

Gastredactie:



ESPRESSIF

Vrijgegeven
Bonuseditie!

Domotica met Espressif chips

ADF, IDF en andere SDKs



Try it with
ESP Launchpad



p. 4

Zo combineer
je **ChatGPT**
met de SOC's
van Espressif

ESP32 en ChatGPT



p. 14

Een tweefactor-
authenticatie-
dongle met de
ESP32-C3



Elke week
nieuwe gratis
bonusartikelen!



ESP32-C3 module stuurt
dekatron aan

p. 18



Interview:
Home Assistant grondlegger
Paulus Schoutsen

p. 24



Flas...cient
en v...

p. 30



Design-In Expertise & Service

YOUR PARTNER FOR



ESPRESSIF

Ineltek

Leading Espressif franchise distribution partner

Experience

36 years of experience in the semiconductor industry

Latest
News

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

Innovation

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

Application
Support

Dedicated and focused in-house support offering

Time
to
Market

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

Information

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

Supply

Popular Espressif SoCs, modules and EVKs available from stock

CONTACT US FOR PARTS & SUPPORT



www.ineltek.com

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

Founded in 1987, Ineltek 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@ineltek.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



Elektor verschijnt acht keer per jaar en is een uitgave van

Elektor International Media B.V.
Postbus 11, 6114 ZG Susteren (Nederland)
Tel.: +31 (0)46 4389444

www.elektor.nl | www.elektormagazine.nl

Voor al uw vragen: service@elektor.nl

Lid worden: www.elektormagazine.nl/abo

Advertenties

Raoul Morreau
Tel. +31 (0)6 4403 9907
raoul.morreau@elektor.com
www.elektormagazine.nl/adverteren

Auteursrecht

© Elektor International Media B.V. - 2023

Niets uit deze uitgave mag verveelvoudigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm of op welke wijze dan ook, zonder voorafgaande schriftelijke toestemming van de uitgever. De auteursrechtelijke bescherming van Elektor strekt zich mede uit tot de illustraties met inbegrip van de printed circuits, evenals de ontwerpen daarvoor. In verband met artikel 30 van de Rijksoctrooiwet mogen de in Elektor opgenomen schakelingen slechts voor particuliere of wetenschappelijke doeleinden vervaardigd worden en niet in of voor een bedrijf. Het toepassen van de schakelingen geschiedt buiten de verantwoordelijkheid van de uitgever. De uitgever is niet verplicht ongevraagd ingezonden bijdragen, die hij niet voor publicatie aanvaardt, terug te zenden. Indien de uitgever een ingezonden bijdrage voor publicatie aanvaardt, is hij gerechtigd deze op zijn kosten te (doen) bewerken. De uitgever is tevens gerechtigd een bijdrage te (doen) vertalen en voor haar andere uitgaven en activiteiten te gebruiken tegen de daarvoor bij de uitgever gebruikelijke vergoeding.

Druk

Senefelder Misset, Mercuriusstraat 35,
7006 RK Doetinchem (Nederland)

Distributie

Betapress, Nederland – AMP, België

Internationaal hoofdredacteur

Jens Nickel

Content Director

C. J. Abate

Directeur

Erik Jansen



Vrijgegeven Bonuseditie!

In de voorbije maanden heeft het contentteam van Elektor nauw samengewerkt met Espressif om diepgravende artikelen te maken over een groot aantal spannende onderwerpen. Al dat harde werk heeft geresulteerd in een gasteditie van Espressif voor Elektor Mag, die we begin december 2023 hebben gepubliceerd. Maar we zijn nog niet klaar met samenwerken. Deze bonuseditie van Elektor Mag staat boordevol extra projecten en tutorials die je maandenlang zullen inspireren.

Op de typische Elektor-wijze heeft deze bonuseditie voor elk wat wils, van professionele ingenieurs die geïnteresseerd zijn in het ontwikkelen

van AIoT-producten tot makers die op zoek zijn naar creatieve weekendprojecten. Elektor en Espressif geven tips voor het implementeren van ChatGPT met Espressif SOC's, een leuk Dekatron-project, advies voor het bouwen van IoT-apps zonder software-expertise, en nog veel meer. Veel plezier!

Als je aan je volgende Espressif-project begint, deel dan je ervaringen met de gebruikersgemeenschap op het Elektor Labs onlineplatform op elektormagazine.com/labs - we zijn benieuwd naar je creaties!

C. J. Abate (Content Director, Elektor)



IN DIT NUMMER

4 Ontketen de kracht van OpenAI en ESP-BOX

Zo combineer je ChatGPT met de SOC's van Espressif



14 ESP-Unlock

Een tweefactor-authenticatie-dongle op basis van de ESP32-C3



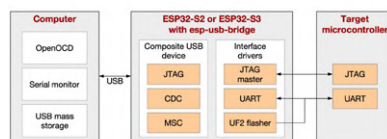
18 Dekatron

Een stukje geschiedenis komt tot leven!



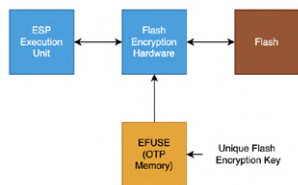
24 De Smart Home-revolutie

Paulus Schoutsen over Home Assistant, ESPHome en meer



30 **Beschikbaar in week 01**
Over het ESP32 2024

36 **Va**
Beschikbaar in week 01
2024



40 **Beschikbaar in week 01**
2024

44 **Ee** en

46 **Beschikbaar in week 01**
2024

48 **Ee** IoT en meer

50 **Sn** Stack
Beschikbaar in week 01
2024

52 **mm** op ESP32

54 **Be**
Beschikbaar in week 01
2024



OpenAI

Ontketen de kracht van OpenAI en ESP-BOX

zo combineer je ChatGPT met de SOC's van Espressif

Ali Hassan Shah, Espressif

Hier verkennen we het potentieel van ChatGPT plus ESP-BOX, een ontwikkelplatform dat bestaat uit boards die van een microfoon zijn voorzien en die vergezeld gaan van een uitgebreid software-ecosysteem voor spraakherkenning en meer. Dit vormt een krachtige combinatie die IoT-apparaten naar een hoger niveau kan tillen!

De wereld is getuige van een technologische revolutie en OpenAI loopt daarbij voorop. Een van de meest opwindende ontwikkelingen is ChatGPT, dat natuurlijke taal verwerkt om boeiende en intuïtieve gebruikerservaringen te creëren. De integratie van OpenAI API's met IoT-apparaten opent een wereld aan mogelijkheden.

Dit artikel is onderverdeeld in drie secties, die elk een essentieel aspect van het project behandelen. Het eerste deel gaat dieper in op het ESP-BOX ontwikkelplatform en geeft details over de eigenschappen en functionaliteit. Het tweede deel is een case study die de stappen schetst die nodig waren om het project vanaf de grond af op te bouwen. Het laatste deel bevat een lijst met aanvullende informatiebronnen die de lezer kan raadplegen om zijn kennis en begrip van het project te verdiepen.

ESP-BOX

De ESP-BOX [1] is een next-generation AIoT-ontwikkelplatform dat de ESP32-S3-BOX en ESP32-S3-BOX-Lite development boards omvat. Deze laatste zijn gebaseerd op de ESP32-S3 [2] WiFi + Bluetooth 5 (LE) SoC en bieden een flexibele en aanpasbare oplossing voor het ontwikkelen van AIoT-toepassingen waarin verschillende sensoren, controllers en gateways zijn geïntegreerd.

De ESP-BOX (**figuur 1**) heeft een groot aantal functies die hem tot een ideaal AIoT-ontwikkelplatform maken. Laten we enkele van de belangrijkste functies eens nader bekijken.

Figuur 1. ESP32-S3-BOX.



- > **Verre-velde spraakinteractie met twee microfoons:** de ESP-BOX ondersteunt spraakinteractie in het verre veld met twee microfoons, zodat gebruikers op afstand met hun apparaten kunnen communiceren.
- > **Offline herkenning van gesproken opdrachten in het Chinees en Engels met grote betrouwbaarheid:** de ESP-BOX biedt offline herkenning van spraakcommando's in zowel het Chinees als het Engels met een hoge herkenningsratio, waardoor het eenvoudig is om spraakgestuurde apparaten te ontwikkelen.
- > **Herconfigureerbare 200+ spraakcommando's in het Chinees en Engels:** ontwikkelaars kunnen desgewenst gemakkelijker meer dan 200 spraakopdrachten in het Chinees en Engels herconfigureren.
- > **Continue identificatie en wek-interrupt:** de ESP-BOX ondersteunt continue identificatie en wek-interrupt, zodat apparaten altijd klaarstaan om spraakcommando's te ontvangen.
- > **Flexibel en herbruikbaar GUI-framework:** de ESP-BOX wordt geleverd met een flexibel en herbruikbaar GUI-framework, waarmee ontwikkelaars aangepaste gebruikersinterfaces voor hun toepassingen kunnen maken.
- > **End-to-end AIoT development framework ESP-RainMaker:** de ESP-BOX is gebouwd op Espressif's end-to-end AIoT/IoT development framework ESP-RainMaker, dat ontwikkelaars alle tools biedt die ze nodig hebben om krachtige en intelligente apparaten te bouwen.

- > **Pmod-compatibele headers ondersteunen de uitbreiding van periferie-modules:** de ESP-BOX is voorzien van Pmod-compatibele headers, waardoor de mogelijkheden eenvoudig kunnen worden uitgebreid met een breed scala aan periferie-modules.

Case study

Deze case study beschrijft de ontwikkeling van een spraakgestuurde chatbot die gebruik maakt van de combinatie van ESP-BOX en OpenAI API. Het systeem accepteert gesproken commando's van gebruikers, geeft deze weer op het display en verwerkt ze via de OpenAI API's om een antwoord te genereren. Het antwoord wordt vervolgens weergegeven op het display en afgespeeld via de ESP-BOX. De stapsgewijze workflow geeft een gedetailleerde uitleg over hoe deze technologieën geïntegreerd kunnen worden om een efficiënte en effectieve spraakgestuurde chatbot te maken (zie **figuur 2** en **figuur 3**).

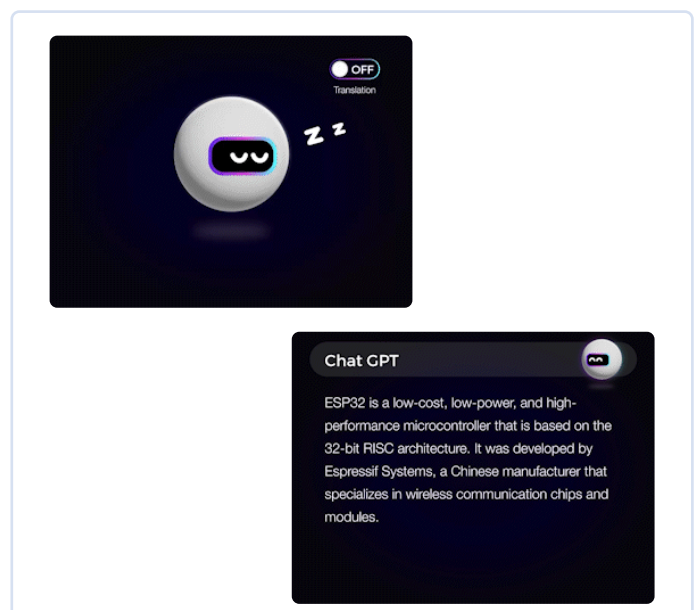
Setup van de ontwikkelomgeving

Het opzetten van een geschikte omgeving en het installeren van de juiste versie is cruciaal om fouten te voorkomen. In deze demonstratie gebruiken we ESP-IDF versie 5.0 (master branch). Als je hulp nodig hebt bij het opzetten van ESP-IDF, raadpleeg dan de officiële IDF Programming Guide [3] voor gedetailleerde informatie. Bij het schrijven van dit artikel is de huidige IDF commit head `df9310ada2`.

Om ChatGPT te gebruiken, een krachtig taalmodel gebaseerd op de GPT-3.5 architectuur, moet je eerst een beveiligde API-sleutel verkrijgen. Daartoe moet je een account aanmaken op het OpenAI-platform [4] en tokens verkrijgen door deze aan te maken of te kopen. Met een API-sleutel krijg je toegang tot een groot aantal functies en mogelijkheden die kunnen worden aangepast aan je specifieke behoeften, zoals natuurlijke taalverwerking en -generatie, tekstaanvulling en conversatie-modellering. Volg de officiële API-referentielink [5]. Het spreekt voor zich dat het handhaven van de vertrouwelijkheid en veiligheid van de API-sleutel cruciaal is om ongeautoriseerde toegang tot de account en gegevens van de gebruiker te voorkomen.



Figuur 2. Workflow van de case study.



Figuur 3. Demo van ESP-BOX als chatbot.

Offline-spraakherkenning toevoegen

Espressif Systems heeft een innovatief framework voor spraakherkenning ontwikkeld onder de naam ESP-SR [6]. Dit framework is ontworpen om apparaten gesproken woorden en zinnen te laten herkennen zonder afhankelijk te zijn van externe cloud-gebaseerde diensten, waardoor het een ideale oplossing is voor offline-spraakherkenningstoepassingen. Het ESP-SR framework bestaat uit verschillende modules, waaronder het Audio Front-end (AFE), Wake Word Engine (WakeNet), Speech Command Word Recognition (MultiNet) en Speech Synthesis (die momenteel alleen de Chinese taal ondersteunt). Volg de officiële documentatie [7] voor meer informatie.

OpenAI API integreren

De OpenAI API biedt talloze functies die ontwikkelaars kunnen gebruiken om hun toepassingen te verbeteren. In ons project hebben we de Audio-to-Text en Completion API's gebruikt en geïmplementeerd met C-code gebaseerd op ESP-IDF. De volgende paragrafen geven een kort overzicht van onze code.

Audio to text

Om tekst uit audio te extraheren, maken we gebruik van HTTPS en de OpenAI Audio API. De code van **listing 1** wordt hiervoor gebruikt. Deze code bestaat uit een functie met de naam `create_whisper_from_record()`, die een pointer naar een buffer met de audiogegevens aanneemt en een geheel getal `audio_len` dat de lengte van de audiogegevens weergeeft. Deze functie stuurt een POST-request naar het OpenAI API-endpoint om de opgenomen audiogegevens om te zetten. De functie begint met het initialiseren van de URL van de OpenAI API en het instellen van de autorisatieheaders met het token `OPENAI_API_KEY`. Vervolgens wordt een HTTP-client geconfigureerd en geïnitieerd met de opgegeven configuratie, waaronder de URL, HTTP-methode, event-handler, buffergrootte, time-out en SSL-certificaat.

