informations plus détaillées sur les structures de réseau, les bases de l'utilisation des systèmes Espressif, etc., regardez la vidéo de notre conférence, *DevCon23 - From Rockets to Cellos: Real-world Applications of ESP32 Series and Dev Board Variants*, sur le site web d'Espressif ou sur YouTube [1].

VF : Laurent Rauber — 230627-04

Comparaison des communications sans-fil

Technologie	Débit de données (Kbps)	Portée en intérieur (vue directe)
WiFi 2.4 GHz (≠ 5.2 GHz)	9,000	25 m (100 m)
ESP-WiFi-MESH	1,000	50 m (200 m)
Bluetooth 5 (audio)	260	50 m (240 m)
Bluetooth LE (bas débit)	120	70 m (500 m)
Zigbee 2.4 Ghz	30	70 m (250 m)
LoRa 915 MHz	30	1.2 km (20 km)
ESP-NOW	1	70 m (500 m)
Modem 900 MHz	25 to 7,000	50 m (45 km)



LIEN

[1] "DevCon23 - From Rockets to Cellos: Real-world Applications of ESP32 Series and Dev Board Variants": <https://youtu.be/jxPUkmaYp2c>



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

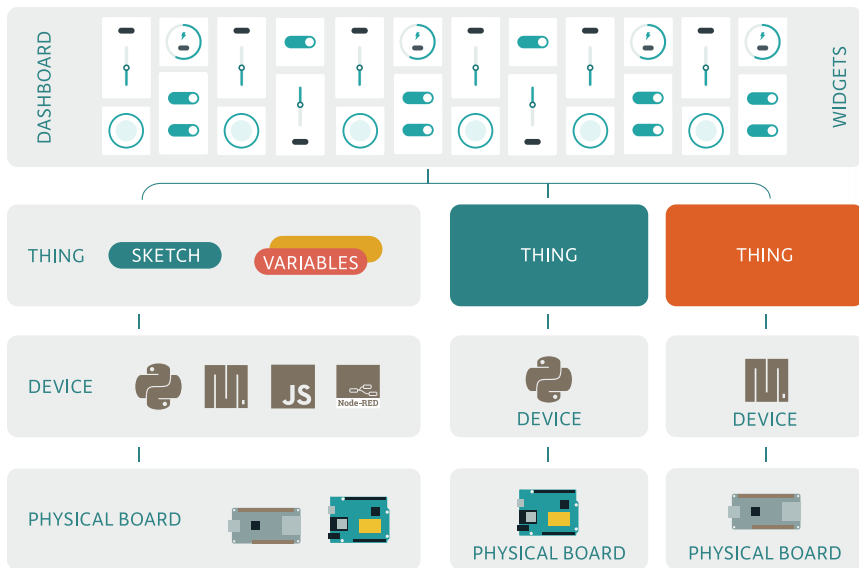
Program/Deploy

- + 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?



FULL CONTROL IN YOUR HANDS
Use your dashboards on the go, and control projects from anywhere in the world, using the free IoT Cloud Remote app.

GET IT ON **Google Play** | Download on the **App Store**

Compatible hardware

WITHIN ARDUINO DEVELOPMENT ENVIRONMENTS



ARDUINO

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



ESP32/ESP8266

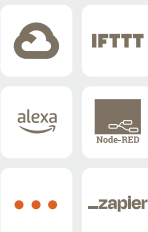
+70% Of Arduino Cloud active users use ESP-based boards.

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

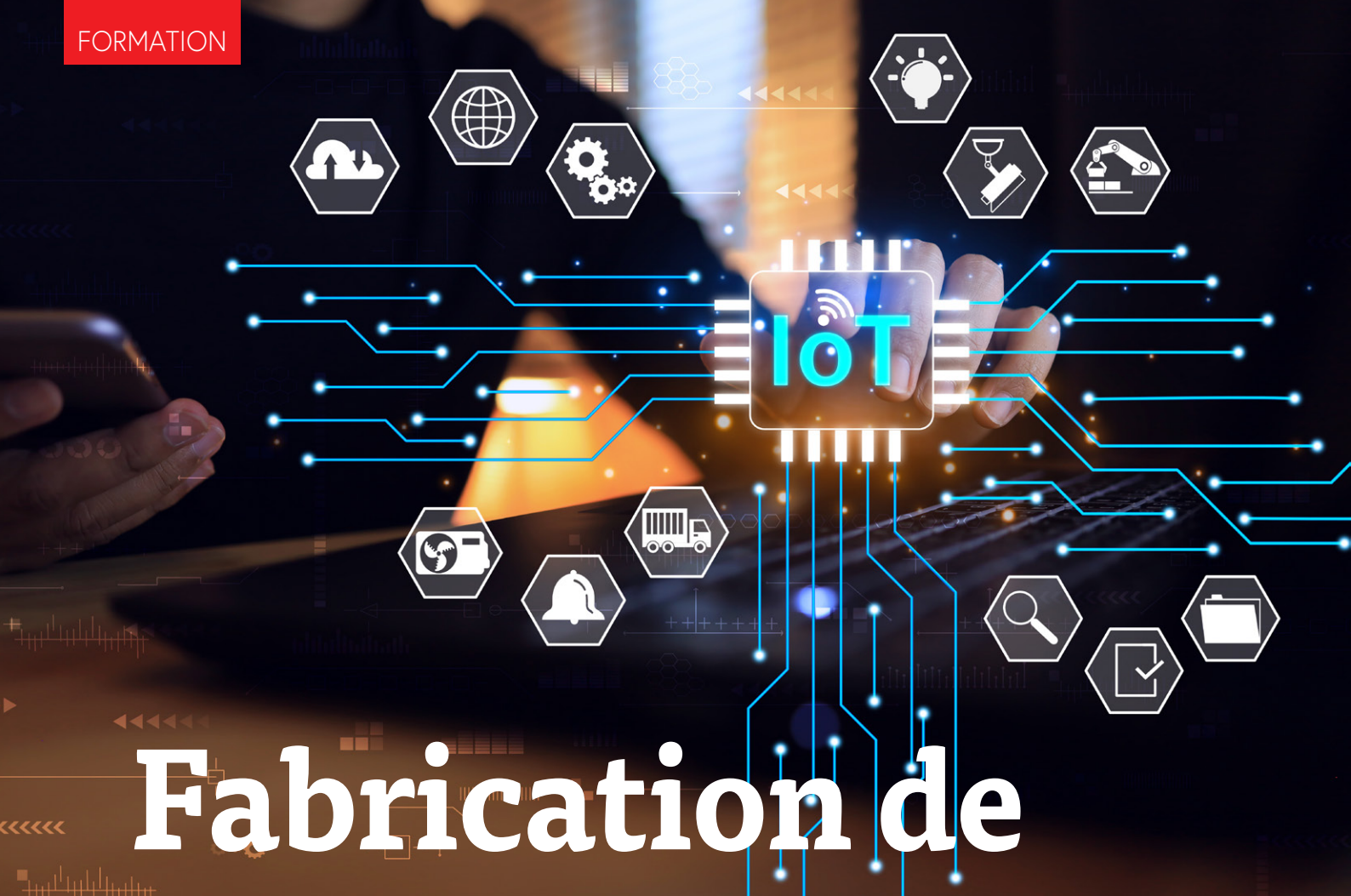
Connect your Arduino Cloud devices to external platforms such as IFTTT, Zapier and Google Services using webhooks and unlock endless possibilities.

Seamlessly integrate your IoT devices with over 2 000 apps, enabling tasks like receiving 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



Fabrication de dispositifs IdO sécurisés

Pourquoi ? Comment ?