Daarna worden het content type en de boundary string voor het multipart/form-data request ingesteld als headers voor de HTTP-client. De bestandsgegevens en de grootte ervan worden ook ingesteld, en een multipart/form-data verzoek wordt gebouwd. De `form_data` buffer wordt toegewezen met een `malloc()` functie, en de benodigde informatie wordt eraan toegevoegd. Dit omvat de bestandsnaam en content type van het audiobestand, de inhoud van het bestand en de naam van het model dat gebruikt zal worden voor de transcriptie. Zodra de `form_data` is gebouwd, wordt het ingesteld als het post-veld in de HTTP-client en de client stuurt het POST-verzoek naar het OpenAI API-endpoint. Als er een fout optreedt tijdens de request, logt de functie een foutmelding. Tot slot wordt de HTTP-client opgeschoond en worden de resources die waren toegewezen aan `form_data` weer vrijgegeven.

De functie retourneert een `esp_err_t` foutcode, die aangeeft of het HTTP-verzoek al dan niet succesvol was.

Chat completion

De OpenAI Chat Completion API [8] wordt gebruikt om HTTPS-requests voor chat completion te verzenden. Dit proces omvat het gebruik van de functie `create_chatgpt_request()`, die een content parameter opneemt die de invoertekst voor het GPT-3.5-model weergeeft (**listing 2**).

De functie stelt eerst de URL, het model en de headers in die nodig zijn voor het HTTP POST-request en maakt vervolgens een JSON payload met het model, het maximale aantal tokens en de inhoud. Vervolgens stelt de functie de headers in voor het HTTP request en stelt de JSON-payload in als het post-veld voor het request. Het HTTP POST-request wordt dan verzonden met `esp_http_client_perform()` en als het request mislukt, wordt er een foutmelding gelogd. Ten slotte wordt de HTTP-client opgeschoond en wordt de foutcode geretourneerd.



Listing 1. Tekst uit audio extraheren.

```
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;
}

```



De integratie van ChatGPT van OpenAI met ESP-BOX van Espressif opent een wereld aan mogelijkheden voor het bouwen van krachtige en intelligente IoT-apparaten.



Listing 2: Chat Completion HTTPS-request.

```
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;
}
```

Afhandelen van de respons

De callback-functie `response_handler` wordt gebruikt door de ESP-IDF HTTP-clientbibliotheek om events af te handelen die optreden tijdens een HTTP-request/respons-uitwisseling (**listing 3**).

In het geval van `HTTP_EVENT_ON_DATA` wijst de functie geheugen toe voor de binnenkomende gegevens, kopieert de gegevens naar de buffer en verhoogt de variabele `data_len` dienovereenkomstig. Dit wordt gedaan om de responsgegevens te verzamelen.

In het geval van `HTTP_EVENT_ON_FINISH` drukt de functie een bericht af dat aangeeft dat de HTTP-uitwisseling is beëindigd en roept dan de functie `parsing_data()` aan om de verzamelde/ruwe gegevens te verwerken. Vervolgens wordt het geheugen vrijgegeven en worden de variabelen `data` en `data_len` op nul gezet. Tenslotte retourneert de functie `ESP_OK` om aan te geven dat de bewerking succesvol was.

Ruwe gegevens parsen

De JSON `parser`-component [9] wordt gebruikt om de ruwe respons verkregen van ChatGPT API en Whisper AI API over HTTPS te parsen. Hiertoe wordt een functie gebruikt die op zijn beurt de parser-component gebruikt (**listing 4**). Meer details over deze tool kunnen worden gevonden op GitHub [10].

TTS API integreren

Op dit moment biedt OpenAI geen publieke toegang tot hun Text-to-Speech (TTS) API. Er zijn echter verschillende andere TTS API's

beschikbaar, waaronder Voicerss [11], TTSMaker [12], en TalkingGenie [13]. Deze API's kunnen spraak genereren uit tekstinput; je kunt meer informatie hierover vinden op hun respectievelijke websites.

In deze tutorial gebruiken we de TalkingGenie API, een van de beste opties voor het genereren van hoogwaardige, natuurlijk klinkende spraak in zowel het Engels als het Chinees. Een van de unieke eigenschappen van TalkingGenie is de mogelijkheid om tekst in een mix van talen, zoals Chinees en Engels, naadloos in spraak om te zetten. Dit kan een waardevol hulpmiddel zijn voor het maken van inhoud die een wereldwijd publiek aanspreekt. De code stuurt een door ChatGPT gegenereerde tekstrespons naar de TalkingGenie-API via HTTPS en speelt vervolgens de resulterende spraak af via een ESP-BOX (**listing 5**).

De functie `text_to_speech()` neemt een berichtstring en een parameter `AUDIO_CODECS_FORMAT` als invoer. De berichtstring is de tekst die in (synthetische) spraak zal omgezet, terwijl de parameter `AUDIO_CODECS_FORMAT` specificeert of de spraak moet worden gecodeerd in MP3- of WAV-formaat.

De functie codeert eerst de berichtstring met de functie `url_encode()` die enkele ongeldige tekens vervangt door hun ASCII-code en converteert die code vervolgens naar een tweecijferige hexadecimale weergave. Vervolgens wordt geheugen toegewezen voor de resulterende gecodeerde string. Het controleert dan de `AUDIO_CODECS_FORMAT` parameter en stelt de juiste codec format string in om te gebruiken in de `url`.



Listing 3: Gebeurtenissen afhandelen tijdens een HTTP request/respons uitwisseling.

```
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: Parsen van de ruwe respons van ChatGPT API en 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);
        }
    }
}
```

Vervolgens bepaalt de functie de grootte van de `url` buffer die nodig is voor een GET-verzoek aan de TalkingGenie API, en wijst dienovereenkomstig geheugen toe voor de `url` buffer. Vervolgens wordt de `url` string opgemaakt met de juiste parameters, waaronder de `voiceId` (die de te gebruiken stem specificeert), de gecodeerde tekst, de snelheid en het volume van de spraak en het audiotype (MP3 of WAV).

De functie maakt dan een `esp_http_client_config_t` struct aan met de `url` en andere configuratieparameters, initialiseert een `esp_http_client_handle_t` met de struct en voert een GET-request uit naar de TalkingGenie API met behulp van `esp_http_client_perform()`. Als het verzoek succesvol is, retourneert de functie `ESP_OK`, anders wordt een foutcode geretourneerd. Tot slot maakt de functie het geheugen vrij dat is toegewezen aan de `url`-buffer en het gecodeerde bericht, ruimt de `esp_http_client_handle_t` op en retourneert de foutcode.

Omgaan met de TTS-respons

Op dezelfde manier wordt de callbackfunctie `http_event_handler()` gedefinieerd om gebeurtenissen af te handelen die optreden tijdens een HTTP request/response uitwisseling (listing 6).

`HTTP_EVENT_ON_DATA` event wordt gebruikt om de audiogegevens af te handelen die worden ontvangen van de server. De audiogegevens

worden opgeslagen in een buffer met de naam `record_audio_buffer` en de totale lengte van de ontvangen audiogegevens wordt opgeslagen in de variabele `file_total_len`. Als de totale lengte van de ontvangen audiogegevens kleiner is dan een vooraf gedefinieerde `MAX_FILE_SIZE`, worden de gegevens gekopieerd naar de `record_audio_buffer`.

Tenslotte wordt het `HTTP_EVENT_ON_FINISH` event gebruikt om het einde van de HTTP-respons af te handelen. In dat geval wordt de `record_audio_buffer` doorgegeven aan de functie `audio_player_play()` die de audio afspeelt.

Display

Voor de weergave gebruiken we LVGL, een open-source embedded grafische bibliotheek die steeds populairder wordt vanwege de krachtige en visueel aantrekkelijke functies en het geringe geheugengebruik. LVGL heeft ook een visuele drag-and-drop UI editor uitgebracht onder de naam SquareLine Studio [14]. Het is een krachtig hulpmiddel waarmee je eenvoudig fraaie GUI's voor je toepassingen kunt maken. Om LVGL in je project te integreren, biedt Espressif Systems een officieel package manager tool [15]. Hiermee kun je LVGL en gerelateerde porting-componenten direct toevoegen aan je project, wat je tijd en moeite bespaart. Volg voor meer informatie de officiële blogs [16] en documentaties [17].



Listing 5: Tekst naar spraak.

```
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 = sprintf(NULL, 0,
        "https://dds.oui.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
    sprintf(url, url_size + 1,
        "https://dds.oui.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;
}
```

Intelligente IoT-apparaten bouwen

De integratie van OpenAI's ChatGPT met ESP-BOX van Espressif heeft nieuwe mogelijkheden geopend voor het maken van krachtige en intelligente IoT-apparaten. De ESP-BOX biedt een flexibel en aanpasbaar AIoT-ontwikkelpatform met functies zoals verre-veld spraakinteractie, offline spraakherkenning en een herbruikbaar GUI-framework. Door deze mogelijkheden te combineren met de OpenAI API kunnen ontwikkelaars spraakgestuurde chatbots maken en de gebruikerservaring bij IoT-toepassingen verbeteren.

Vergeet niet de Systems GitHub-repository [19] van Espressif [18] te bekijken voor meer open source-demo's voor ESP-IoT-Solution [20], ESP-SR en ESP-BOX. De broncode voor dit project is te vinden op GitHub [21]. Als onderdeel van onze toekomstplannen willen we een component voor de OpenAI API introduceren die gebruiksvriendelijke functies biedt. 

230462-03

Vragen of opmerkingen?

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



Over de auteur

Ali Hassan Shah is een embedded software engineer, gedreven door een diepe passie voor IoT-technologie. Als gewaardeerd lid van het Application Engineering-team van Espressif Systems kanaliseert hij zijn expertise in een missie om technologie moeiteloos toegankelijk en gebruiksvriendelijk te maken voor iedereen, onder het motto 'technologie vereenvoudigen voor iedereen.'



Gerelateerd product

> **ESP32-S3-BOX** 
www.elektor.nl/20627



WEBLINKS

- [1] ESP-Box: <https://github.com/espressif/esp-box>
- [2] ESP32-S3 product selector: <https://tinyurl.com/esp32s3prodsel>
- [3] IDF Programming Guide: <https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32/index.html>
- [4] OpenAI platform: <https://openai.com>
- [5] Officiële API referentielink: <https://platform.openai.com/docs/api-reference>
- [6] ESP-SR: <https://github.com/espressif/esp-sr>
- [7] ESP-SR User Guide: <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] Officieel package manager tool voor LVGL: <https://components.espressif.com/components/lvgl/lvgl>
- [16] Blog overLVGL: <https://tinyurl.com/esp fancyui>
- [17] Documentatie van LVGL: <https://docs.lvgl.io/master/index.html>
- [18] Espressif Systems: <https://espressif.com/>
- [19] GitHub-repository van Espressif Systems: <https://github.com/orgs/espressif/repositories>
- [20] ESP-IoT-Solution: <https://github.com/espressif/esp-iot-solution>
- [21] Broncode voor dit project: <https://github.com/espressif/esp-box/tree/master/examples>



Listing 6: Afhandeling TTS-respons.

```
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;
}
```



ESP-Unlock

een tweefactor-authenticatie-dongle op basis van de ESP32-C3

Jakob Hasse, Espressif

Veel online-services bieden tegenwoordig tweefactor-authenticatie. De smartphone-authenticatie-app is waarschijnlijk de meest populaire keuze, maar een smartphone kan gestolen worden en vaak lopen privé en zakelijke dingen door elkaar. Dit is waar hardware-tokens in de vorm van een USB-stick van pas komen. In dit project wordt zo'n token gerealiseerd op basis van een goedkoop ESP32-C3-board.

Stel je voor dat je een cybercrimineel bent en de gebruikersnaam en het wachtwoord van je slachtoffer voor een willekeurige online-service te pakken hebt gekregen. Nu probeer je in te loggen. Het lukt! Maar dan vraagt de online-service je om een 'verificatiecode.' Waarom? Omdat die online-service tweefactor-authenticatie biedt en je slachtoffer die heeft ingeschakeld.

Tweefactor-authenticatie – veelgebruikte afkortingen zijn 2FA, TFA of MFA – betekent in feite dat je een extra stap nodig hebt om jezelf te authenticeren bij een online-service. Je authenticert jezelf niet alleen met iets dat je *weet*, maar ook met iets dat je *bezit*. In de praktijk heeft deze extra stap vaak betrekking op een tastbaar apparaat waar je over kunt beschikken. Een goed voorbeeld is geld opnemen bij een geldautomaat, waarbij je een pincode (kennis) en je pinpas (bezit) nodig hebt. Alleen het kennen van de pincode of het bezit van de kaart is niet genoeg om toegang te krijgen tot de bankrekening van het slachtoffer.

Maar niet alleen banken gebruiken tweefactor-authenticatie. Veel online-services bieden tegenwoordig ook tweefactor-authenticatie.

In tegenstelling tot banken geven ze geen bankpasjes uit. In plaats daarvan kun je een smartphone-authenticatie-app gebruiken of een standaard elektronisch apparaat, dat we vanaf nu 'hardware-token' zullen noemen. Je moet dan nog steeds je normale gebruikersnaam onthouden om jezelf te authenticeren (kennis), maar je moet ook je smartphone-authenticator-app of hardware-token gebruiken (bezit).

De smartphone-authenticatie-app is waarschijnlijk de meest populaire keuze, omdat bijna iedereen een smartphone heeft. Maar hoe zit het met een backup als je telefoon wordt gestolen? Hoe zit het met de scheiding tussen werk-accounts en je privételefoon? Wat als de accu van de telefoon leeg is? Of als het je gewoon niet uitkomt om je superdeluxe king-size smartphone uit je zak te vissen?

Dit is waar hardware-tokens van pas komen. Ze maken je niet alleen minder afhankelijk van een smartphone, maar ze kunnen ook klein en goedkoop worden vervaardigd. Een klein hardware-token kan eigenlijk altijd worden meegenomen. Een goedkoop hardware-token is ook voor mensen met een krap budget

een reëel alternatief, en het maakt het ook mogelijk om meerdere hardware-tokens te bezitten voor verschillende doeleinden (of als backup) zonder dat de eigenaar tot de bedelstaf wordt gebracht.

Hoe hardware-tokens werken

Je vraagt je misschien af: hoe kan de online-service jouw hardware-token onderscheiden van andere tokens? Het antwoord is dat zo'n hardware-token eigenlijk 'slim' is, omdat het een kleine microprocessor en wat geheugen bevat. Terwijl je het hardware-token instelt als authenticator voor je online-service, krijg je van de online-service een cryptografische sleutel die naar het geheugen wordt geschreven (waarover later meer). Die sleutel wordt vervolgens gebruikt om de authenticiteit van het hardware-token te bewijzen. Je kunt het zien als een wachtwoord dat door het hardware-token wordt onthouden om in te loggen bij de online-service. De manier waarop je dit soort hardware-tokens gebruikt, lijkt echter erg op het gebruik van een fysieke sleutel. Als je in plaats daarvan een smartphone-authenticator-app gebruikt, zal de sleutel ergens op je smartphone zijn opgeslagen, maar het bovengenoemde principe blijft gelijk. Eén standaard voor authenticatie op basis van twee factoren is het veelgebruikte Time-based One-time Password (TOTP) [1]. TOTP is gebaseerd op eenmalige wachtwoorden (one-time passwords, OTP), gemaakt van een cryptografische sleutel en de huidige tijd op de klok, vandaar het *time-based* in de naam. De sleutel wordt gedeeld tussen de online-service en het hardware-token. Een hashfunctie creëert een numerieke OTP van zes cijfers uit de sleutel en de huidige tijd. De online-service doet dezelfde berekening met zijn eigen kopie van de sleutel en de huidige tijd.