Aditya Patwardhan, *Espressif*

L'intelligence accrue des dispositifs implantés en périphérie de réseau, et en particulier leur évolution continue, impose des niveaux de sécurité accrus et une dynamique constante d'adaptation de la part des fabricants. Nous allons découvrir ici en quoi la démarche de fabrication de dispositifs IdO sécurisés d'Espressif répond à ces besoins en pleine évolution

L'Internet des objets - IdO (ou IoT en anglais, pour *Internet of Things*) s'est parfaitement intégré à notre vie quotidienne. Avec les progrès des environnements de développement vers davantage de sophistication, la création de nouveaux produits pour l'IdO est devenue nettement plus accessible. À mesure que le secteur de l'Internet des objets évolue dans le sillage des innovations, les applications deviennent de plus en plus complexes. Un aspect important de cette évolution est la préoccupation croissante à propos de la

sécurité. Sur le marché, il existe désormais des normes de sécurité plus strictes, assorties individuellement de leurs propres exigences. Le processus de fabrication est ainsi devenu plus complexe et impose un contrôle de sécurité approfondi à chaque étape. Dans cet article, nous allons passer en revue différents aspects de la fabrication sécurisée de dispositifs pour l'IdO. Nous aborderons aussi les problèmes rencontrés à chaque étape et la manière dont, chez Espressif, nous facilitons la tâche à nos clients.

Secrets des appareils et données uniques par appareil

Les anciens processus de fabrication des dispositifs à base de microcontrôleurs étaient relativement simples. Après avoir flashé le micrologiciel sur le microcontrôleur et y avoir appliqué différents tests d'assurance qualité, l'appareil était prêt à être déployé sur le terrain. Cependant, avec l'essor de la connectivité, l'environnement a significativement évolué. Aujourd'hui, la plupart des produits IdO ne se contentent plus d'une simple adresse MAC unique ; des informations d'identification individuelles propres à chaque appareil sont nécessaires. Certaines de ces informations sont formées de données secrètes servant à communiquer en toute sécurité avec des serveurs distants. D'autres données secrètes interviennent pour sécuriser l'intégration de l'appareil ou sont spécifiques au microcontrôleur, comme par exemple celles servant à chiffrer les données au repos stockées en mémoire flash.

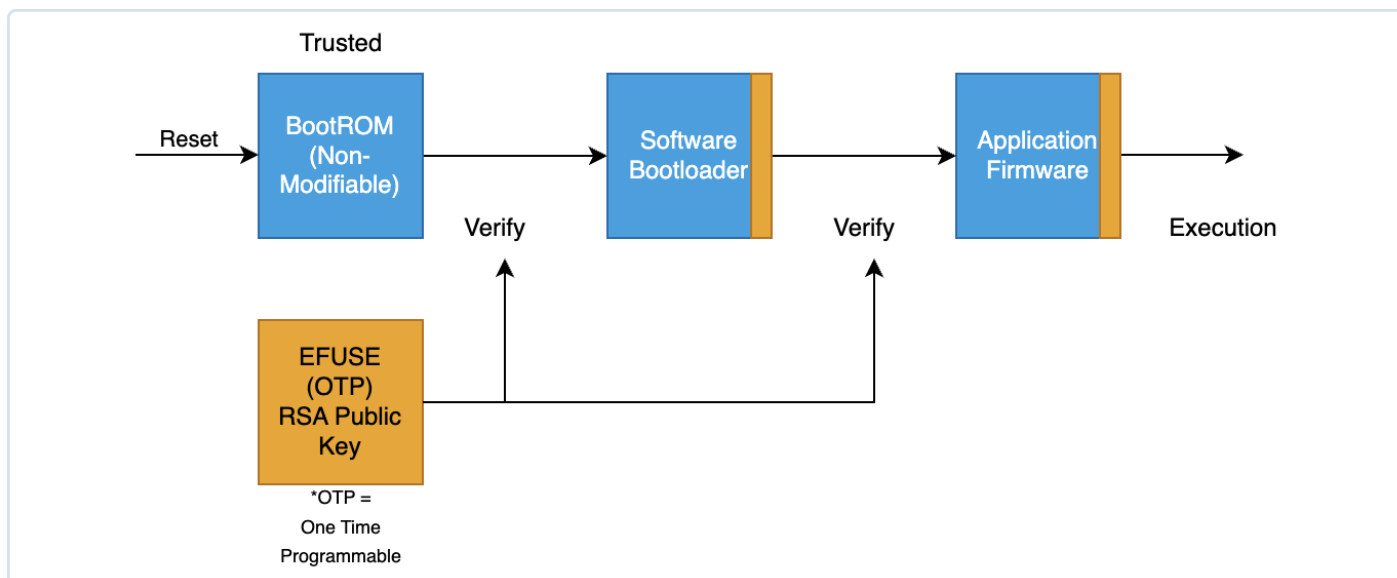


Figure 1. Schéma du modèle de confiance transitive pour assurer un démarrage sécurisé.

La cryptographie fondée sur une infrastructure à clés publiques (PKI) joue un rôle important dans la sécurité des appareils. En général, un appareil dispose de plusieurs clés : une clé publique pour authentifier directement ou indirectement le serveur avec lequel il communique, une clé publique pour vérifier l'authenticité de tout nouveau micrologiciel installé sur l'appareil et une paire de clés publique-privée formant un certificat d'appareil unique, enregistré auprès du serveur cloud ou de la base de données centrale afin de permettre l'authentification de l'appareil à partir du cloud ou d'un client. De plus, l'appareil doit stocker des informations sensibles, notamment des informations d'identification du réseau Wi-Fi. Dans ce cas spécifique, les principales exigences en matière de sécurité sont les suivantes :

1. Veiller à ce que les clés publiques servant à authentifier d'autres entités ne soient pas altérées.
2. S'assurer que la clé privée de l'appareil soit générée dans un environnement sécurisé à l'aide d'un générateur de nombres aléatoires de bonne qualité.
3. Veiller à ce que la clé privée de l'appareil soit stockée en toute sécurité, de manière à ce qu'elle ne soit pas la proie des pirates, car elle constitue l'identité de l'appareil.
4. Le certificat de l'appareil doit porter la signature d'une entité autorisée à signer des certificats.
5. S'assurer que les informations sensibles, notamment les identifiants Wi-Fi, soient stockées en mémoire flash dans un format chiffré, de sorte qu'une simple lecture de cette mémoire ne puisse pas révéler ces informations.

Cela se traduit par des exigences de sécurité plus fondamentales :

1. Veiller à ce que l'appareil n'exécute que le micrologiciel de confiance afin qu'il ne puisse pas être programmé à l'aide d'un code malveillant susceptible d'entraîner la fuite de données sensibles.
2. S'assurer que chaque appareil puisse chiffrer sa mémoire flash, de préférence avec une clé de chiffrement unique, de sorte que même si cette clé est trouvée, l'ensemble du parc d'appareils ne sera pas compromis. Il est donc préférable de générer cette clé de chiffrement unique de manière aléatoire à l'intérieur de la puce et de ne pas y donner accès à l'extérieur.
3. Veiller à ce que le processus de fabrication génère une clé privée sur la puce et que cette clé ne soit pas accessible de l'extérieur.
4. S'assurer que le processus de fabrication fonctionne avec du matériel de confiance ou un service dans le cloud pour la signature des certificats.

Complexité de la fabrication

Bien que les microcontrôleurs modernes, parmi lesquels la gamme ESP32, offrent les fonctions de sécurité nécessaires à la mise en œuvre des exigences déjà précisées dans cet article, il convient de faire preuve d'une grande prudence au moment de la fabrication. En règle générale, les fabricants sous-traitants qui programment ces types de dispositifs ne connaissent pas nécessairement les meilleures pratiques en matière de sécurité, et la complexité d'une fabrication sécurisée peut s'avérer insurmontable. Espressif propose ainsi un certain nombre de solutions pour simplifier ce processus.

Activation du démarrage sécurisé automatique

La fonction de démarrage sécurisé garantit que le matériel authentifie, à chaque démarrage, le micrologiciel exécuté sur la puce. Ainsi, cette puce est verrouillée à l'aide d'une clé publique qui peut vérifier l'authenticité de la signature contenue dans le micrologiciel signé. Le framework de développement ESP-IDF d'Espressif permet d'activer un démarrage sécurisé dans le chargeur d'amorçage de deuxième niveau. Ce qui garantit que ce chargeur, lorsqu'il est programmé sur la puce et exécuté pour la première fois, code la mémoire OTP (écrite une seule fois) avec la clé publique utilisée par le matériel pour le démarrage sécurisé. Le chargeur d'amorçage peut également désactiver les interfaces de débogage et de programmation afin de verrouiller la puce. Ainsi, le processus de fabrication n'a pas la responsabilité de programmer correctement la mémoire OTP pour sécuriser l'appareil. Ce processus peut être confié aux développeurs qui maîtrisent mieux la sécurité que ceux qui fabriquent l'appareil.

La **figure 1** illustre le modèle de confiance transitive pour assurer un démarrage sécurisé. Une fois la fonction de démarrage sécurisé activée, la puce peut exécuter un micrologiciel de confiance, qui peut lui-même contenir les différentes clés publiques mentionnées ci-dessus, ces clés ne pouvant pas être modifiées de force.

Activation du chiffrement automatique de la mémoire flash

Nous avons discuté de la nécessité de disposer d'une clé de chiffrement symétrique unique dans l'appareil pour chiffrer la mémoire flash. Les microcontrôleurs de la série ESP32 sont dotés d'une fonction de chiffrement de la

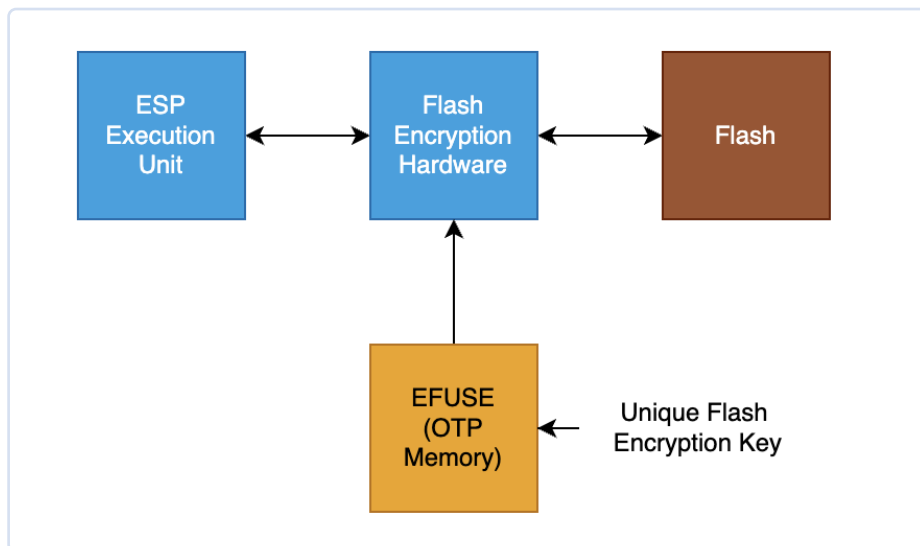


Figure 2. Schéma d'une version simplifiée du chiffrement de la mémoire flash.

mémoire flash grâce à laquelle le matériel prend en charge une clé de chiffrement « non lisible par logiciel » dans la mémoire OTP. Cette clé peut chiffrer et déchiffrer de manière transparente et dynamique le contenu de la mémoire flash. Le chargeur d'amorçage du framework ESP-IDF permet d'activer cette fonction en cours d'exécution. Lors du premier démarrage de l'appareil, le chargeur peut générer la clé de chiffrement de la mémoire flash dans la puce à l'aide de son générateur de nombres aléatoires (TRNG) et la programmer dans la mémoire OTP avec activation d'une protection anti-lecture. Facultativement, le chargeur de démarrage logiciel peut également chiffrer le micrologiciel et d'autres contenus flash nécessaires qui doivent être protégés lors du premier démarrage. Ainsi, le chiffrement de la mémoire flash, lorsqu'il est activé par le chargeur d'amorçage du logiciel, permet de générer une clé aléatoire de chiffrement de la mémoire flash

par appareil et de sécuriser ledit chiffrement pour protéger toutes les données sensibles, notamment les informations d'identification du réseau Wi-Fi ou la clé privée du certificat de l'appareil.

La **figure 2** présente une version simplifiée du chiffrement de la mémoire flash.

Provisionnement préalable des certificats sécurisés

En ce qui concerne le certificat de l'appareil, nous avons vu qu'il est important de protéger la génération et le stockage de la clé privée. Il est également essentiel de faire signer le certificat de l'appareil à une entité de confiance - en l'occurrence, une autorité de certification. Espressif applique un processus sécurisé de provisionnement de certificats dans ses usines, les clients passant ainsi commande de modules préalablement provisionnés.

Dans ce processus de provisionnement préalable de modules, trois entités travaillent

conjointement. Un hôte de provisionnement, formé d'un PC fonctionnant avec une puce ou un module de la gamme ESP32 pour le provisionnement. L'hôte de provisionnement est connecté à un module matériel de sécurité (HSM) (ou boîte noire transactionnelle en français) local ou dans le cloud, qui détient l'autorité de certification et peut signer n'importe quel certificat sans divulguer la clé privée utilisée pour cette signature. La plupart des HSM sont également capables de générer des journaux non répudiables qui peuvent garantir le nombre de certificats signés par ces HSM.

Le processus est illustré par la **figure 3**.

La clé privée du certificat de l'appareil est générée sur la puce et ne la quitte jamais. L'hôte fournit les paramètres du certificat (notamment la durée de validité, le nom commun, etc.), et reçoit la demande de signature dudit certificat (CSR). Cette CSR est ensuite envoyée au HSM local ou dans le cloud pour obtenir un certificat signé. Ce certificat est ensuite envoyé à l'appareil, qui est alors verrouillé à l'aide d'un amorçage sécurisé et d'un chiffrement de la mémoire flash. Nombre de nouvelles puces de la gamme ESP32 disposent également d'un périphérique dédié à la « signature numérique » (ou DS), chargé de protéger la clé privée de l'appareil à l'aide d'un bloc matériel spécial. Ce périphérique DS peut fournir des opérations de certificat directement avec la clé privée chiffrée, garantissant ainsi que la clé privée en clair (donc non chiffrée) est également inaccessible pour le logiciel.

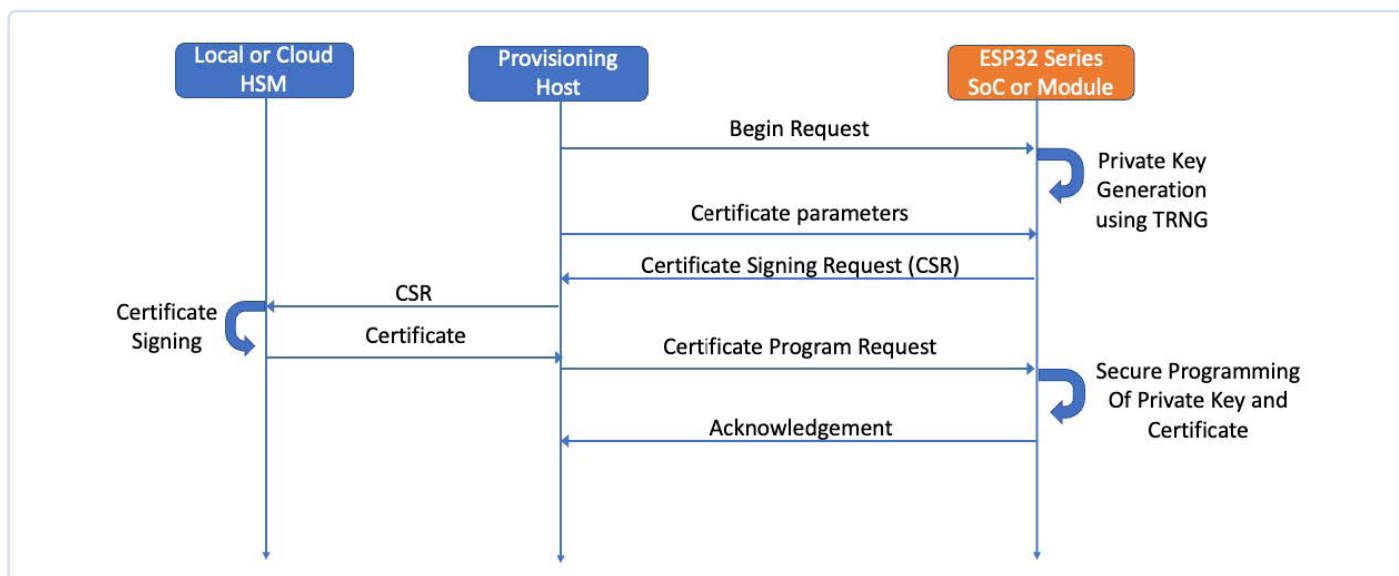


Figure 3. Organigramme du processus de provisionnement préalable des certificats sécurisés.