De OTP zoals door het hardware-token berekend, wordt naar de online-service gestuurd, die deze OTP vergelijkt met de OTP berekend uit zijn eigen kopie van de sleutel. Alleen als de twee OTP's overeenkomen, wordt de gebruiker succesvol geauthenticeerd. Tijdens de OTP-berekening wordt een hashfunctie gebruikt om ervoor te zorgen dat de originele sleutel niet gecompromitteerd wordt tijdens de overdracht van het hardware-token naar de online-service. Meer informatie over de TOTP-standaard is te vinden in [1].

Het ESP-Unclock hardware-token

Omdat hardware-tokens voor tweefactor-authenticatie zo handig zijn en de TOTP-standaard veel wordt gebruikt en eenvoudig te implementeren is, heb ik de 'ESP-Unclock' gemaakt: een open-source TOTP-compatibel tweefactor-authenticatie hardware-token. Ik koos de relatief goedkope ESP32-C3-chip voor het hardware-token omdat ik daarmee bekend ben en ook met het Espressif IoT-development framework (ESP-IDF) [2], waarmee een ESP32-C3 geprogrammeerd kan worden. Om precies te zijn koos ik een ESP32-C3-WROOM-02U module waarin de ESP32-C3 en andere nuttige componenten zijn gecombineerd.

De ESP32-C3 biedt een aantal erg handige functies:

- een geïntegreerde USB/serieel-omzetter die communicatie mogelijk maakt via USB, een interface die op vrijwel elke computer beschikbaar is;
- een CPU met cryptografische versnellers om de OTP te berekenen;
- hardware die flash-encryptie mogelijk maakt om de geheime sleutel te versleutelen en beveiligd opstarten om te voorkomen dat onbevoegde code op het hardwaretoken wordt uitgevoerd;
- de ESP32-C3-WROOM-02U module heeft flash-geheugen om programmeercode en sleutel in op te slaan.

Het ESP-IDF development framework implementeert flash-encryptie en secure boot bovenop de hardware als een gemakkelijk toegankelijke functie. De enige ontbrekende onderdelen waren een print voor alle hardware, en software om de stukjes en beetjes te koppelen. Het ESP-Unclock project combineert alle onderdelen, creëert een eenvoudig en goedkoop open-source TOTP

hardware-token en een authenticatie-toepassing voor een host-computer, die later wordt besproken.

We zullen nu de architectuur van het hele project behandelen, de hardware en de software, waarna enkele overwegingen volgen over beveiliging. Tenslotte bekijken hoe je de ESP-Unclock kunt gebruiken.

Projectarchitectuur

Om de OTP te berekenen en weer te geven, heb je het hardware-token zelf nodig (de ESP-Unclock dus) en een host-computer met een USB-A poort. De ESP-Unclock slaat de sleutels op om OTP's te genereren voor verschillende online-services. De host levert de huidige tijd via zijn real-time klok. Communicatie tussen de twee wordt gerealiseerd als seriële communicatie via USB, met behulp van de USB/serieel-periferie van de ESP32-C3. Het berekenen van een OTP gaat in vier stappen (zie **figuur 1**):

1. de hostcomputer haalt de huidige tijd op uit zijn eigen realtime-klok;
2. de hostcomputer stuurt een request om OTP-berekening naar de ESP-Unclock;
3. de ESP-Unclock gebruikt de juiste sleutel om de OTP te berekenen;
4. de ESP-Unclock stuurt de OTP terug in een resultaatbericht.

De communicatie (stappen 2 en 4 in **figuur 1**) tussen de host en ESP-Unclock bestaat uit een vrij eenvoudig vraag/antwoord-protocol waarbij de host altijd optreedt als initiator,

terwijl de ESP-Unclock alleen op verzoeken antwoordt. Alle berichten worden als platte tekst verzonden om het debuggen te vergemakkelijken. Een servicenaam in het request is noodzakelijk omdat het hardware-token met meerdere online-services werkt, waarvoor één sleutel per service nodig is. Het voorbeeldrequest in **figuur 1** is:

TOTP:github,1690975870

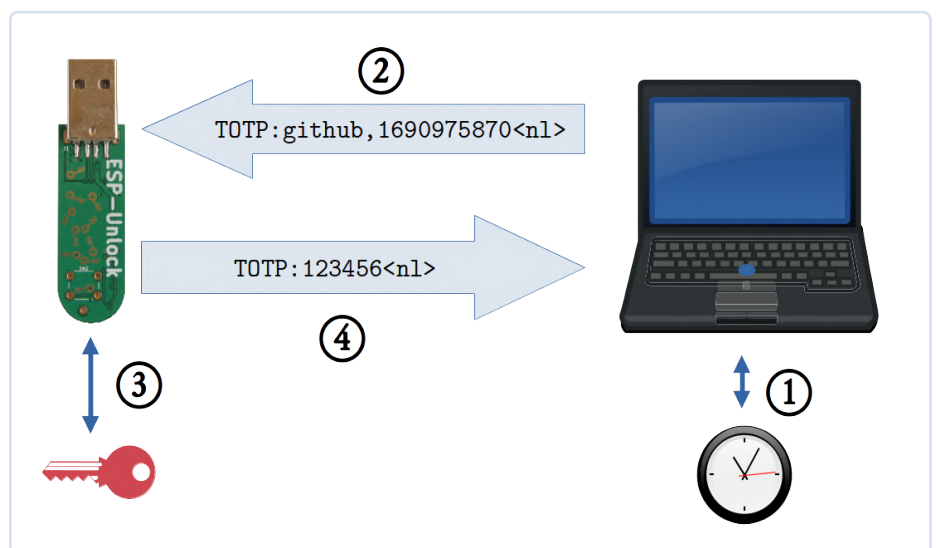
waarbij **TOTP:** en het newline-karakter aan het einde scheidingstekens zijn voor parsing, **github** de servicenaam van de te kiezen sleutel is en **1690975870** de huidige UNIX-tijd, geformatteerd als een decimaal getal. Het overeenkomstige antwoord in **figuur 1** is:

TOTP:123456

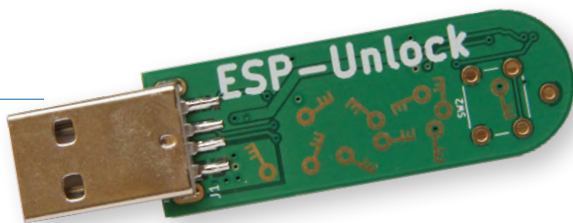
waarbij **TOTP:** en het newline-karakter weer scheidingstekens zijn en **123456** de OTP is. Bezoek voor meer informatie over dit protocol en de aanvullende berichten om servicenamen op te sommen en nieuwe sets van sleutel- en servicenamen toe te voegen de repository van dit project [3].

Hardware

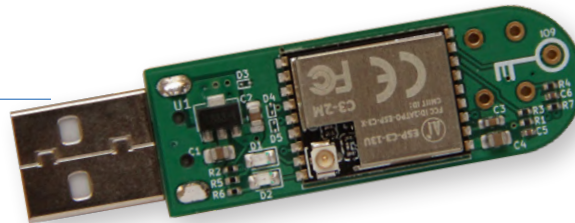
Omdat een klein formaat een eerste vereiste was voor de hardware, bevat de print voor de ESP-Unclock alleen de noodzakelijke componenten: een ESP32-C3-WROOM module (zonder antenne), de bijbehorende bootstrapschakelingen, voeding (maakt 3,3 V van de 5V-USB-voeding), twee LED's



Figuur1. Communicatie tussen een ESP-Unclock hardware-token en de host-computer.



Figuur 2. ESP-Unlock: voorzijde.



Figuur 3. ESP-Unlock: achterzijde.

en een USB-A connector voor aansluiting op een computer. Er is ook ruimte gereserveerd voor een optionele knop die door toekomstige softwareversies kan worden gebruikt om OTP-berekeningen alleen toe te staan als de knop wordt ingedrukt, waardoor de beveiliging wordt vergroot. Inclusief de USB-connector meet de hardware 6x1,6 cm, ongeveer de grootte van een normale deursleutel (**figuur 2** en **figuur 3**).

De hardware is ontworpen met KiCad, een open-source softwarepakket waarmee schakelingen kunnen worden gemaakt en print-layouts kunnen worden ontworpen. De print voor de ESP-Unlock is tweelaags (**figuur 4**), omdat dit doorgaans de goedkoopste optie is. Alle componenten behalve de through-hole exemplaren zitten aan één kant, waardoor alle SMD-componenten gemakkelijk met reflow-solderen gemonteerd kunnen worden.

Software

Twee softwaretoepassingen zijn relevant voor dit project: een firmware die op de ESP-Unlock draait, waardoor deze 'slim' wordt, en een toepassing die op de hostcomputer draait, de ESP-Unlock authenticatietoepassing.

De firmware die op de ESP-Unlock draait is een C++ applicatie die gebruik maakt van ESP-IDF [2] en ESP-IDF-C++ [4]. Zodra de ESP-Unlock firmware opstart, gaat de applicatie luisteren naar OTP-requests. De twee LED's knipperen eenmaal in een afwisselend patroon wanneer het apparaat klaar is voor requests. Zodra een OTP-request is ontvangen, controleert de ESP-Unlock firmware of het een sleutel heeft die overeenkomt met de naam in het verzoek. Zo ja, dan zoekt het de corresponderende sleutel op. Vervolgens berekent het de OTP met behulp van de zojuist gelezen sleutel en de tijd uit het OTP-request. Een resulterend antwoord met de OTP wordt teruggestuurd naar de host (zie figuur 1). Merk op dat de gehele OTP-berekening wordt uitgevoerd op de ESP-Unlock – de sleutel verlaat deze nooit.

De ESP-Unlock authenticator-applicatie is verantwoordelijk voor het aanbieden van een eenvoudige gebruikersinterface en voor het coördineren van de communicatie met de ESP-Unlock. De gebruikersinterface geeft

een lijst van alle diensten waarvoor sleutels beschikbaar zijn op de momenteel verbonden ESP-Unlock, toont OTP's voor corresponderende services en maakt het mogelijk om nieuwe sleutels toe te voegen. De applicatie is alleen geschreven voor Linux, maar de ESP-Unlock firmware trekt zich niets aan van het besturingssysteem van de host, dus de applicatie kan opnieuw worden geïmplementeerd voor of geport naar elk ander besturingssysteem.

Veiligheidsoverwegingen

Houd er rekening mee dat dit geen formele beveiligingsanalyse is, die te veel zou zijn voor het project in zijn huidige stadium. Het is bedoeld om te wijzen op gevaren die bestaan en deze gevaren begrijpelijk te maken voor de doorsnee-lezer.

De kritieke gegevens op de ESP-Unlock zijn de sleutels, die altijd geheim moeten blijven. Directe toegang tot de sleutels kan worden voorkomen door flash-encryptie te gebruiken. Maar de software die op de ESP-Unlock draait heeft ook toegang tot de sleutels. Daarom mag er alleen geautoriseerde software draaien om te voorkomen dat ongeautoriseerde software de sleutels naar buiten brengt. Secure Boot zorgt ervoor dat alleen geautoriseerde code op de ESP32-C3 draait. Als Secure Boot en flash-encryptie zijn ingeschakeld, kan zelfs een aanvaller met fysieke toegang tot het ESP-Unlock de sleutelgegevens niet lezen.

Een ander punt is dat de huidige software iedereen met fysieke toegang tot de ESP-Unlock in staat stelt om OTP's te genereren en uit te lezen. Als je hem verliest en iemand anders vindt hem, dan kan die persoon op dezelfde manier als jij OTP's genereren.

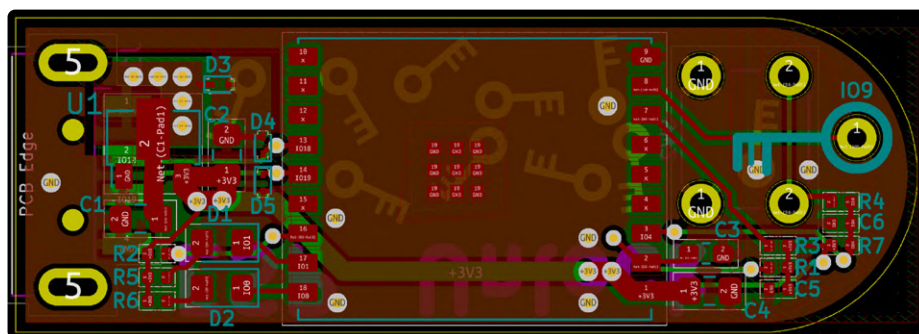
De firmware die op de ESP-Unlock draait kan

exploits bevatten. In het ergste geval zou zo'n exploit een aanvaller in staat stellen om zijn eigen code op de ESP-Unlock uit te voeren. Een verbetering die sommige klassen van exploits reduceert is het inschakelen van de stack protector in de firmware, die kan worden ingesteld bij de configuratie van de firmware. Een andere verbetering zou een op beveiliging gerichte code-review zijn, wat nog niet is gedaan omdat de code een vroeg prototype is. Merk op dat beveiligd opstarten geen bescherming biedt tegen code-exploits. Beveiligd opstarten beschermt alleen tegen het uitvoeren van ongeautoriseerde code op het hardware-token. Het beschermt niet tegen geautoriseerde code die zich misdraagt en de uitvoering van willekeurige code toestaat. Exploitatie van de code is het meest waarschijnlijk met fysieke toegang tot de ESP-Unlock. Aanvallen met software op de hostcomputer als proxy moeten ook overwogen worden, maar zijn minder waarschijnlijk vanwege de extra stappen.

Als een aanvaller fysiek toegang krijgt tot een ESP-Unlock, moet de eigenaar zo snel mogelijk een nieuw ESP-Unlock instellen en de geheime sleutels van alle gerelateerde online-services wijzigen. Bedenk echter dat zolang de aanvaller de gebruikersgegevens (gebruikersnaam en wachtwoord) niet bemachtigt, hij nog steeds niet kan inloggen. De OTP is immers slechts de tweede factor, en je zou nog steeds even veilig moeten zijn als zonder enige tweefactor-authenticatie.