Autres données uniques par appareil

Au-delà de ces artefacts de sécurité, l'appareil peut également avoir besoin d'autres données uniques qui lui sont propres, notamment son identifiant. Espressif propose des outils simples d'utilisation pour générer des contenus binaires de données à partir d'un fichier CSV contenant les données uniques de chaque appareil sous forme de tableaux. N'hésitez pas à consulter le document [1].

Tout se tient

Les exigences de sécurité et la programmation de données uniques pour chaque appareil contribuent à la complexité de la fabrication de ces dispositifs pour l'Internet des objets. Espressif offre une grande flexibilité dans le processus de fabrication, ce qui aide les

À propos de l'auteur

Aditya Patwardhan est ingénieur logiciel chez Espressif depuis plus de quatre ans. Ses domaines d'intervention concernent les systèmes, la sécurité, l'apprentissage automatique et l'univers passionnant de l'Internet des objets. Aditya est passionné par les nouveaux développements dans le domaine de la sécurité et par leur utilisation pour créer des applications robustes et sécurisées pour l'IdO.

clients à mettre en place des fonctions de sécurité comme l'amorçage sécurisé et le chiffrement de la mémoire flash, mais sans compromettre la sécurité. Le processus de provisionnement préalable des modules d'Espressif permet une programmation sécurisée des certificats de l'appareil. Espressif est également une autorité d'attestation de produit approuvée par la CSA (Connectivity Standards Alliance) et peut provisionner au préalable

les puces et les modules avec les certificats d'attestation de dispositif (DAC) nécessaires pour construire des dispositifs conformes au protocole Matter. De plus, Espressif facilite la programmation de micrologiciels personnalisés et la préprogrammation de données uniques, ce qui simplifie grandement la fabrication de dispositifs IdO. ◀

VF : Pascal Godart — 230638-04

LIEN

[1] Manufacturing utility: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/storage/mass_mfg.html

Révolutionnez votre gamme de produits avec WiFi Motion™,
désormais disponible sur les puces Wi-Fi Espressif.



Explorez l'avenir de la détection de mouvement.

Contactez-nous dès maintenant sur www.cognitivesystems.com

COGNITIVE ∞

une vie plus confortable et plus facile



un projet amateur basé sur le module ESP8266 Espressif

Contribué par Transfer Multisort Elektronik Sp. z.o.o.

L'idée de SmartHome devient de plus en plus populaire chaque année et les solutions qui nous aident à gérer plus efficacement notre espace de vie deviennent progressivement plus disponibles. De plus, il existe également sur le marché des produits qui permettent d'adapter des appareils encore plus anciens aux évolutions technologiques. Le contrôle à distance des appareils domestiques et l'automatisation de nombreux processus permettent d'augmenter l'efficacité énergétique, de prendre soin de l'environnement, d'augmenter le confort et de réduire les coûts. Ceci est confirmé par le projet Smart ESP8266 remote développé dans le cadre du concours de plateforme TechMasterEvent.

Espressif est un fabricant de systèmes SoC et de modules de transmission sans fil populaires qui sont disponibles chez TME. Grâce à leur taille compacte et leur faible consommation d'énergie, les produits Espressif peuvent être utilisés avec succès aussi bien dans l'électronique grand public que dans les systèmes industriels.

Ci-dessous, nous présentons une description de l'appareil basé sur le module ESP8266. Il s'agit d'un projet amateur créé par un participant au concours de la plateforme TechMasterEvent. La tâche du concours était de créer des projets électroniques qui facilitent la vie. Le module ESP8266 et de nombreux autres éléments utiles dans les projets IdO (ordinateurs monocartes, modules de communication et modules de mémoire, écrans et bien d'autres) sont disponibles sur [1].


ESP8266, LED IR et récepteur IR

L'objectif du projet Smart ESP8266 remote était de faciliter le fonctionnement des appareils que l'on retrouve dans chaque maison. Grâce à l'utilisation de la puce ESP8266, d'une diode émettrice et d'un récepteur infrarouge, ce projet permet d'éviter l'utilisation de plusieurs télécommandes pour faire fonctionner divers appareils tels que les climatiseurs et les téléviseurs. Smart ESP8266 remote se connecte à une application téléphonique, ce qui facilite l'utilisation des appareils par les utilisateurs et leur permet d'enregistrer les signaux envoyés par leurs télécommandes actuelles pour une utilisation ultérieure.

En plus de sa commodité et de sa facilité d'utilisation, le projet Smart ESP8266 remote constitue donc une excellente solution pour les appareils plus anciens qui ne sont pas

compatibles avec la technologie traditionnelle de maison intelligente. Grâce à la fonction de lecture et d'écriture des signaux envoyés par les télécommandes traditionnelles, il permet de faire fonctionner des appareils plus anciens qui ne peuvent pas se connecter à Internet ou à d'autres systèmes de maison intelligente. En conséquence, nous n'avons pas besoin de mettre à niveau des appareils plus anciens ou d'acheter des appareils intelligents coûteux, ce qui permet d'économiser de l'argent. La diode émettrice infrarouge et le récepteur infrarouge sont utilisés pour envoyer et recevoir des signaux infrarouges utilisés pour faire fonctionner les appareils de nos maisons. Ce projet permet de prendre en charge des appareils plus anciens qui ne peuvent pas se connecter à Internet ou à d'autres systèmes smart home.

En plus des éléments structurels, le projet *Smart ESP8266 remote* a également besoin d'un logiciel pour fonctionner correctement. Vous les trouverez sur [2].

Smart ESP8266 remote offre un certain nombre d'avantages tels que la commodité et la simplicité d'utilisation, des dépenses réduites et la polyvalence. Grâce à cela, nous n'avons pas besoin d'acheter des appareils intelligents coûteux ou d'améliorer des appareils plus anciens, réduisant ainsi les coûts. Cette conception peut être facilement modifiée ou adaptée pour fonctionner avec divers appareils et protocoles IR, ce qui en fait une solution polyvalente pour prendre en charge une variété d'appareils. 

230656-04

LIENS

[1] TME shop : <https://tme.eu>

[2] Code source pour ce projet : <https://techmasterevent.com/project/how-to-make-old-devices-smarter-with-a-esp8266>

MACNICA

ATD EUROPE

Your official authorized distributor
in Europe for Espressif Systems



ESPRESSIF



**empowered
connectivity
everywhere**

Macnica ATD Europe

+49 (0)89 899 143-11

sales.mae@macnica.com



www.macnica-atd-europe.com

comment construire des applications IoT sans expertise logicielle

avec la plateforme IoT Blynk et le matériel Espressif

Contribué par Blynk Inc.

Imaginez développer une application mobile sans coder, la personnaliser et la publier en un mois. Lancer un logiciel IoT de qualité professionnelle sans recruter d'ingénieurs logiciels ? Avec Blynk IoT, c'est possible en un mois et non en années !

Qu'est-ce qui est inclus dans la plateforme IoT Blynk ?

Blynk est une solution IoT low-code offrant le cloud, des bibliothèques de firmware, un constructeur d'application mobile sans code, et une console web pour tout gérer. Dès le départ, vous bénéficiez de la mise en service WiFi, de la visualisation de données, des automatisations, des notifications, des mises à jour OTA, ainsi que d'un système solide de gestion d'utilisateurs et d'appareils [1].

Constructeur d'Application Blynk pour iOS et Android

Il permet de créer rapidement des prototypes et des applications complètes sans compétences en codage. En mode constructeur, sélectionnez parmi plus de 50 éléments UI personnalisables et glissez-les pour créer une interface utilisateur (UI) unique pour votre produit connecté. Personnalisez images, polices, couleurs et icônes.

Constructeur de tableau de bord web

Il offre une architecture similaire pour créer des visualisations de données en temps réel, pour contrôler et surveiller les appareils. Créez des interfaces indépendantes pour mobile et web selon vos besoins.

La bibliothèque de firmware Blynk prend en charge :

- ESP32
- ESP32-S2
- ESP32-S3
- ESP32-C3
- ESP8266
- et d'autres



Figure 1. Interfaces No-Code créées avec Blynk.

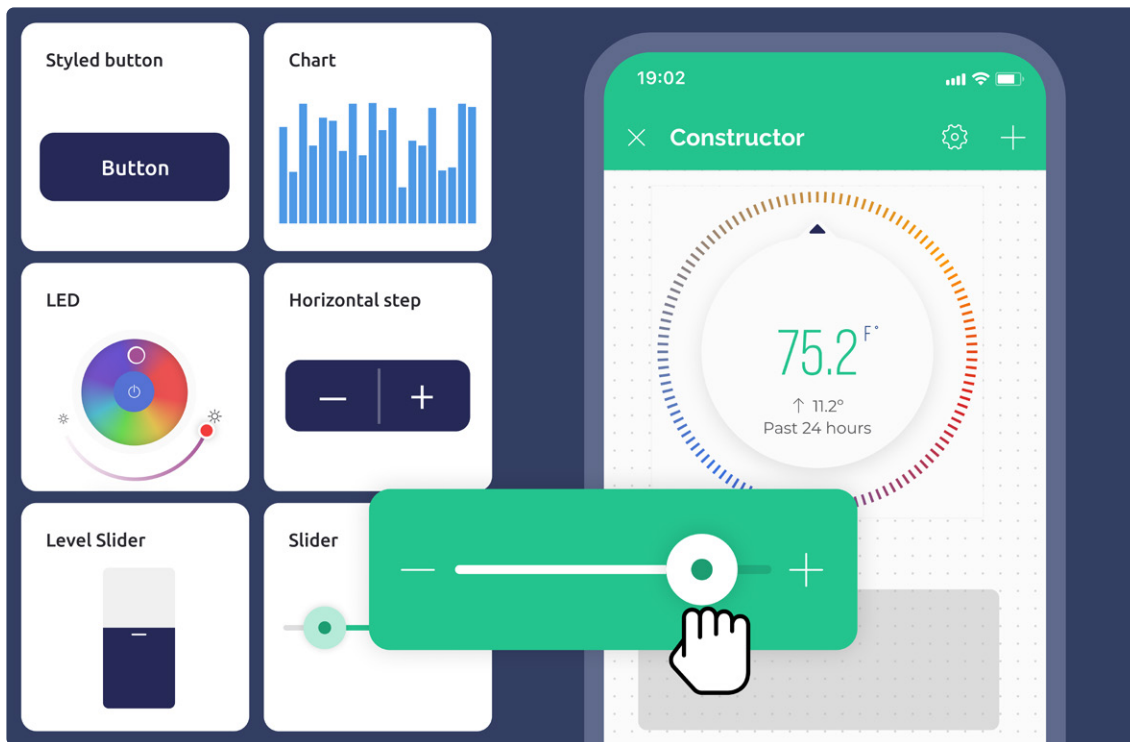


Figure 2. Constructeur d'application Blynk par glisser-déposer.

Système avancé de gestion des utilisateurs

Ce système permet de tout structurer, même à l'échelle de l'entreprise. Vous pouvez créer une structure d'organisation à plusieurs niveaux et gérer les appareils et les rôles des utilisateurs, les autorisations, les mots de passe et bien plus encore.

Gestion intégrée du cycle de vie des appareils

Cette fonctionnalité couvre tout ce qui est lié à la gestion des tokens, la mise en service WiFi, l'ajout d'appareils et leur affectation aux utilisateurs. Elle offre des mises à jour de firmware OTA fiables et sécurisées.

Automatisations sans Code

Peuvent être définis selon la date, l'heure, les actions des utilisateurs ou l'état de l'appareil. Informez les utilisateurs d'événements importants via des notifications push, des in-app, des emails ou des SMS.

Comment connecter votre ESP à Blynk ? Quel est l'effort d'intégration ?

Selon votre configuration, choisissez *Blynk.Edgent* [3] pour les appareils à microcontrôleur (MCU) unique. Si vous déchargez la connectivité sur un MCU secondaire, optez pour *Blynk.NCP* [4][5]. Les deux méthodes nécessitent un effort d'implémentation minimal avec des exemples fournis par Blynk. Pour la configuration à double MCU, utilisez un binaire prêt pour le NCP et une bibliothèque pour le MCU principal. Votre parcours, de la configuration de l'appareil à une infrastructure IoT complète, peut prendre seulement quelques semaines [6].

230659-04

Obtenez 30 % de réduction sur le plan Blynk PRO pour la première année !

Code promo : **ELEKTOR**

Valable jusqu'au 31 janvier 2024 [2]

LIENS

[1] Site officiel : <https://bit.ly/blynk-io>

[2] Blynk.Console - créez votre compte gratuit : <https://bit.ly/web-cloud>

[3] Documentation Blynk.Edgent : <https://bit.ly/docs-edgent>

[4] Documentation Blynk.NCP : <https://bit.ly/docs-ncp>

[5] Qu'est-ce que Blynk.NCP : <https://bit.ly/info-ncp>

[6] Projet de station météo prêt à l'emploi pour expérimenter : <https://bit.ly/blueprint-weather>

un distributeur à valeur ajoutée de solutions IoT et plus encore

Source : Adobe Stock

contribué par Stelium Technology

Stelium Technology est une entreprise innovante spécialisée dans les solutions électroniques. Elle se distingue par son expertise en ingénierie et sa passion pour l'innovation. Stelium Technology, à travers son partenaire Espressif, fournit des composants électroniques essentiels à la connectivité sans fil et à l'IoT, tels que l'ESP32-C5, l'ESP32-C6, l'ESP32-P4, l'ESP32-S6 et bien d'autres produits.

Fondée en 2018, Stelium Technology [1] se définit comme un distributeur à valeur ajoutée de solutions électroniques. Interfaces Hommes-Machines, écrans et solutions tactiles, connectivité et IoT, autant de domaines d'expertises largement maîtrisés par Stelium et qui lui octroient une réputation déjà bien établie sur le marché de l'électronique.

Stelium Technology est reconnue pour ses partenariats stratégiques avec des entreprises de pointe dans l'électronique, lui permettant de renforcer son positionnement dans les domaines de l'IoT et de la connectivité.

Espressif & Stelium Technology entretiennent une relation de partenariat historique, qui ne cesse de se renforcer au fil des années. En tant que distributeur officiel pour la France et l'Italie, Stelium Technology est le guichet unique pour les solutions Espressif.

Stelium Technology est parfaitement équipée pour offrir un support complet de toute la gamme des produits Espressif, à la fois au niveau hardware et software

embarqué. L'équipe possède une expertise technique approfondie qui couvre tous les aspects de la connectivité, de la conception matérielle à la programmation logicielle. Cela signifie que Stelium Technology est en mesure de fournir une assistance complète aux clients, garantissant ainsi des solutions de connectivité robustes et performantes. Ce partenariat solide garantit à nos clients un accès privilégié aux meilleures solutions de connectivité sur le marché, comme les dernières générations Espressif : ESP32-C5, ESP32-C6, ESP32-P4, ESP32-S6. Stelium Technology à travers son partenaire Espressif fournit des composants électroniques essentiels à la connectivité sans fil et à l'IoT dans une variété de marchés et de secteurs, contribuant ainsi à l'évolution constante de la technologie.

Solutions IoT et bien plus encore

Tout d'abord, dans le domaine de l'Internet des objets (IoT), les microcontrôleurs Wi-Fi et Bluetooth d'Espressif sont largement utilisés. Ces composants sont essentiels



pour les applications domestiques intelligentes, comme les thermostats connectés, les caméras de sécurité, et les dispositifs de contrôle d'éclairage. Elles jouent un rôle prépondérant dans l'interopérabilité dans les produits connectés de la maison avec des solutions MATTER compatibles (au travers le Wi-Fi, le thread notamment). Les solutions d'Espressif sont également utilisées dans le secteur industriel pour la surveillance à distance, la collecte de données, et le contrôle des machines. En outre, les produits d'Espressif sont présents dans le secteur de la santé, où ils alimentent des appareils médicaux connectés et des dispositifs de suivi de la condition physique. En tant que partenaire électronique global, Stelium a la capacité de proposer des solutions intégrant les produits d'Espressif pour le contrôle des écrans tactiles. Forts de son expertise, Stelium Technology est en mesure de concevoir des solutions intégrées dans lesquelles les produits d'Espressif contrôlent l'écran tactile, avec à notre actif plusieurs succès stories, notamment pour des tailles d'écran de 7 pouces. Notre capacité à offrir une solution globale est renforcée par un support technique spécifique couvrant l'ensemble de ces domaines. ◀

230661-04

Des demandes ?