Gebruik van de ESP-Unlock

Voordat de ESP-Unlock OTP's kan genereren, moet de ESP-Unlock authenticatie-applicatie geïnstalleerd worden en moet de sleutel voor de tweefactor-authenticatie worden



Figuur 4. Bovenste, onderste en zeefdruk-layer van de print.

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 XAM0 BWLR 57AC EFT0 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

Figuur 5. Base32-gecodeerde sleutel in GitLab (rood aangegeven).

esp-unlock

Device Found

ADD NEW

gitlab

r57ACEFT0X3MN6N

Add Entry

Figuur 6. Een nieuwe sleutel toevoegen.

esp-unlock

Device Found

github

782741

ADD NEW

Figuur 7. Een OTP aanvragen.

toegevoegd. Raadpleeg voor meer informatie over de installatie van de ESP-Unlock authenticatie-applicatie de instructies in de repository [5]. Om je sleutel toe te voegen, moet je deze eerst ophalen bij de compatibele online-service die je wilt gebruiken. Hoe dat in zijn werk gaat hangt af van de specifieke service, maar normaal gesproken leiden ze je door het proces van het instellen van je tweefactor-authenticatie-apparaat of applicatie. Ze tonen de sleutel meestal als een QR-code, wat handig is als je een smartphone-authenticatie app gebruikt, maar voor de ESP-Unlock is de tekstvorm nodig, geformatteerd als base32. GitLab bijvoorbeeld, geeft de base32 gecodeerde sleutel op de setup-pagina voor tweefactor-authenticatie naast de QR-code (figuur 5).

Zodra de ESP-Unlock op de hostcomputer is aangesloten, kan tweefactor-authenticatie voor een nieuwe service worden ingesteld in de authenticatie-toepassing door deze stappen te volgen (zie figuur 6):

1. klik op **ADD NEW**;
2. kies in het venster dat verschijnt een servicenaam en bevestig met **Enter**;
3. voer de sleutel in base32-formaat zelf in. Let erop dat je alle spaties weglaat, anders is de sleutel op de ESP-Unlock onjuist;
4. klik op **Add entry**.

Wanneer je vanaf nu een OTP nodig hebt, prik je de ESP-Unlock in de USB-poort van de host en start je de ESP-Unlock authenticatie-toepassing, die de namen van alle sleutels die op de ESP-Unlock zijn opgeslagen weergeeft en met een druk op de knop (stap 1 in figuur 7) de bijbehorende OTP opvraagt en weergeeft (stap 2 in figuur 7).

Als je dit project interessant vindt, voel je dan vrij om een kijkje te nemen of het te proberen en bezoek de ESP-Unlock repository [3] en de bijbehorende ESP-Unlock authenticator-applicatie repository [5]. Schema's en print-layout zijn te vinden in de hardware-repository [6], maar de ESP-Unlock hardware is niet noodzakelijk! Je kunt inderdaad elk development board uit de ESP32-serie gebruiken, mits de configuratie waar nodig wordt aangepast (raadpleeg in dat geval ook de README.md van de ESP-Unlock repository [3]). Alle bijdragen aan het project, zoals suggesties, hints, verbeteringen en bugrapporten, zijn van harte welkom!  230559-03

Vragen of opmerkingen?

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



Over de auteur

Jakob Hasse heeft een graad in computwetenschappen en heeft verschillende interesses. Tijdens zijn studie werkte hij bij het Duitse lucht- en ruimtevaartcentrum op het gebied van robotica. Later ontwikkelde hij een passie voor embedded systemen en open-source software. Nadat hij begonnen was als embedded Linux-systems engineer, kwam Jakob bij Espressif om te werken aan hun FreeRTOS-gebaseerde Espressif IoT Development Framework. Daarnaast is hij geïnteresseerd in IT-beveiliging, wat de belangrijkste drijfveer is achter het ESP-Unlock project.

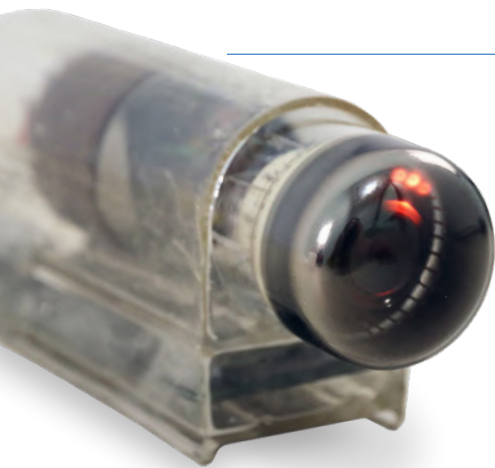


Gerelateerde producten

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

WEBLINKS

- [1] Time-based One Time Password (TOTP) standaard: <https://ietf.org/rfc/rfc6238.txt>
- [2] Espressif IoT Development Framework (ESP-IDF): <https://github.com/espressif/esp-idf>
- [3] De repository van dit project: <https://github.com/0xjakob/esp-unlock>
- [4] ESP-IDF-C++: <https://github.com/espressif/esp-idf-cxx>
- [5] De ESP-Unlock authenticatie-app: <https://github.com/0xjakob/esp-unlock-host-gui>
- [6] Hardware-repository: <https://github.com/0xjakob/esp-unlock-hardware>



Dekatron

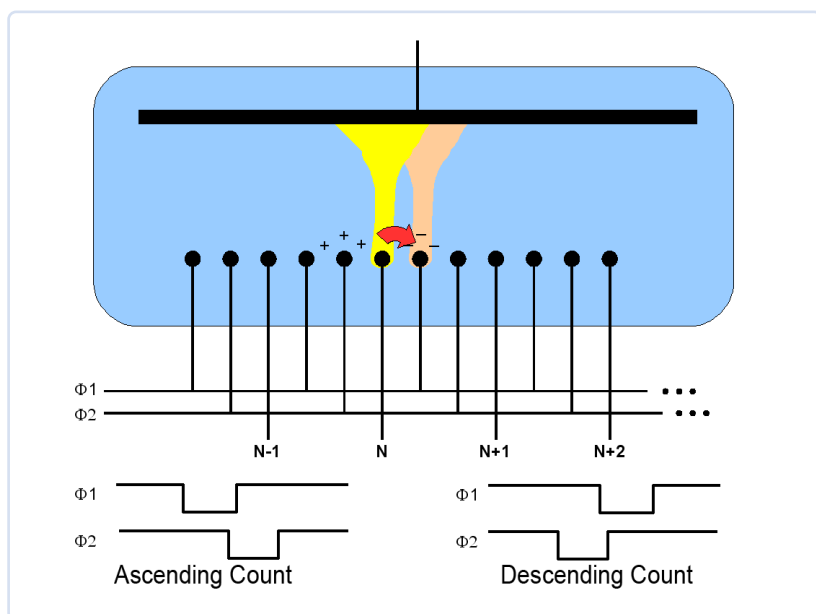
een stukje geschiedenis komt tot leven!

Jeroen Domburg, Espressif

In de muziek gaan oude en moderne stijlen vaak samen om iets bijzonders op te leveren; in de elektronica is het niet anders. In dit artikel stuurt een uiterst moderne ESP32-C3-module effectief een Dekatron aan, een tienteller-buis uit de jaren '50 – waardoor deze na 70 jaar weer tot leven komt!

In tegenstelling tot sommige eerbiedwaardige instituten zoals Elektor, heeft Espressif geen geschiedenis die erg ver teruggaat. En dat is misschien maar goed ook: we betwijfelen of er in 1961 een grote markt zou zijn geweest voor IoT WiFi-chips. Nee, Espressif werd in 2008 opgericht, en bracht ruim vijf jaar later zijn eerste chips uit: de ESP8089 en de ESP8266. Terwijl de ESP8089 vooral een OEM-chip was, ontworpen om bijvoorbeeld Android-tablets te voorzien van WiFi-functionaliteit, is de ESP8266 een echte IoT WiFi-chip. Het feit dat er zelfs nu nog nieuwe projecten worden gestart met de ESP8266 illustreert dat deze nog lang niet als 'retro' kan worden

Figuur 1. Basisprincipe van een Dekatron (bron: Wikimedia Commons).



beschouwd. (Hoewel – als je zelf overweegt om een project te starten met een ESP8266-chip, willen we je vragen om in plaats daarvan bijvoorbeeld een ESP32-C3 te overwegen; het blijkt dat een half decennium aan innovatie een veel mooiere chip voor ongeveer dezelfde prijs heeft opgeleverd).

In dit licht bezien leek het me interessant om iets ouds met iets nieuws te versmelten. In dit specifieke geval ga ik een meer dan 50 jaar oude Dekatron koppelen met de tamelijk nieuwe ESP32-C3 en die combinatie gebruiken als indicator voor de internetbandbreedte die we momenteel in beslag nemen. Zoiets had gegarandeerd niemand in gedachten toen de Dekatron werd ontwikkeld, maar het is desondanks een nuttige toepassing.

De Dekatron-buis

Voor het geval je niet bekend bent met antieke elektronica: een Dekatron is een met gas gevulde glazen buis (in tegenstelling is tot een echte vacuümbuis, waar normaal gesproken geen gas in te vinden is). Het gas in kwestie is meestal een inert gas zoals neon, maar ook gassen zoals waterstof werden gebruikt. Als de naam 'neon' je doet denken aan de kleine neonlampjes die je vroeger zag in aan/uit-knoppen van oude apparaten, dan zit je er niet ver naast: die werken volgens een vergelijkbaar principe. Een Dekatron is in feite een teller met een visuele indicatie van de tellerstand. Hij werkt als volgt: de bovenkant van een Dekatron bevat een ronde anode. Hij heeft 30 kathodes in de vorm van pinnen die gelijkmatig rondom die anode zijn geplaatst, zoals geschetst in **figuur 1** en te zien in de kleine foto's van **figuur 2**. Tien van deze kathodes zijn zogenaamde uitgangskathodes, de andere twintig zijn geleidekathodes, die weer zijn onderverdeeld in twee groepen: G1 en G2 (respectievelijk Φ1 en Φ2 in **figuur 1**). Elke uitgangskathode heeft een G1-geleidekathode aan de ene kant en een G2 aan de andere kant. Alle G1-kathodes zijn parallel geschakeld, net als alle G2-kathodes. Bij het starten wordt een spanning van 400 V (stroombegrensd) aangelegd tussen de anode en de uitgangskathodes. Aangezien dit meer is dan de ionisatiespanning van het gas, zal het gas ioniseren en gloeien bij één van deze kathodes. Aangezien hierdoor de anodespanning tot onder de ionisatiespanning wordt verlaagd, zal geen enkele andere elektrode oplichten. Deze gloed kan worden 'overgebracht' naar de aangrenzende uitgangskathodes door G1 en G2 in de juiste volgorde aan massa



te leggen – als bijvoorbeeld G1 aan massa wordt gelegd en vervolgens G2, dan zal de ionisatie zich met de klok mee verplaatsen, terwijl bij omgekeerde G1/G2-volgorde de ionisatie tegen de klok wordt verplaatst.

Wat is het nut van al dat geïoniseer? Nou, het maakt het mogelijk dat deze combinaties uit glas, metaal en gas de G1/G2-pulsen tellen; om precies te zijn kan een Dekatron tot tien tellen voordat hij weer bij nul begint (vandaar de 'deka' in 'Dekatron'). Doorgaans worden sommige of alle uitgangskathodes naar afzonderlijke pinnen aan de voet van de Dekatron naar buiten gevoerd, zodat bepaalde tellerstanden kunnen worden gedetecteerd. Op deze manier kunnen ze worden gebruikt als pulstellers en frequentiedelers, maar ook als geheugenelementen: een beroemde computer uit 1951 gebruikt Dekatrons op die manier [1]. (Let op de tegenwoordige tijd in de vorige zin: die computer functioneert nog steeds – als museumstuk!)

Het is duidelijk dat Dekatrons inmiddels neerdere malen obsoleet gemaakt zijn, aanvankelijk door tellers met van discrete transistoren. Toen IC's de ingewikkelder taken overnamen, werden ze vervangen door chips zoals de eerbiedwaardige CD4017-tienteller. En nu, in de tijd van goedkope microcontrollers en FPGA's, wordt de functie van de Dekatron meestal vervuld door een paar regels code of HDL.

Een nadeel van al deze nieuwerwetse teltechnieken is echter dat ze, in tegenstelling tot de Dekatron, zelf geen visuele output leveren. Natuurlijk kun je LED's aansluiten op je CD4017 (zoals veel lezers in het verleden hebben gedaan), of een draai-indicatie maken met dat LC-display dat je hebt aangesloten op je nieuwe microcontroller, maar die neongloed van de Dekatron heeft iets bijzonders dat niet echt kan worden vervangen door pixels of oplichtende halfgeleiders.

Voedingsuitdagingen

En precies daar komen we ook een probleem tegen: de ESP32-C3 is bij uitstek geschikt om die pixels of LED's aan te sturen, maar is minder geschikt als interface met 70 jaar oud glaswerk: zo'n Dekatron mag nog zo goed zijn, maar hij merkt de 3,3 V die de ESP32-C3 naar zijn IO-pinnen stuurt niet eens op. Met andere woorden, we moeten niet alleen die 400 V genereren, maar ook diverse hoge(re) spanningen kunnen schakelen.

Laten we beginnen met die 400 V. De eenvoudigste manier om die te genereren is door een spanningsvervubbelaar op de netspanning aan te sluiten. Ik wilde dit project echter liever niet op het net aansluiten; de daarvoor vereiste veiligheidsmaatregelen zouden een beetje te veel van het goede zijn. In plaats daarvan besloot ik dat een moderne USB-C connector de juiste keuze was; tussen laptop-powerbanken, smartphone-adapters en die nondescripte lader die je bij een elektronisch apparaat hebt gekregen (maar waarvan je je met geen mogelijkheid kunt herinneren waar die voor was) zijn er genoeg voedingen met zo'n connector te vinden.

Bovendien is USB-PD tegenwoordig een veelgebruikte standaard, waardoor je hogere spanningen kunt krijgen dan de 5 V die normaal op een USB-poort beschikbaar is. Dat bleek hier goed van pas te komen.