Stelium est à la disposition des clients pour toutes demandes. Merci de contacter [remi.krief@stelium-technology.com](mailto:krief@stelium-technology.com) pour toute demande relative aux solutions Espressif.

LIEN

[1] Stelium Technology : <https://stelium-technology.com>

The Next Era of
Microcontrollers



High Performance MCU

With RISC-V Dual-Core Upto 400MHz

AI
Acceleration

High-Speed
MEMORY

Powerful
IMAGE & VOICE
Processing Capabilities

HMI Capabilities

- MIPI-CSI with ISP
- MIPI-DSI - 1080P
- Capacitive Touch
- H.264 Encoding - 1080P@30fps
- Pixel Processing Accelerator

Best-in-Class Security

- Cryptographic Accelerators
- Secure Boot, Flash Encryption
- Private Key protection
- Access Controls



Connectivity

- USB2.0 High Speed
- Ethernet
- SPI
- SDIO3.0
- UART
- I2C, I2S
-



IP Camera



Touch Panel



Video Door bell



Robotic Control



Industrial Robot

www.espressif.com



Learn More About
ESP SoCs

développement IoT rapide et facile avec M5Stack

Contribué par M5Stack

En tant que plateforme de développement IoT (Internet of Things, en français IdO = Internet des Objets) de renommée mondiale basée sur ESP32, le M5STACK permet de réaliser des centaines de contrôleurs, de capteurs, d'actionneurs et de modules de communication de façon modulaire, pouvant être connectés à l'aide d'interfaces standards. En empilant des modules de fonctionnalités diverses, les utilisateurs peuvent ainsi rapidement développer et vérifier leurs produits.

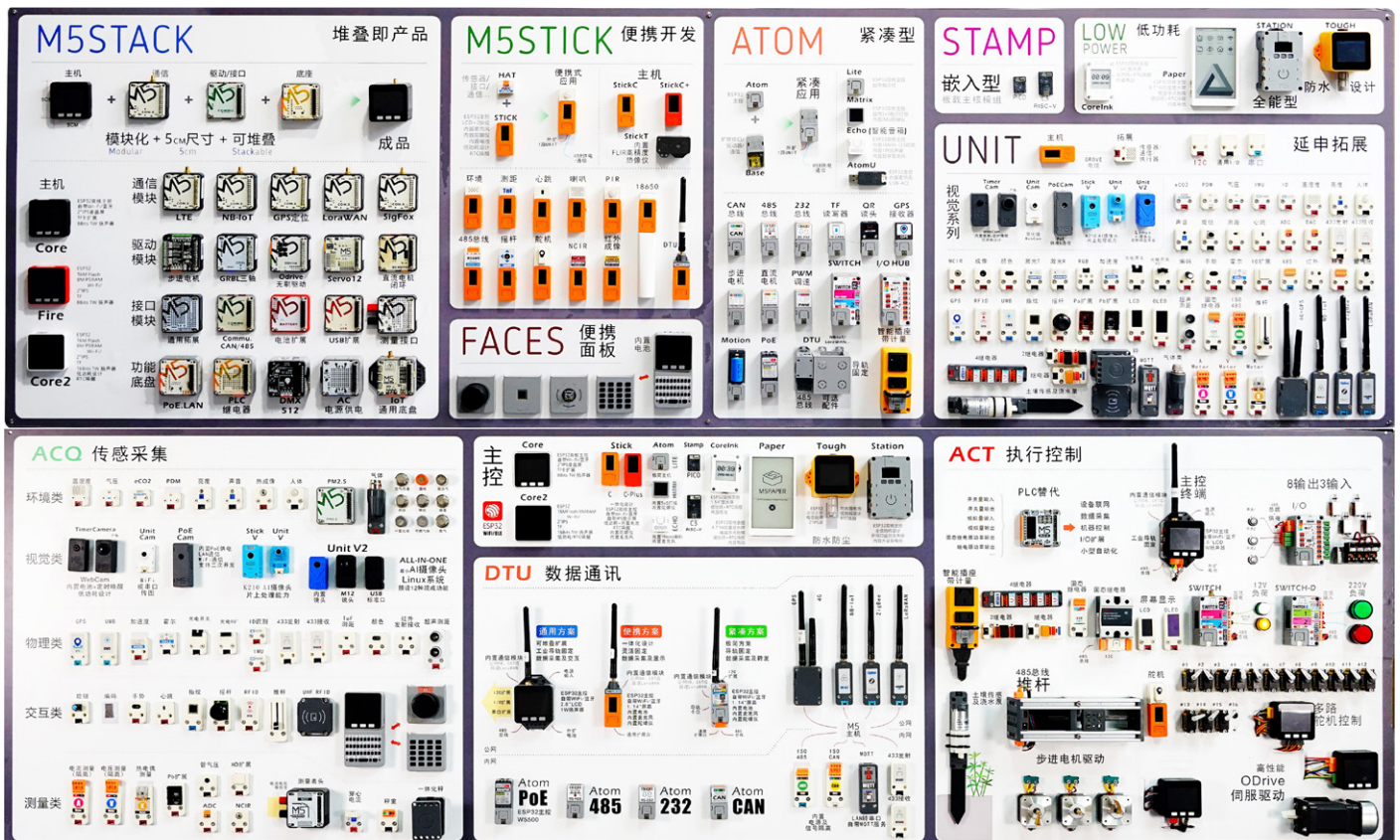


Figure 1. Famille Eco-system M5Stack.



Figure 2. Le M5Dial est adapté à la maison intelligente.



Si vous êtes un fan de l'ESP32, alors le M5Stack est indispensable !

Autant pour les débutants que pour les développeurs professionnels, les modules M5Stack (figure 1) [1] peuvent être connectés et gérés par l'IDE (Integrated Development Environment, en français EDI = environnement de développement intégré) de programmation graphique « UIFlow », ne nécessitant que peu de code tout en offrant une expérience optimale de prototypage de projets IoT.

Grâce aux modules matériels empilables ainsi qu'à la plateforme de programmation graphique conviviale, M5Stack propose aux développeurs des secteurs de l'IoT industriel, de la domotique, de la vente numérique, de l'agriculture intelligente et de l'éducation STEM (acronyme de science, technology, engineering, and mathematics, en français STIM = Science, Technologie, Ingénierie et Mathématiques), une programmation efficace et fiable, à l'aide d'une méthode rapide et facile (Quick & Easy).

Nouveau : le M5Dial

Le M5Dial [2], lancé récemment, est un produit très adapté à la maison intelligente (= maison équipée d'un système qui connecte des objets, des appareils et des systèmes entre eux, permettant de les gérer directement depuis une application installée sur un smartphone, une tablette ou une montre

intelligente). Le M5Dial, une carte de développement polyvalente, embarquée, intègre les diverses fonctionnalités et capteurs nécessaires pour le contrôle d'une maison connectée (figure 2).

Le contrôleur principal du M5Dial est un M5StampS3, un microcontrôleur basé sur la puce ESP32-S3, connue pour ses hautes performances et sa faible consommation d'énergie. Il prend en charge les communications Wi-Fi et Bluetooth, ainsi que de multiples interfaces pour périphériques telles que SPI, I2C, UART, ADC, etc. Le M5StampS3 est également doté d'une mémoire flash intégrée de 8 Mo, offrant ainsi un espace de stockage suffisant pour l'utilisateur.

Le M5Dial dispose d'un écran tactile TFT (Thin-Film-Transistor, en français Transistor à Couche Mince) rond de 1.28 pouces, d'un encodeur rotatif, d'un module de détection RFID (Radio Frequency IDentification, en français IFR = Identification par Fréquence

Radio), d'un circuit RTC (Real Time Clock, en français HTR = Horloge Temps Réel), d'un buzzer (petit vibreur), de boutons mécaniques et d'autres fonctionnalités, permettant aux utilisateurs de mettre facilement en œuvre divers projets.