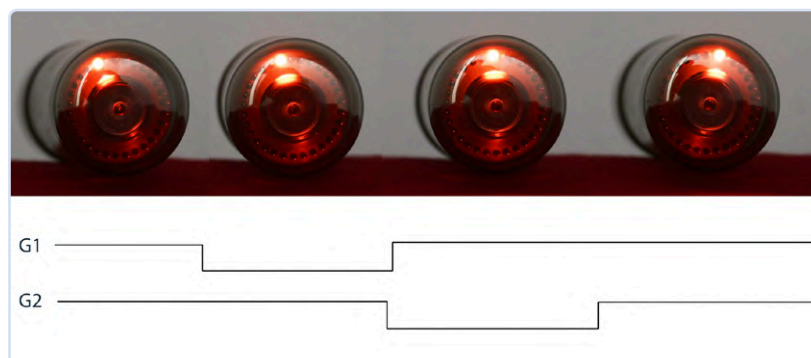
Dus hoe maak je die 400 V? Er zijn meerdere manieren om dit te doen. De simpelste oplossing gebruikt transformatoren: je kunt bijvoorbeeld je laagspanning converteren naar een wisselspanning, die door de laagspanningswikkeling van een transformator jagen en hoogspanning van het andere wikkeling afnemen. Een andere manier is om die trafo te gebruiken als een flyback-converter: laad het magnetische veld op door een spanning aan te leggen op de laagspanningszijde, schakel die spanning dan af en laat het afnemende magnetische veld oppikken door de hoogspanningszijde, en daar heb je je hoogspanning. Het probleem met beide oplossingen is dat miniatuurtransformatoren in veel gevallen een nicheproduct zijn, en dat brengt het risico met zich mee dat het ontwerp niet meer nagebouwd kan worden als de gekozen transformator niet meer leverbaar is. Hoewel ik niet verwacht dat dit ontwerp in zeer grote aantallen zal worden nagebouwd, wil ik wel dat het lang werkt en te repareren is, dus ik wil eigenlijk geen niet-standaard componenten gebruiken.

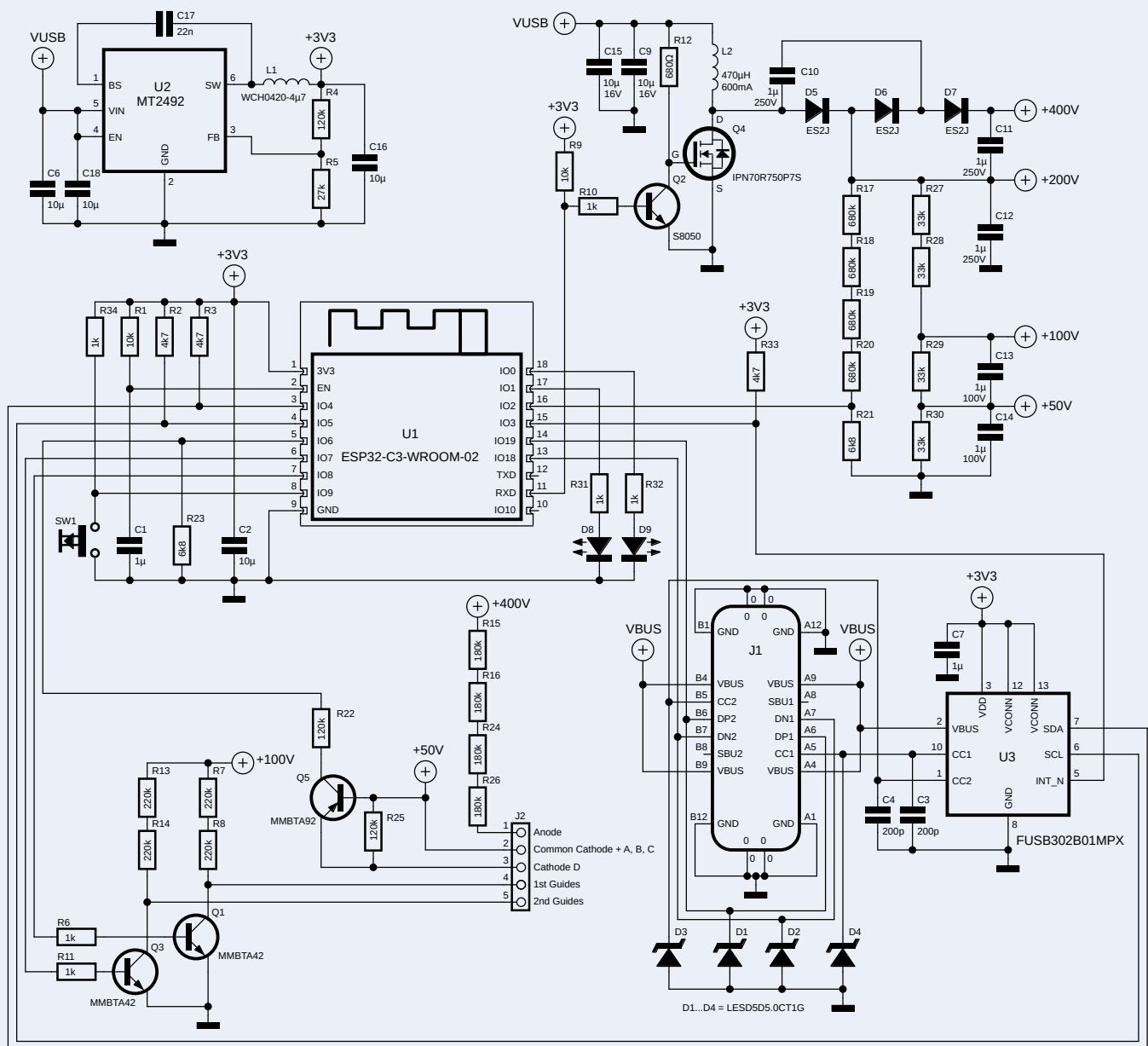
De oplossing

In plaats daarvan heb ik gekozen voor een andere mogelijkheid: een boost-converter. Een boost-converter wordt meestal gebruikt om spanningen om te zetten naar iets hogere spanningen. Je kunt er bijvoorbeeld een gebruiken als je 5V-logica wilt laten werken met twee AA-batterijen. Ze hebben het nadeel dat het genereren van (zeer) hoge spanningen problematisch is – de halfgeleider die de lage spanning schakelt moet ook bestand zijn tegen de hoge uitgangsspanning, en de maximale factor waarmee je de ingangsspanning kunt opvoeren wordt beperkt door zaken als de DC-weerstand van de spoel die je gebruikt.

Een truc om deze problemen te omzeilen is het gebruik van een spanningsvervubbelaar na de boost-converter; op die manier hoeven we maar tot de helft van de benodigde 400 V te boosten. (Eerlijkheidshalve moet ik opmerken dat deze truc afkomstig is van een beproefde Dekatron-boost-converter [2].) De andere truc is dat we al met een hogere spanning kunnen beginnen: over het

Figuur 2. De Dekatron in actie (optellend), met de bijbehorende G1- en G2-signalen.





Figuur 3. Het schema van dit project.

algemeen kunnen PD-compatibele USB-C-laders niet alleen 5 V leveren, maar desgewenst ook 9 V, 15 V of 20 V. Dat betekent dat de boost-converter niet zo hard hoeft te werken om de ingangsspanning om te voeren. (Een andere mogelijkheid zou zijn om een tweede boost-converter te gebruiken om de voedingsspanning eerst een beetje te verhogen voordat deze helemaal naar 400 V wordt opgefokt. Maar eigenlijk wilde ik altijd al met USB-PD spelen, dus daarom heb ik de USB-PD-route gekozen).

De details van de implementatie zijn te vinden in **figuur 3**. De spanning van de USB-poort J1 wordt eerst gestabiliseerd met twee keramische condensatoren. De belangrijkste boost-logica wordt gevormd door L2 en Q4: als Q4 in geleiding is, bouwt hij een magnetisch veld op, en als hij open is, wordt de stroom van het verdwijnende veld gedumpt in C11 en C12 via de gelijkrichter annex spanningsverdubbelaar gevormd door D5...D7. Dit resul-

teert in twee gelijkspanningen, één van 200 V en één van 400 V.

Q4 is speciaal; zoals eerder gezegd moet deze specifiek bestand zijn tegen de hoge spanningen die L2 genereert. Helaas zijn MOSFET's die dit aankunnen meestal geen logic-level MOSFETS, dus is Q2 toegevoegd om het inkomende 3,3V-PWM-sigitaal om te zetten in iets dat Q4 laat geleiden. Dat PWM-sigitaal wordt gegenereerd door de ESP32-C3 en de duty cycle wordt geregeld door de 200V-spanning te meten en dienovereenkomstig aan te passen. Hiertoe dient spanningsdelers R17...R21, die de 200 V reduceert tot iets dat de ESP32-C3 aan kan.

Naast 400 V hebben we ook een aantal lagere spanningen nodig, en deze worden gegenereerd door de 200 V te delen met behulp van een aantal weerstanden in serie getekend zijn hoewel er in theorie één zou kunnen volstaan: dat is omdat ik 0603-componenten

wilde gebruiken, en deze zijn slechts bestand tegen 75 V of zo. De serieschakeling betekent een geringere spanningsval over de afzonderlijke weerstanden, zodat er geen vonk-overslag kan optreden: hoewel hoogspannings-boogontladingen een spectaculaire indicatie van een supersnelle download zouden zijn, is dat slechts éénmalig en gaat de kamer zo stinken...

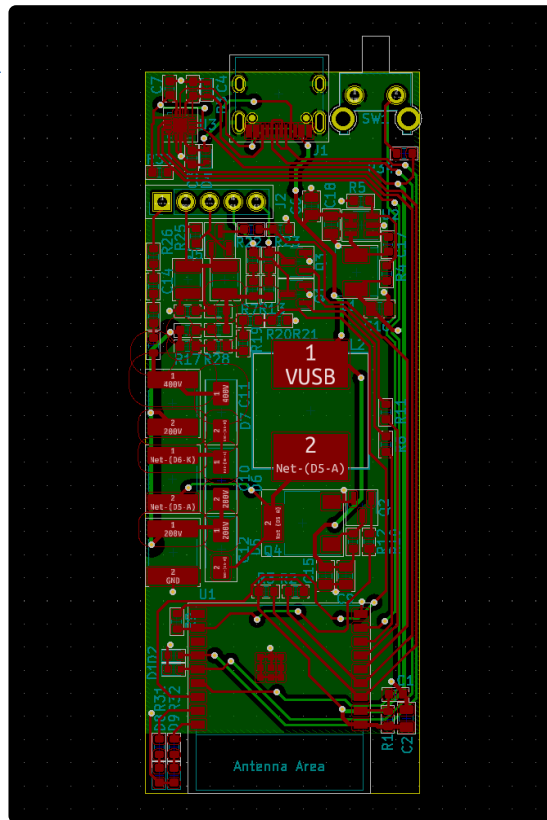
Aan de Dekatron-zijde, zoals geïllustreerd in het schema in **figuur 3**, wordt de 400 V naar de anode geleid via een 720kΩ-weerstand. Volgens de datasheet [3] moet de anodestroom beperkt worden tot ongeveer 310 μA, en deze reeks weerstanden zorgt daarvoor. De G1- en G2-signalen worden door twee NPN-transistors op de vereiste spanning gebracht, en een van de niet-gemeenschappelijke kathodes wordt naar een level shifter met een PNP-transistor gebracht om de ESP32-C3 te laten weten dat de 'gloed' deze kathode passeert.

De ESP32-C3-module

De ESP32-C3 zelf is zichtbaar in het hart van het schema van figuur 3. Aangezien ik een ESP32-C3-WROOM02-module gebruik, is de meeste ondersteuningshardware al aan boord: flash, RF-aanpasnetwerken en de antenne zijn allemaal aanwezig. De enige dingen aan de buitenkant zijn twee indicatie-LED's, een drukknop en een RC-netwerk om een power-on reset te genereren. Als je KiCad gebruikt voor je print-layout (zoals ik), moet je weten dat Espressif een gratis repository van symbolen en footprints voor alle modules en chips onderhoudt [4]. Tot slot wordt rechtsboven in het schema het voedingsgedeelte getoond. De USB-C connector heeft twee doelen: als er een USB-PD voeding is aangesloten, voedt deze alles, maar je kunt hem ook aansluiten op je computer. In dat geval zal de Dekatron niet opstarten, omdat de 5V-voeding daarvan niet genoeg is, maar het is dan wel mogelijk om de code in de ESP32-C3 flash te herprogrammeren. Om de USB-PD-verkeer te regelen, wordt een FUSB302 USB-C controller gebruikt; de ESP32-C3 kan hiermee via I²C praten om hem te laten communiceren met de voeding. Om de binnenkomende spanning om te zetten naar iets waar de ESP32-C3 op kan draaien, heb ik een synchrone buck-converter gebruikt, namelijk de Arosemi MT2492. Deze kleine chip werkt tot 16 V en het feit dat het een schakelende voeding is, betekent dat hij lekker koel blijft terwijl hij de 3,3V-spanning genereert waar de ESP32-C3 op draait.

Print-layout

Figuur 4 toont het ontwerp van de print. Ik heb gekozen voor een print die even breed is als de Dekatron zelf, zodat hij eronder kan 'schuilen'. Het routeren van deze print is eigenlijk iets lastiger dan het lijkt: vanwege de hoge spanningen moest ik rekening houden met kruipafstanden, anders zou er overslag kunnen optreden, en de hoge spanning maakt sowieso fysiek grotere componenten noodzakelijk. Toch kreeg ik het voor elkaar om alles onder te brengen op een tweelaags-print, zelfs met een



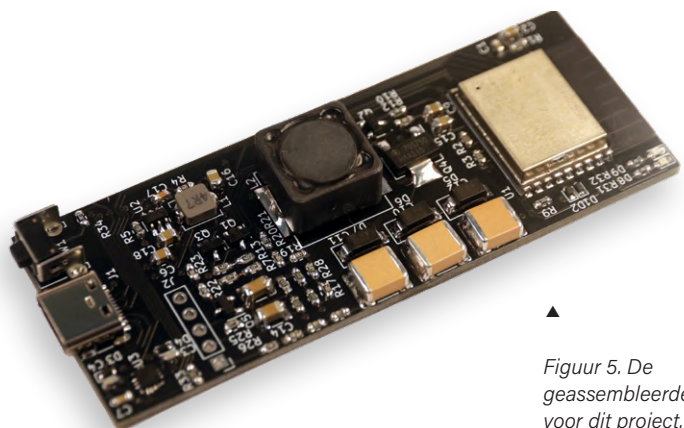
Figuur 4. Print-layout van de ESP32-C3-gebaseerde Dekatron-interface.



massavlak zonder veel onderbrekingen. Ik besloot om de print te voorzien van een mooi zwart soldeermasker zodat hij niet zou opvallen en de show zou stelen van de Dekatron, zoals je kunt zien in **figuur 5**.

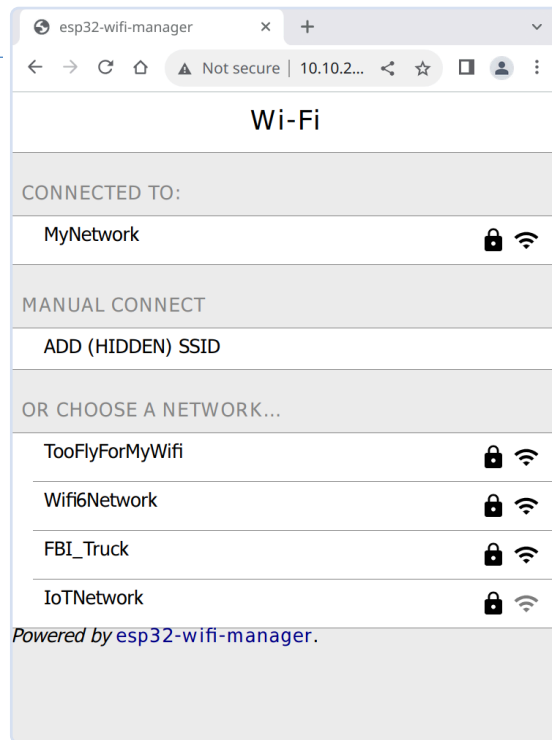
Software-implementatie

Toen de hardware gebouwd was, begon ik met coderen. De software moet een paar dingen doen. Allereerst moet hij ervoor zorgen dat de rest van de hardware de juiste spanning krijgt, en om dat te doen moet het een USB-PD stack implementeren om met de aangesloten voeding te communiceren. Ik heb hiervoor een bestaande bibliotheek gebruikt die goed werkt [5]. De hardware zelf kan met een vrij breed spanningsbereik overweg; de software probeert daarom 12 V van de voeding te krijgen, maar als dat niet lukt is ook 9 V of 15 V acceptabel. Om de Dekatron ook zijn hoge spanning te laten krijgen, wordt de momentane spanning op de 200V-rail gemeten en past de software de PWM-duty-cycle naar de boost-converter indien nodig aan.

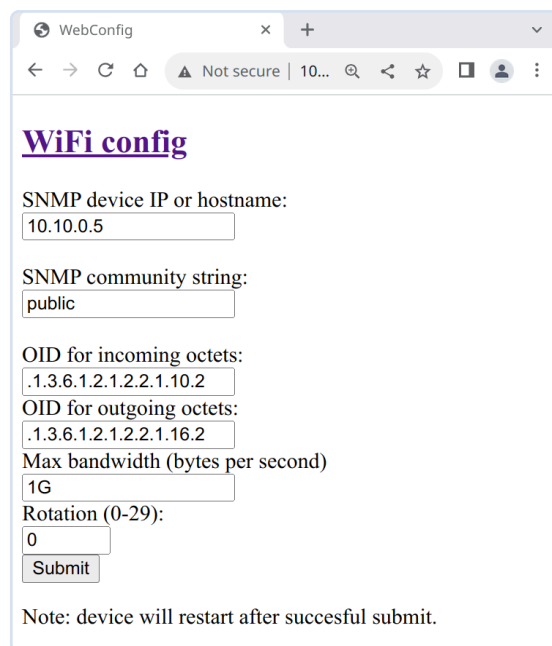


Figuur 5. De geassembleerde print voor dit project.

Figuur 6. Het juiste toegangspunt selecteren op de ESP32 WiFi Manager-pagina.



Figuur 7. Voer hier de configuratieparameters voor de verbinding in.



Figuur 8. 3D-geprinte behuizingen voor het project: onbewerkt (links) en afgewerkt met een glanzende en transparante laklaag (rechts).



Ten tweede moet de software uitvinden wat de huidige gebruikte internetbandbreedte is. Mijn internetverbinding wordt eigenlijk gedistribueerd via een Ethernet-switch, en deze houdt statistieken bij over alle gegevens die er doorheen stromen. Deze gegevens zijn eenvoudig te verkrijgen via een protocol dat SNMP (Simple Network Management Protocol) heet. Dit protocol wordt over het algemeen gebruikt om IT-infrastructuur zoals routers, switches, servers enzovoort te monitoren. Het heeft weinig overhead omdat er maar een paar UDP-pakketten nodig zijn om de statistieken voor een netwerkpoort op te vragen, dus ik zou dit gemakkelijk meerdere keren per seconde kunnen doen om een real-time idee te krijgen van het actuele dataverkeer. Deze gegevens worden vervolgens gebruikt om de snelheid van de G1- en G2-pulsen in te stellen, wat betekent dat een snellere download resulteert in snellere rotatie met de klok mee. Als iemand in plaats daarvan data uploadt, zal het Dekatron-diaplay ook draaien, maar dan tegen de klok in.

De laatste functie die de firmware moet hebben is een soort configuratie-interface. Ik hou niet van hard-gecodeerde configuraties in mijn firmware, dus ik had een manier nodig om de WiFi-referenties in te voeren, evenals de IP- en SNMP-informatie van de switch. Het invoeren van WiFi-informatie wordt *provisioning* genoemd, en ESP-IDF heeft daar een prima oplossing voor die een smartphone-app gebruikt om het proces te stroomlijnen. In dit geval koos ik echter voor een zogenaamde *WiFi manager*. Wanneer deze niet geconfigureerd is, creëert hij een WiFi-toegangspunt waar je een laptop of telefoon op kunt inrichten. Eenmaal verbonden wordt een webpagina geopend waarmee je het WiFi-toegangspunt kunt selecteren en het wachtwoord kunt invoeren (figuur 6). Omdat de WiFi-manager toch al een webserver nodig heeft, is het eenvoudig om deze uit te breiden en ook een configuratiepagina voor de SNMP-parameters toe te voegen (figuur 7). Om het debuggen te vergemakkelijken, zijn er ook enkele statistieken beschikbaar, zoals de feitelijke spanning die de voeding levert.

Ontwerp van de behuizing

Toen de software en hardware klaar waren, had ik nog één ding nodig: een behuizing. Voor dit project was een behuizing niet alleen om decoratieve redenen nodig: de print voert spanningen die, hoewel ze waarschijnlijk niet dodelijk zijn, toch behoorlijk schokkend kunnen zijn. Ik heb dit werk uitbesteed: ik vroeg een van onze industriële vormgevers of hij een behuizing voor me kon maken en deze kon printen op onze SLA 3D-printer. Hij was zo aardig om me van dienst te zijn (dankjewel Kaijie!) en printte een behuizing met een transparante hars, zodat je nog steeds de ingewanden van de Dekatron kunt zien. Tenminste, dat was het idee. Het blijkt dat de manier waarop de behuizing is geprint, deze ondoorzichtig maakt in plaats van transparant. Gelukkig kan dit op een vrij eenvoudige manier worden opgelost: bedek het geprinte object met een dun laagje hars en hard dit vervolgens



uit met een UV-lamp. Het resultaat is te zien **figuur 8**, naast een onbehandeld behuizing. Waarschijnlijk had ik de afdeklaag nog gelijkmatiger kunnen maken, maar eerlijk gezegd vind ik de 'handgemaakte' esthetiek van de behuizing niet erg: wanneer de Dekatron gemonteerd is, zoals in **figuur 9**, trekt deze automatisch de aandacht naar de voorzijde. En dat ziet er natuurlijk nog beter uit als het apparaat is ingeschakeld: zelfs als je geen idee hebt van wat er eigenlijk gebeurt, is de gloeiende punt die rond en rond gaat echt een blikvanger; **figuur 10** geeft een impressie.



Figuur 9. Het geassembleerde project in zijn uiteindelijke behuizing.



Een laatste opmerking

Al met al heb ik iets gemaakt waarvan ik hoop dat het, in ieder geval gedeeltelijk, als 'retrotronisch' kan worden beschouwd. Misschien zal het hele project zich ooit als zodanig kwalificeren. En misschien werken Elektor en Espressif tegen die tijd weer samen aan een tijdschrift, en misschien – héél misschien – mag ik dit apparaat dan uit de kast halen en lyrisch vertellen over alles wat tot zijn geboorte heeft geleid. Maar voorlopig is kijk ik er met plezier naar als ik wacht tot een grote download klaar is. Als je een Dekatron hebt en dit ontwerp wilt nabouwen (of stukjes hard- of software wilt hergebruiken voor je eigen projecten): dit hele project is open-source. Je kunt de firmware, de print-layout en de 3D-ontwerpbestanden voor de behuizing vinden op GitHub [6].

230560-03

Vragen of opmerkingen?

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

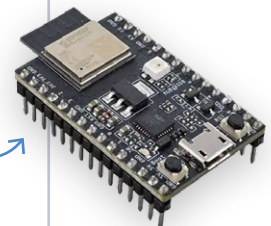
Over de auteur

Jeroen Domburg is Senior Software en Technical Marketing Manager bij Espressif Systems. Met meer dan 20 jaar embedded ervaring is hij betrokken bij zowel het software- als het hardware-ontwerpproces van de SoC's van Espressif. In zijn vrije tijd knutselt hij ook graag met elektronica om dingen te maken die praktisch bruikbaar zijn, maar ook minder nuttige zaken.



Gerelateerde producten

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



Figuur 10. Het prototype aan het werk: een onweerstaanbare blikvanger!



WEBLINKS

- [1] De Harwell Dekatron-computer (WITCH): <https://tnmoc.org/witch>
- [2] Dekatron boost-converter: <https://threeneurons.wordpress.com/dekatron-stuff/dekatron-do-hickie-kit>
- [3] GC10/4B datasheet: <http://r-type.org/pdfs/gc10-4b.pdf>
- [4] KiCad-bibliotheken voor Espressif-chips en -modules: <https://github.com/espressif/kicad-libraries>
- [5] USB-PD driver-stack voor de FUSB302: <https://github.com/Ralim/usb-pd>
- [6] Repository voor dit project: https://github.com/Spritem/elek_dekatron

De Smart Home-revolutie

Paulus Schoutsen over Home Assistant, ESPHome en meer

Vragen door Saad Imtiaz (Elektor)

Paulus Schoutsen, het brein achter Home Assistant en ESPHome, vertelt over zijn reis naar de wereld van domotica, IoT en meer. Ontdek hoe gebruikersvriendelijke firmware van ESPHome heeft gezorgd voor een doorbraak in slimme apparaten voor zelfbouw.



Paulus Schoutsen (bron: Midjourney en Photoshop).

Elektor: Vertel eerst eens iets over jezelf. Ben je altijd geïnteresseerd geweest in elektronica en techniek? Wat deed je kiezen voor een studie computerwetenschappen aan de TU Twente?

Schoutsen: Ik heb Business Information Technology gestudeerd. Dat betrof ook programmeren en databases, maar vooral informatiesystemen: welke informatiestromen doorlopen bedrijfsprocessen en hoe kun je die vastleggen, verwerken en analyseren? In zekere zin is daar de kiem gelegd van mijn ontwikkeling in de richting van domotica. Ik heb altijd belangstelling gehad voor programmeren, maar niet zoveel voor elektronica. Dat is er toevallig bij gekomen. Toen in 2012 de Philips Hue-lampen op de markt kwamen, heb ik er meteen een gekocht. Ik begreep dat ze een lokale API hadden en schreef een script om ze te besturen vanaf mijn computer. Dat script groeide, en op

17 september 2013 publiceerden we de eerste versie op GitHub. Dat was de geboorte van Home Assistant. Dit open source-domoticaplatform gericht op lokale besturing en privacy is het op één na actiefste project op GitHub [1] geworden.

Home Assistant groeide en ik begon met elektronica te spelen, maar er was een groot probleem: elektronica met het internet verbinden was te kostbaar. Een Arduino WiFi-shield kost \$ 70. Maar toen kwam de chip die alles zou veranderen: de ESP8266 van Espressif.

De ESP8266 was een chip van \$ 3,50, die Arduino-compatibele code kon draaien en verbinding kon maken met een draadloos netwerk. Deze betaalbare chip maakte het eindelijk financieel mogelijk om elektronica te bouwen die in een slimme woning kon worden geïntegreerd.

De ESP8266 kreeg langzaam voet aan de grond in de hobbywereld, en ik richtte Nabu Casa op. Nabu Casa garandeert continuïteit in de ontwikkeling van Home Assistant. Home Assistant is een groot open source-project, en voor zo'n project is veel administratie, structuur en onderhoud nodig – meer dan iemand in zijn vrije tijd kan doen. Nabu Casa-medewerkers in

Over Paulus Schoutsen

Paulus Schoutsen is de oprichter van Home Assistant, één van de grootste open source-projecten op GitHub, en van Nabu Casa, het bedrijf dat de continuïteit van Home Assistant garandeert. Zijn werk draait om het bouwen van 'Open Home' – zijn visie op een slimme woning met privacy, keuze en duurzaamheid.

vaste dienst maken dat mogelijk; het wordt betaald uit abonnementen op extra diensten die we de gebruikers bieden. Nabu Casa heeft geen investeerders en bestaat alleen voor de gebruikers.

Naarmate Home Assistant groeide hebben wij onze Open Home-visie gedefinieerd [2]. Dat is een visie voor slimme woningen gebaseerd op drie waarden: privacy, keuze en duurzaamheid. Vanuit die visie is het logisch dat een slimme woning lokaal en privé moet zijn. Dat geldt zowel voor het centrale platform (Home Assistant zelf) als voor de aangesloten apparaten.

Een van de belangrijkste ontwikkelaars van Home Assistant was Otto Winter. Hij was geïnteresseerd in elektronica maar vond dat de eerste ESP8266-projecten teveel overhead met zich mee brachten. Er moest een gemakkelijker manier komen voor integratie met Home Assistant. Hij ging op zoek naar een oplossing en op 21 januari 2018 publiceerde hij EspHome [3]. ESPHome werd steeds populairder, maar Otto had geen tijd meer om het te beheren. Omdat ESPHome essentieel is voor mensen die apparaten willen ontwikkelen die bij onze Open Home-waarden passen, wilden we het ondersteunen. Dus op 21 maart 2018 nam Nabu Casa ESPHome over [4]. We hebben nu twee voltijds ontwikkelaars die werken aan ESPHome, betaald door de abonnees van de Home Assistant Cloud van Nabu Casa.

Elektor: Kun je kort uitleggen wat ESPHome is? Hoe verschilt het van andere IoT-firmware en domotica-oplossingen?

Schoutsen: ESPHome is firmware voor ESP8266-, ESP32- en Raspberry Pi Pico-boards waarmee gebruikers gemakkelijk slimme domotica-apparaten kunnen bouwen. ESPHome neemt alle saaie overhead van de software voor zijn rekening, zodat de gebruiker zich kan concentreren op zijn eigen hardware.

We kunnen dat toelichten aan de hand van een voorbeeld: sluit een simpele temperatuursensor aan op een ESP32, vertel ESPHome in een configuratiebestand met welke pin de sensor is verbonden en klik op *Install*. ESPHome genereert dan alle benodigde firmware en installeert die op je ESP32. Die start dan op, maakt verbinding met het WiFi-netwerk, en de temperatuursensor is meteen beschikbaar in Home Assistant. Dat is alles.