La caractéristique principale du M5Dial est son encodeur rotatif, qui enregistre avec précision la position et l'orientation du bouton, apportant ainsi à l'utilisateur un contrôle interactif amélioré. Avec ce bouton rotatif, l'utilisateur peut régler des paramètres comme le volume, la luminosité, parcourir les menus, ou contrôler des équipements domestiques tels que des éclairages, des rideaux, la climatisation, etc. L'écran intégré de l'appareil peut également afficher différents effets en couleurs. Avec ses dimensions compactes de 45 mm × 45 mm × 32.2 mm et son poids léger de 46.6 g, le M5Dial est très facile à mettre en œuvre.

Qu'il soit utilisé pour contrôler des appareils ménagers dans le cadre de la maison intelligente ou pour surveiller et contrôler des systèmes dans le domaine de l'automatisation industrielle, le M5Dial peut être facilement intégré, en proposant un contrôle intelligent par l'intermédiaire d'une sélection interactive des fonctionnalités disponibles. ◀

230662-04

LIENS

[1] The Innovator of Modular IoT Development Platform | M5Stack: <https://m5stack.com>

[2] ESP32-S3 Smart Rotary Knob w/ 1.28" Round Touch Screen: <https://shop.m5stack.com/products/m5stack-dial-esp32-s3-smart-rotary-knob-w-1-28-round-touch-screen>

concevoir une Interface Graphique sur ESP32

contribué par Slint

Les smartphones ont redéfini l'expérience utilisateur des interfaces utilisateur tactiles. La construction d'une interface utilisateur moderne nécessite l'utilisation de bibliothèques graphiques et d'outils modernes. Dans cet article, nous partagerons des conseils et présenterons Slint, une boîte à outils permettant de créer des UI interactives qui répondent et dépassent les attentes des utilisateurs.



Figure 1. Logos C++ et Rust.

À propos de Slint – une boîte à outils de nouvelle génération pour construire des UI graphiques natives en C++, Rust et JavaScript, avec un support multiplateforme, et plus de 10 000 étoiles sur GitHub.

Choisissez un langage de programmation - C/C++ ou Rust

En programmation embarquée, C et C++ ont été les langages de programmation préférés pendant longtemps. Mais Rust, réputé pour sa sécurité et ses performances, devient populaire parmi les développeurs embarqués.

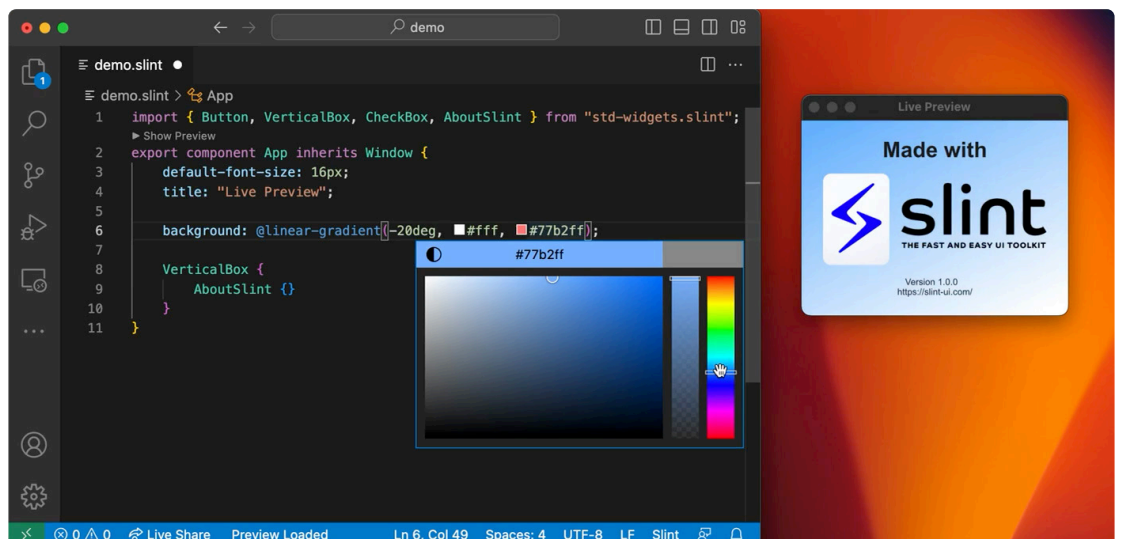


Figure 2. Itérations rapides avec l'aperçu en temps réel de Slint (Live-Preview).

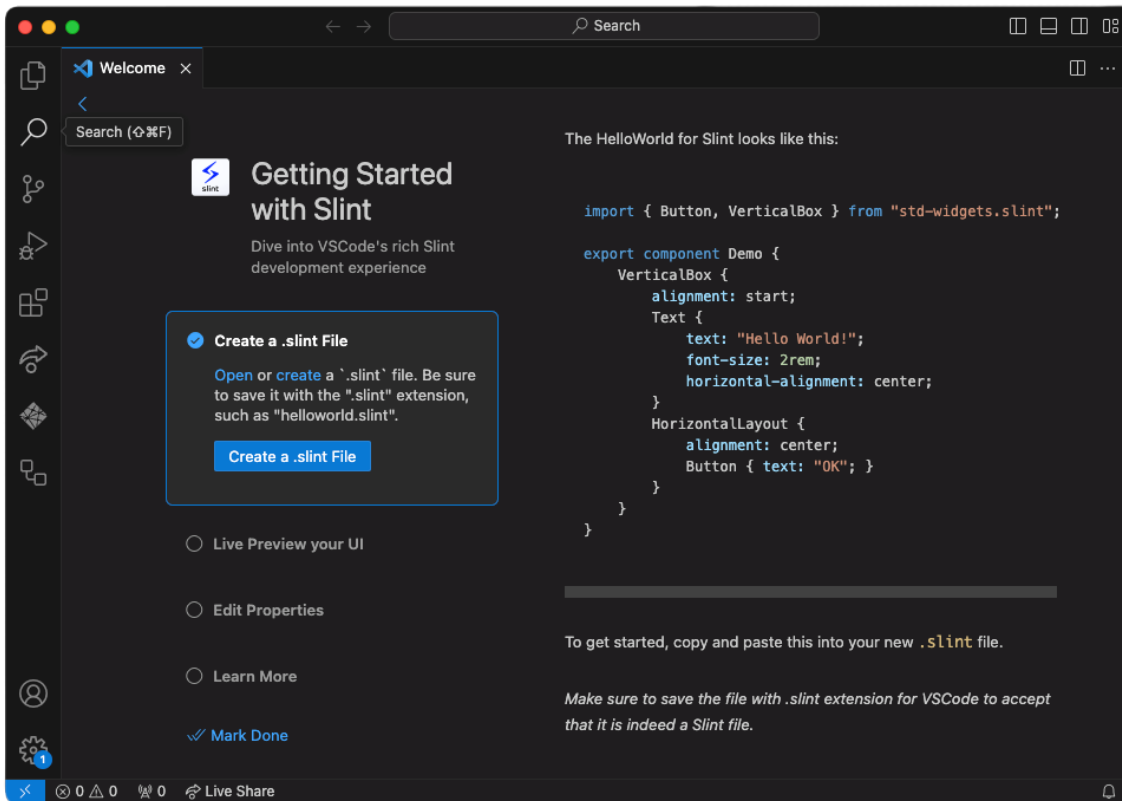


Figure 3. Démarrer avec Slint.

Slint, la seule boîte à outils offrant des API natives à la fois pour C++ et Rust (**figure 1**), offre aux développeurs le choix : écrivez votre logique métier dans l'un ou l'autre de ces langages. De plus, il offre un chemin de transition pour ceux qui souhaitent passer leur code de C/C++ à Rust.

Séparez l'interface utilisateur de la logique métier

Dans Slint, l'interface utilisateur est définie à l'aide d'un langage similaire à HTML/CSS, favorisant une stricte division entre la présentation et la logique métier. Finalisez votre conception d'interface utilisateur grâce à des itérations rapides avec l'aperçu en temps réel de Slint (Live-Preview) (**figure 2**).

Profitez d'une bonne expérience de développement

La complexité actuelle du développement logiciel exige de bons outils.

Profitez de l'autocomplétion, de la coloration syntaxique, des diagnostics, de l'aperçu en direct, et bien plus encore (**figure 3**). De plus, Slint offre un composant ESP-IDF, simplifiant son intégration avec le framework de développement IoT d'Espressif.