De eerste keer dat je ESPHome installeert, moet het board met je computer worden verbonden. Maar daarna gaan alle updates draadloos. Een tweede sensor op hetzelfde board? Pas het configuratiebestand aan en druk op *Install*. Het maakt niet uit waar in huis het apparaat zich bevindt, de updates werken altijd. Een bijkomend voordeel van ESPHome is dat de apparaten ook kunnen werken als een Bluetooth-

proxy voor Home Assistant. Zo kan Home Assistant zijn Bluetooth-bereik vergroten door de ESP32's met ESPHome te laten luisteren naar BLE-packets en directe besturingsverbindingen te maken. Om de bruikbaarheid voor knutselaars te vergroten, hebben we ook BTHome [5] geïntroduceerd, een BLE-protocol voor het verzenden van data die deze Bluetooth-proxy's oppikken (**figuur 1**).

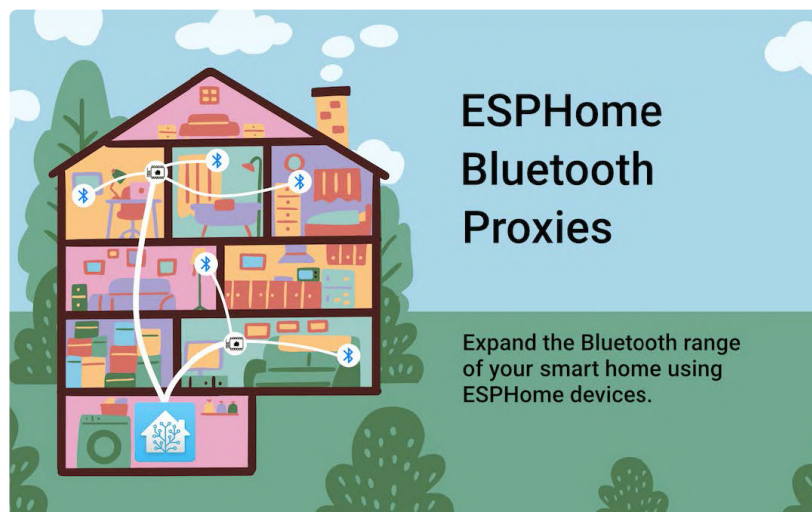
Elektor: ESPHome is geliefd in de community van smart home-hobbyisten. Wat heeft het zo populair gemaakt?

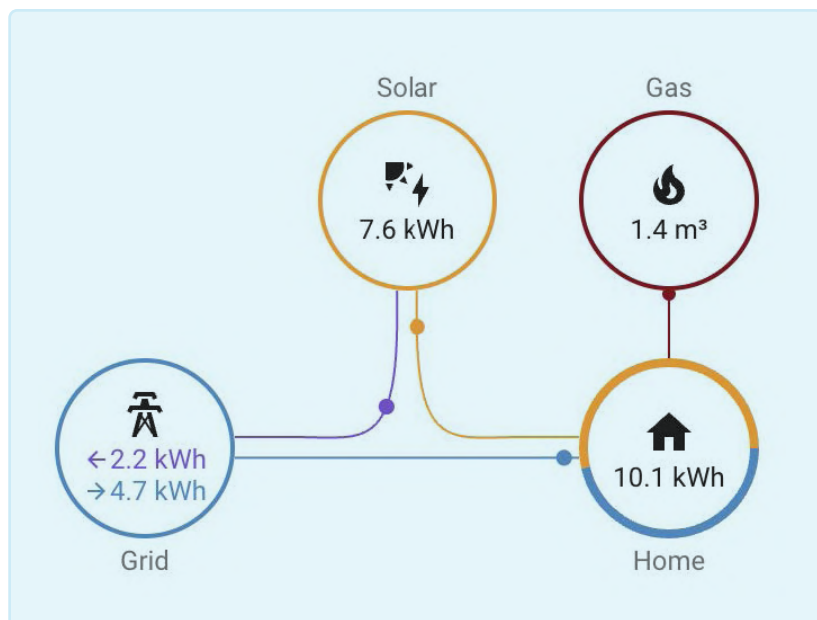
Schoutsen: Gebruiksgemak. We hebben ons er volledig op gericht om gebruikers te laten spelen met hardware. Met ESPHome heb je niet te maken met C++ compilatiefouten of verkeerd geconfigureerde MQTT-topics. En de configuratiefiles kunnen gemakkelijk worden gedeeld met andere gebruikers, zodat beginners meteen werkende apparaten kunnen bouwen.

We hebben ook een tool ontwikkeld om het delen van configuraties gemakkelijker te maken: ESP Web Tools. Dit is een installatieprogramma voor ESP8266's en ESP32's op het web, waarmee gebruikers met één klik vanuit hun browser firmware kunnen installeren op hun apparaten. Gebruikers kunnen nu kant-en-klare apparaten kopen en er ESPHome op installeren. Zo kunnen gebruikers bijvoorbeeld hun M5Stack Atom Echo Dev Kit veranderen in een voice assistant voor Home Assistant [6].

ESP Web Tools was heel handig om onze ESPHome-projecten te delen, maar wij vinden dat dit soort van technologie voor iedereen beschikbaar zou moeten zijn. Daarom werkt het nu ook met de installers van Tasmota, WLED en verschillende andere firmwares.

Figuur 1. Het Bluetooth-bereik van Home Assistant is uit te breiden met ESPHome-proxy-devices (bron: Paulus Schoutsen).





▲
Figuur 2. Energie-
dashboard in
Home Assistant voor een
gemakkelijk overzicht
van het energieverbruik.
Het biedt ook indicatoren
om de afhankelijkheid
van het spanningsnet te
beoordelen (bron: Paulus
Schoutsen).

Elektor: Hoe kan ESPHome helpen bij integreren van een groot aantal sensoren en apparaten in een domotica-omgeving, vooral voor mensen met weinig codeerervaring?

Schoutsen: Het mooie is dat je niet hoeft te kunnen programmeren om ESPHome te gebruiken. Het probleem met elektronica is dat je drie dingen tegelijk probeert te leren: hoe bouw ik de hardware, hoe stuur ik die aan en hoe laat ik die samenwerken met een ander systeem? Met ESPHome kan de gebruiker zich concentreren op de eerste stap: experimenteren met hardware. De rest regelen wij. Zo kunnen de gebruikers zich concentreren op het plezier van hardware bouwen.

Elektor: De configuratie van ESPHome gaat met YAML-bestanden. Waarom hebben jullie daarvoor gekozen en wat zijn de voordelen voor de gebruiker?

Schoutsen: Toen we met ESPHome begonnen, hebben we het laten aansluiten op Home Assistant's architectuur en configuratieformaat. Die passen heel goed bij ESPHome. Een integratie geeft elke verbonden sensor weer. Via die integratie kun je de sensoren configureren, en ze dan met een update naar de microcontroller sturen.

Elektor: De ESPHome-community heeft geholpen om de lijst van compatibele apparaten en componenten uit te breiden. Hoe essentieel is de community voor het project, en hoe beheren jullie die samenwerking?

Schoutsen: De community is onmisbaar voor open source-domotica. Het kost tijd om uit te zoeken hoe een sensor werkt, om aan drivers te komen en dat in ESPHome in te bedden. Onze voltijds medewerkers beoordelen de bijdragen uit de community en integreren die.

Interessante onderwerpen voor ESPHome waren dit jaar voice assistant, Bluetooth-proxy en e-ink-displays:

- Voice assistant verandert een ESPHome-apparaat in een stemassistent die Home Assistant aanstuurt. ESPHome luistert naar de gebruiker en Home Assistant verwerkt de gesproken tekst en voert handelingen uit. Je kunt een voice assistant installeren vanuit een browser [6].
- Bluetooth-proxy verandert een ESP32 in een Bluetooth-proxy voor Home Assistant. Home Assistant gebruikt de BLE van de ESP32 om te luisteren naar packets en verbinding te maken met apparaten om ze te besturen. Je kunt een Bluetooth-proxy installeren vanuit een browser [7].
- E-ink-displays zijn erg in trek. We hebben ondersteuning voor meer van dit soort displays toegevoegd zodat je zelf de fraaiste dashboards voor je slimme huis kunt maken.

Elektor: De beveiliging van IoT- en domoticasystemen is een grote zorg. Welke veiligheids- en privacymaatregelen neemt ESPHome om de data en de connected apparaten van gebruikers te beveiligen?

Schoutsen: Ieder nieuw ESPHome-apparaat is standaard versleuteld met het Noise-protocol. Dat is een snel en efficiënt protocol dat Wireguard en WhatsApp ook gebruiken. Niemand kan dus de ESPHome-data op het netwerk bekijken.

Door ESPHome-apparaten te gebruiken met Home Assistant, is een slim huis volledig open source en werkt het geheel lokaal en versleuteld. Er wordt geen data met anderen gedeeld, en het slimme huis kan zelfs functioneren zonder internetverbinding.

Als ESPHome een beveiligingsupdate bevat, kan het zijn configuratiebestand hercompileren en je apparaat draadloos updaten zodat altijd de laatste versie draait. Het updaten van ESPHome-apparaten kan met Home Assistant.

Elektor: Kun je voorbeelden noemen waar Home Assistant en ESPHome een krachtig of uniek verschil hebben gemaakt?

Schoutsen: Mensen produceren CO₂. Daardoor verandert het klimaat en warmt de aarde op. Eén ding dat we daartegen kunnen doen, is onze energetische footprint zo klein mogelijk maken. Een groot deel van ons energieverbruik zit in onze woning.

In 2021 hebben we energiemanagement geïntroduceerd in Home Assistant [8]. Hiermee kunnen gebruikers hun energieverbruik in de gaten houden, overstappen op duurzame energie en geld besparen (figuur 2).

De belangrijkste informatie is hoeveel energie er wordt getransporteerd tussen ons huis en het



*ESPHome is firmware
voor ESP8266-,
ESP32- en
Raspberry Pi Pico-boards
waarmee gebruikers
gemakkelijk slimme
domotica-apparaten
kunnen maken.*

energienet. Met die informatie kunnen gebruikers beslissen wanneer en hoe ze hun energieverbruik aanpassen door hun e-bike op te laden, de was te doen of de thermostaat lager te zetten.

We kunnen die informatie gemakkelijk uit onze slimme energiemeter halen. Het grootste struikelblok is nog dat de codering per land of per regio verschilt. Samen met onze community hebben we open source ESPHome-apparaten ontwikkeld die die verschillende standaards ondersteunen. De SlimmeLezer van Marcel Zuidwijk [9] bijvoorbeeld wordt rechtstreeks op de P1-poort aangesloten voor real time en cumulatieve informatie over stroom- en gasverbruik. (Dit werkt in Nederland en in enkele andere EU-landen.) Andere elektriciteitsmeters hebben een pulserende LED in plaats van een poort. Elke flits staat voor een bepaalde hoeveelheid energie. Met Home Assistant Glow van Klaas Schoute [10] kunnen die LED-pulsen worden geteld om nauwkeurig het energieverbruik te meten, en Home Assistant volgt het verbruik als functie van de tijd (figuur 3).

Elektor: Welke andere vooruitgang of verbeteringen mogen ESPHome-gebruikers verwachten als het IoT-ecosysteem verandert? Zijn er plannen om ESPHome en Home Assistant samen te voegen?

Schoutsen: Eén van de waarden van Open Home is keuzevrijheid. Dus we garanderen wel een optimale integratie tussen Home Assistant en ESPHome, maar we willen geen mensen uitsluiten van het besturen van ESPHome-apparaten met andere systemen. Er zijn nog meer mogelijkheden voor strakkere integratie. Als iemand een apparaat met ESPHome-firmware koopt, moet hij het ESPHome-dashboard gebruiken om het te configureren en updates te installeren. We willen dat stroomlijnen, zodat gebruikers dat niet allemaal hoeven te leren om op de laatste stand te blijven. Dat is onderdeel van onze voortdurende inspanningen om de productervaring te verbeteren in ons programma *Works with ESPHome*.

Met het *Works with ESPHome*-programma kunnen fabrikanten ons logo voeren als hun apparaten voldoen aan een aantal eisen, zoals functies om gemakkelijk in te stappen en de configuratie van het apparaat aan te passen.

Elektor: Heb je tips voor beginners die aan de slag willen met ESPHome en hun eigen domotica-projecten? Zijn er hulpmiddelen of aanbevolen werkwijzen?

Schoutsen: Begin met een eenvoudig ESP32-development board en een temperatuursensor. Krijg die aan de praat, koppel hem aan Home Assistant, en ga experimenteren met de ESPHome-configuratie. De wijzigingen die je aanbrengt, zie je meteen terug in Home Assistant als je de ESP32 actualiseert. De gemakkelijkste manier om met ESPHome te beginnen, is het te installeren uit de add-on store in Home Assistant [11]. Dat gaat met één klik. Klik op *Add device* en een wizard begeleidt je bij het installeren van je eerste device. Veel plezier met automatiseren!

Elektor: ESPHome staat bekend om zijn gebruikersvriendelijke benadering, maar welke acties of hulpmiddelen zou je aanbevelen voor absolute beginners bij hun eerste ESPHome-project?

Schoutsen: Kijk op YouTube. Daar staan heel veel video's voor beginners.

Figuur 3. Home Assistant Glow van Klaas Schoute (bron: Paulus Schoutsen).



Elektor: Is er ook een plan voor meer gedetailleerde documentatie of cursussen voor beginners om domotica toegankelijker te maken?

Schoutsen: Daar willen we in 2024 naar gaan kijken. Het zou fijn zijn om een gemakkelijke starter-kit voor ESPHome te hebben.

Elektor: Kun je iets zeggen over geplande functionaliteit of upgrades die ESPHome nog gebruikersvriendelijker gaan maken? Zijn er plannen om de gebruikersinterface te verbeteren?

Schoutsen: Als je een ESPHome-apparaat maakt, moet je de inloggegevens van je WiFi-netwerk in de firmware inbouwen. Zo kan het automatisch je netwerk vinden en verbinding maken. Dat is fijn, behalve als je je apparaat wilt delen met vrienden, of het wilt verkopen. Daarom hebben we nu Improv Wi-Fi [12] ontwikkeld. Dat is een standaard om apparaten te configureren met behulp van BLE of USB.

We zijn van plan om ondersteuning voor Improv Wi-Fi over BLE aan Home Assistant toe te voegen. Als je dan een commercieel apparaat aansluit, helpt Home Assistant je bij de installatie en configuratie.

Elektor: ESPHome werkt al met heel veel verschillende sensoren en apparaten. Zijn er plannen om nog meer sensoren en componenten toe te voegen, en zo ja: wat voor soort sensoren kunnen gebruikers gaan verwachten?

Schoutsen: We houden nooit op! We zien de laatste tijd veel millimetergolf-sensoren. Dat is een nieuwe manier om aanwezigheid van mensen in een ruimte te detecteren, zelf als ze heel stil zitten.

We willen ook op zoek naar goedkopere luchtkwaliteitssensoren. Als de lucht in een ruimte niet goed is, beïnvloedt dat je denkvermogen en prestaties. Het zou mooi zijn om goedkope sensoren met ESPHome te maken om mensen te helpen om gezonder te leven.

Elektor: Het kan voor gebruikers lastig zijn om eigen sensoren en apparaten toe te voegen. Zijn er plannen om dat nog meer te vereenvoudigen en makers te helpen bij het aansluiten van zelfgebouwde sensoren aan ESPHome?

Schoutsen: Elke ESPHome-configuratie genereert

een C++ project dat wordt gecompileerd en geïnstalleerd op een apparaat. Als een ontwikkelaar van niet-standaard sensoren een eigen protocol gebruikt, moet daar code in C++ voor geschreven worden om het te integreren in ESPHome. We hebben protocolen zoals BTHome [5] om data te zenden, maar dat is bedoeld om de gegevens in Home Assistant te krijgen.

Elektor: Zijn er ook plannen om Home Assistant om te vormen tot een cloud-service, waarbij ESP32-sensoren data naar de cloud sturen om ze daar te verwerken, in plaats van naar Home Assistant op een lokale server?

Schoutsen: Nee. Een slimme woning moet een lokale server hebben en de data binnenshuis houden. We hebben Home Assistant Green [13] uitgebracht om het gemakkelijker te maken om zelf Home Assistant te draaien. Dit is een kant-en-klaar Home Assistant-systeem en is de gemakkelijkste manier om met Home Assistant te beginnen. Het is te koop voor \$ 99.



Met ESPHome kan de gebruiker zich concentreren op de eerste stap: experimenteren met hardware.

De rest regelen wij. Zo kunnen de gebruikers zich concentreren op het plezier van hardware bouwen.

Elektor: Zijn ESPHome en Home Assistant betrokken bij samenwerkingsverbanden met hardwarefabrikanten, of bij andere projecten om de compatibiliteit en gebruikservaring te verbeteren?

Schoutsen: Ja. We hebben het Made for ESPHome-programma [14] voor samenwerking met ontwerpers die ESPHome-apparaten willen verkopen. Ze moeten dan voldoen aan een aantal eisen om te garanderen dat de apparaten open en aanpasbaar zijn. We hebben ook het Works with Home Assistant-programma [15]. Dit geeft de zekerheid dat we apparaten testen op compatibiliteit en samenwerken met de leverancier om eventuele problemen op te lossen.

teit en samenwerken met de leverancier om eventuele problemen op te lossen.

Elektor: Hoe lukt het Home Assistant om steeds nieuwe eigenschappen te introduceren en de bestaande functies te blijven ondersteunen voor een betrouwbare en gebruikersvriendelijke ervaring?

Schoutsen: Daar zijn veel mensen voor nodig. Vorig jaar was Home Assistant het op één na actiefste open source-project op heel GitHub. Er waren bijdragen van 13.200 mensen. Al die mensen werken eraan om Home Assistant compatibel te houden in de

veranderende domotica-wereld door nieuwe apparaten of nieuwe functionaliteit in al bestaande apparaten te ondersteunen. De kracht van Home Assistant zit in de community die erachter staat. De community maakt Home Assistant het beste platform en geeft ons de beste ideeën om onze huizen slim te maken.

Elektor: Veel nieuwkomers in de domotica en IoT worden misschien afgeschrikt door de leercurve. Heb je tips voor nieuwe gebruikers hoe ze hindernissen en frustraties kunnen overwinnen?

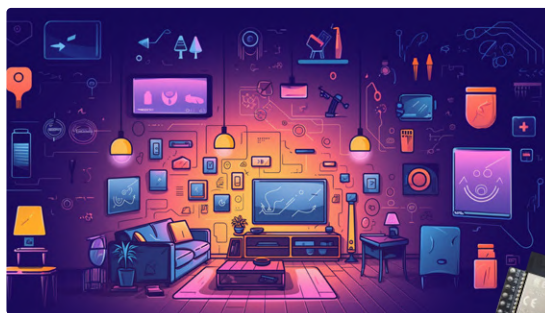
Schoutsen: Maak het in het begin niet te moeilijk; volg de standaard-route. Er zijn andere manieren om Home Assistant te installeren, maar Home Assistant Operating System is de gemakkelijkste weg. Dat geldt ook voor ESPHome: kies eerst voor standaard ESP32-boards, en probeer niet meteen iets exotisch, zoals de ESP32-S3. Dat maakt het alleen maar ingewikkeld.

Elektor: Hoe worden firmware en bug patches afgehandeld? Zijn er plannen om dit proces te automatiseren of te stroomlijnen, zodat gebruikers altijd de meest recente en stabiele versie hebben?

Schoutsen: In onze visie is veiligheid heel belangrijk voor Open Home, dus dat krijgt de hoogste prioriteit bij Home Assistant en ESPHome. We hebben het heel gemakkelijk gemaakt om alles up-to-date

te houden: Gebruikers kunnen alle onderdelen van Home Assistant actualiseren met één muisklik in de interface. Dat geldt ook voor het actualiseren van ESPHome-apparaten als er een nieuwe versie uitkomt. ◀

vertaling: Evelien Snel — 230594-03



Gerelateerde producten

- > Espressif ESP32-C3-DevKitM-1
www.elektor.nl/20324
- > Espressif ESP32-DevKitC-32E
www.elektor.nl/20518



WEBLINKS

- [1] Top open source-projecten: <https://octoverse.github.com/2022/state-of-open-source>
- [2] The 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] ESPHome acquisition: <https://home-assistant.io/blog/2021/03/18/nabu-casa-has-acquired-esphome>
- [5] BTHome protocol: <https://bthome.io>
- [6] Spraakassistent voor Home Assistant: https://home-assistant.io/voice_control/thirteen-usd-voice-remote
- [7] Bluetooth proxy installation: <https://esphome.github.io/bluetooth-proxies>
- [8] Energiebeheer in Home Assistant: <https://home-assistant.io/blog/2021/08/04/home-energy-management>
- [9] SlimmeLezer door Marcel Zuidwijk: <https://slimmelezer.nl>
- [10] Home Assistant Glow door Klaas Schoute: <https://github.com/klaasnicolaas/home-assistant-glow>
- [11] Add-on store in 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] Made for ESPHome: https://esphome.io/guides/made_for_esphome.html
- [15] Works with Home Assistant: <https://partner.home-assistant.io>

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Het co
eenma
meerb
keer na
het feit
wordt o
stroom
In teger
een ext
te flash

Flash

Op de ES
microc
worden
(hoewel
- alleen
gebruikt
het inter

ESP32

Om de fi
modus' z
fase boot
ren de fla
applicati
De down
(meestal
en dan de
de ESP32
flashgehe

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Om F
teren
elke b
Flash
kun je
flashe

ESP
Soms
en ge

je ESP Launchpad rechtstreeks vanuit je Chrome-browser proberen. dig en intuïtief.

 **ESPRESSIF** x  **elektor**

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF

Stel
appa
Flash
host-
een E
Een v
word
re-ve
Mom

- > ESP
- > STM
- > Ras
- > elke

ESPRESSIF x **e**lektor

als host om te flashen.

Computer

ESP32-S2 or ESP32-S3

Vrijgegeven Bonuseditie!

Figuur
(PC)

Veel

In de
nieuw
beste
dat be
grafis
cedur
zonde
Dit ar
en -pr
mogel
werken
grond

Beschikbaar in week 01 2024

Vraag

Hebt u
van di
pedro.
redact

WEB

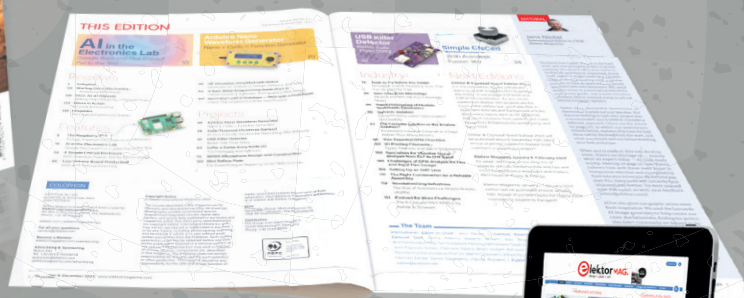
[1] ESP1
[2] ESP
[3] ESP
<https://www.elektormagazine.nl>

ESPRESSIF × eilektor

20%
korting
op het eerste jaar van
uw lidmaatschap

Word lid van de Elektor Community

Neem **nu** een
lidmaatschap!



- ✓ Een compleet web-archief t/m 1980!
- ✓ 8x Elektor Magazine (Print)
- ✓ 8x digitaal (PDF)
- ✓ 10% korting in onze webshop, en exclusieve aanbiedingen
- ✓ Toegang tot meer dan 5000 Gerberfiles
- ✓ Gratis bezorging binnen Nederland en België



www.elektormagazine.nl/gold-member

Gebruik kortingscode

ESPRESSIF20



ESPRESSIF



elektor

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

De draadloze

In dit artikel wordt de opstart van de kernnetwerken overweogen. We beginnen met draadloze netwerken te verdelen in familieleiden met de realisatie van elke knop, al dan niet een of ander maken. We die tegelijk naar verdelen. Dit is ogenblikken al. Doo knop voor knop op korte afstand, maar daar sensoren aan toe te voegen voor het registreren van omgevings-

– bijvoorbeeld ad hoc 'mesh'-netwerken of 'hub-and-spoke'-netwerken enzovoort. Deze netwerken kunnen relatief kleinschalig zijn –

ESPRESSIF × eilektor

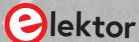
Point to Point

Hub & Spoke

Mesh

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF x 

+ geoptimaliseerd voor laag
energieverbruik

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024



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

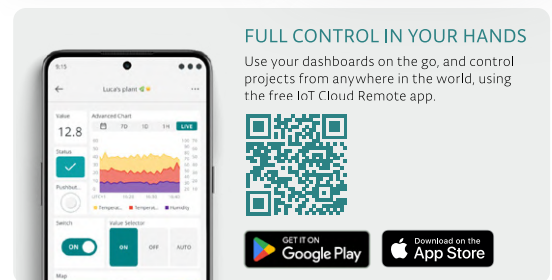
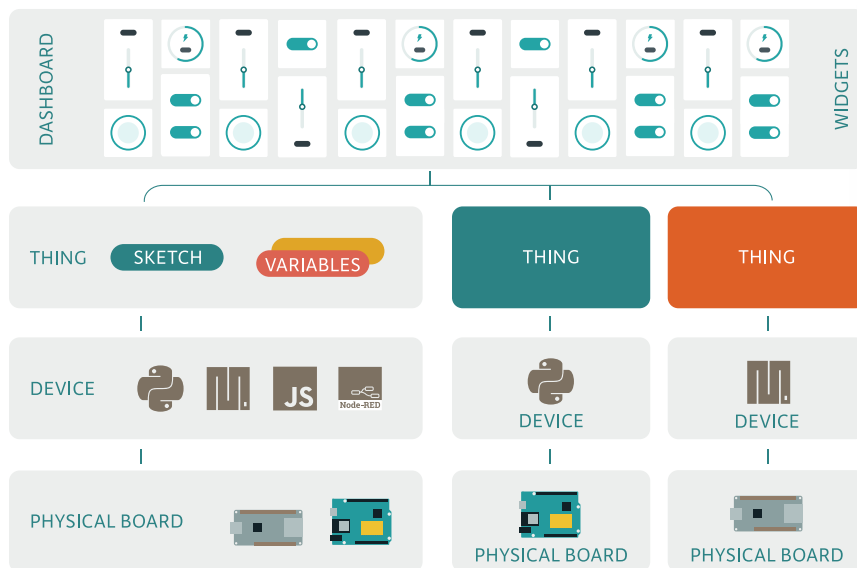
Program/Deploy

- **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?



Compatible hardware

WITHIN ARDUINO DEVELOPMENT ENVIRONMENTS



ARDUINO



ESP32/ESP8266

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

+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

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Het internet (IoT) dringt ons leven. Dat betekent fragmentatie van de IoT-productie industrie ontdekt de toepassing. Belangrijke aspecten. Er komen steeds stringenter

En klanten helpen de productieprocessen te vereenvoudigen.

beveiliging van IoT-apparaten. Meestal heeft een apparaat een publieke sleutel die direct of

ESPRESSIF × **e**lektor

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Figuur 1. Schema van het 'transitive trust'-model voor beveiligd opstarten.

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Figuur 2

Auto flash

We no
symme
waarm
worder
hebben
een sle
voor d
non-re
kan he
firmwa
ESP-II
functie
de eer
softwa
tel gen
tor (tru
de chip
wordt
liging

Figuur 3. Stroomdiagram van veilige uitwisseling van certificaten.

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF × e!lektor

Revolutioneer uw producten met WiFi Motion™,
nu beschikbaar op Espressif Wi-Fi chips.



Ontdek de toekomst van bewegingsdetectie.
Neem nu contact met ons op via www.cognitivesystems.com

COGNITIVE ∞



Borislav Milcevic (e8b407)

14-05-2023

How to make old devices smarter with a ESP8266?

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Espress
deerde
en draa
waarvan
compact
bruik kun
succes w
ten- als i
Hieronde
op de ES
project ge
TechMast
om een e
het leven
Je kunt de
andere or

WEB

- [1] TME-
- [2] Bron



ESPRESSIF × e

lektor

...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

B

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

De
bibl
van
ond
• ES
• ES
• ES
• ES
• ES
en

met Blynk.

ESPRESSIF × e-lektor

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

G

Di
on
tu
ge
ve

In

De
bet
dyn
app
bie
die

Au

Kan
de d
geb
of s
hard

Uw

mo

Afha
volge

W

- [1] C
- [2] E
- [3] E
- [4] B
- [5] V
- [6] K

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Steliau
omsch
meerv
gen. M
aanraa
allemaa
beheers
gemaal
Steliau
strategi
gevende
zijn posi
tiviteit k
is al lang
van Esp
loop der
ele distr
Steliau T
Espressi
Steliau T
volledig
het gehe
ten, zowe
embedde
over gro
betrekkir

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

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF × **e**lektor

Figuur 1. De M5Stack ecosysteem-familie.



ACK

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Fig

De
wo
UIF
wa
typ
ner
ont
Me
geb
plat
trial
Sma
effic
IoT-c

Nie

De o
prod
Sma
deve
versc
die n
(figu

W

[1] T
[2] E
rc

ESPRESSIF × e!lektor

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF × **e**lektor

Figuur 2
met de liv

UTF-8, Col 49 Spaces: 4 UTF-8 LF Slint

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

Bemachtig de nieuwe

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ES
ES
op
ge
So
te
tra
ES
ee
on
kit
uit

w

o



ESPRESSIF × **e**lektor

ESP32-C3-DevKitM-1 op een breadboard monteren.

www.elektor.nl/20324

Internet of Things-toepassingen.

www.elektor.nl/19333

Vrijgegeven Bonuseditie!

Beschikbaar in week 01 2024

ESPRESSIF × **e**lektor

www.elektor.nl/20558

www.elektor.nl/19736