Offrez une expérience utilisateur exceptionnelle

La performance de l'interface graphique est cruciale pour une (UX) exceptionnelle. Profitez de la flexibilité de Slint avec les capacités de rendu ligne par ligne ou sur framebuffer (**figure 4**).

Essayez Slint sur ESP32, visitez [1]. [▶](#)

230670-04



Figure 4. Démo Slint sur ESP32.

LIEN

[1] Slint sur ESP32 : <https://slint.dev/esp32>

Mettez la main sur le nouveau matériel ESPRESSIF



Il n'y a rien qui nous excite plus que de mettre la main sur du nouveau matériel, et cette collaboration avec Espressif a été un vrai régal ! Vous voulez en faire l'expérience vous-même ? Elektor a approvisionné ses magasins pour offrir tous les produits présentés dans cette édition !



ESP32-S3-Box-3

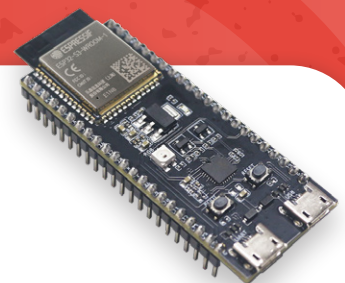
L'ESP32-S3-BOX-3 est un kit de développement AIoT entièrement open-source basé sur le puissant SoC AI ESP32-S3, et conçu pour révolutionner le domaine des cartes de développement ordinaires. L'ESP32-S3-BOX-3 est livré avec un ensemble riche de modules complémentaires, permettant aux développeurs de personnaliser et d'étendre facilement les applications de ce kit.

www.elektor.fr/20627

ESP32-S3-Eye

L'ESP32-S3-EYE est une carte de développement IA de petit format. Elle est basée sur le SoC ESP32-S3 et sur l'ESP-WHO, le cadre de développement IA d'Espressif. Elle comporte un appareil photo de 2 mégapixels, un écran LCD et un microphone, utilisés pour la reconnaissance d'images et le traitement audio.

www.elektor.fr/20626



ESP32-S3-DevkitC-1

L'ESP32-S3-DevKitC-1 est une carte de développement pour débutants équipée de l'ESP32-S3-WROOM-1, l'ESP32-S3-WROOM-1U ou l'ESP32-S3-WROOM-2, un module microcontrôleur polyvalent Wifi + Bluetooth Low Energy qui intègre des fonctions Wifi et Bluetooth LE complètes.

www.elektor.fr/20697



ESP32-Cam-CH340

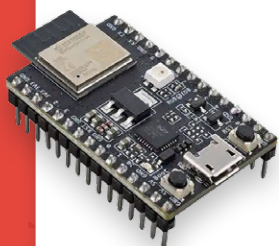
La carte de développement ESP32-Cam-CH340 est adaptée aux applications IoD, telles que les appareils intelligents, le contrôle industriel sans fil, la surveillance sans fil, la lecture de code QR sans fil, les signaux de systèmes de positionnement sans fil et d'autres applications de l'Internet des objets.

www.elektor.fr/19333

ESP32-C3-DevKitM-1

ESP32-C3-DevKitM-1 est une carte de développement pour débutants basée sur l'ESP32-C3-MINI-1, un module nommé ainsi en raison de sa petite taille. Cette carte intègre des fonctions Wifi et Bluetooth LE complètes. La plupart des broches d'E/S du module ESP32-C3-MINI-1 sont disponibles sur les connecteurs des deux côtés de la carte pour faciliter l'interfaçage. Les développeurs peuvent connecter les périphériques avec des fils de connexion ou monter l'ESP32-C3-DevKitM-1 sur une plaque d'essai.

www.elektor.fr/20324

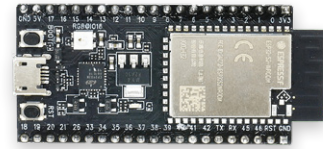




Elektor Cloc 2.0 Kit

Cloc est un réveil basé sur l'ESP32, facile à construire, qui se connecte à un serveur de temps et contrôle la radio et la TV. Il est doté d'un afficheur rétro double à 7 segments avec une luminosité variable. L'un des écrans affiche l'heure actuelle, l'autre l'heure de l'alarme.

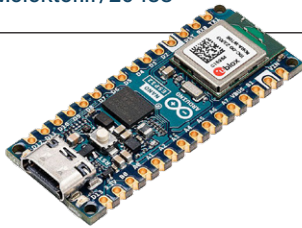
www.elektor.fr/20438



ESP32-S2-Saola-1M

L'ESP32-S2-Saola-1M est une carte de développement de petite taille basée sur l'ESP32-S2. La plupart des broches d'E/S sont disponibles sur les connecteurs des deux côtés pour faciliter l'interfaçage. Les développeurs peuvent connecter les périphériques avec des fils de connexion ou monter l'ESP32-S2-Saola-1M sur une plaque d'essai. L'ESP32-S2-Saola-1M est équipée du module ESP32-S2-WROOM, un puissant module microcontrôleur Wifi générique doté d'un riche ensemble de périphériques.

www.elektor.fr/19694



Arduino Nano ESP32

L'Arduino Nano ESP32 est une carte au format de l'Arduino Nano basée sur l'ESP32-S3 (intégré dans le NORA-W106-10B d'u-blox). Il s'agit de la première carte Arduino entièrement basée sur un ESP32. Elle offre aussi les fonctions Wifi, Bluetooth LE, le débogage via l'USB natif dans l'EDI Arduino ainsi qu'une faible consommation d'énergie.

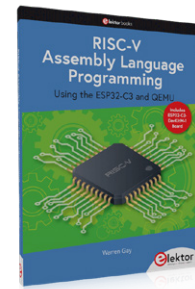
www.elektor.fr/20562



MakePython ESP32 Development Kit

Le kit MakePython ESP32 est un kit de développement indispensable pour la programmation MicroPython de l'ESP32. En plus de la carte de développement MakePython ESP32, le kit contient les composants électroniques de base et les modules dont vous avez besoin pour commencer à programmer. Avec les 46 projets du livre accompagnant le kit, vous pouvez vous attaquer à des projets électroniques simples avec MicroPython sur ESP32 et réaliser vos propres projets IoT.

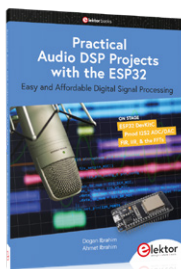
www.elektor.fr/20137



RISC-V Assembly Language Programming using ESP32-C3 and QEMU (+ carte ESP32 RISC-V GRATUITE)

La puce ESP32-C3 d'Espressif permet d'acquérir une expérience pratique de RISC-V. L'émulateur QEMU à source ouverte permet d'acquérir une expérience de RISC-V 64 bits sous Linux. Les étudiants et les passionnés trouveront dans ce livre deux façons d'explorer RISC-V. Les projets de ce livre sont réduits à l'essentiel pour que les concepts du langage assembleur restent clairs et simples.

www.elektor.fr/20296



Practical Audio DSP Projects with the ESP32

L'objectif de ce livre est de présenter les principes de base du traitement numérique du signal (DSP) et de l'introduire d'un point de vue pratique en utilisant un minimum de calcul. Seul le niveau de base de la théorie des systèmes à temps discret est abordé, ce qui est suffisant pour réaliser des applications DSP en temps réel. Les applications pratiques utilisent la fameuse carte de développement à microcontrôleur ESP32 DevKitC et sont décrites en temps réel.

www.elektor.fr/20558

MicroPython for Microcontrollers

Les microcontrôleurs puissants tels que l'ESP32 offrent d'excellentes performances ainsi que des fonctions Wifi et Bluetooth à un prix abordable. Grâce à ces caractéristiques, la scène des maker a été prise d'assaut. Par rapport à d'autres contrôleurs, les capacités des mémoires flash et SRAM de l'ESP32 beaucoup plus grandes et la vitesse de l'unité centrale est plus élevée. Grâce à ces caractéristiques, la puce convient non seulement aux applications en C classiques, mais aussi à la programmation en MicroPython. Ce livre présente les applications des systèmes monopuces modernes.

www.elektor.fr/19736

