vos premiers pas avec l'**ESP32-C3** et l'**IdO**

Bouton Wi-Fi + relais Wi-Fi

Mathias Claußen (Elektor)

L'IdO n'est pas un livre fermé sur des secrets. De puissants contrôleurs, par ex. le nouvel ESP32-C3, et des environnements conviviaux pour les débutants comme l'EDI Arduino font du développement de petits projets un jeu d'enfant.

En parlant de l'IdO (en anglais IoT = *Internet of Things*), nous admettons que de plus en plus de choses de notre quotidien sont connectées à l'internet. Cela va de l'éclairage aux capteurs domotiques en passant par les voitures, feux de circulation, conteneurs d'expédition et bien plus encore. Dans ces *objets* connectés sont installées des puces interfaçables avec le réseau, ce qui permet d'échanger des informations. La meilleure façon d'aborder la connexion de vos propres applications à l'IdO est de construire un montage pratique simple qui, par ex., crée un lien entre un bouton-poussoir Wi-Fi et un relais Wi-Fi : le bouton peut activer à distance le relais qui renvoie son état au bouton.

Sélection des composants

Soyons ordonnés, choisissons d'abord les composants adéquats. Le kit ESP-C3-12F (**fig. 1**), disponible dans la boutique Elektor s'impose de lui-même. La carte est équipée d'un µcontrôleur ESP32-C3 d'Espressif avec fonction Wi-Fi. L'ESP32-C3 succède à l'ESP8266, qui a fait ses preuves. Outre un noyau CPU moderne, la puce offre un bon choix de périphériques intégrés à la fois faciles à utiliser et puissants. (Consultez notre article sur l'ESP32-C3 [1].) La **figure 2** donne un aperçu des blocs matériels intégrés dans la puce. En plus de l'ESP32-C3, la carte accueille une LED RVB et un convertisseur série-USB. Notre projet

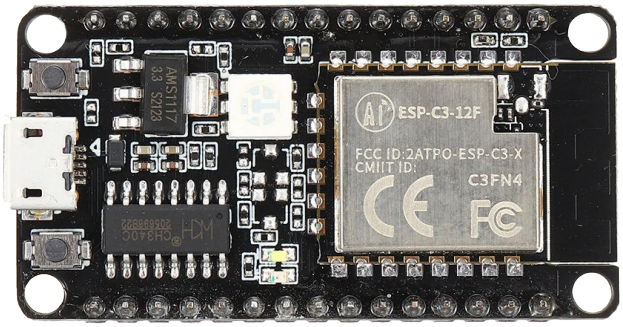


Figure 1. Kit ESP32-C3-12F.

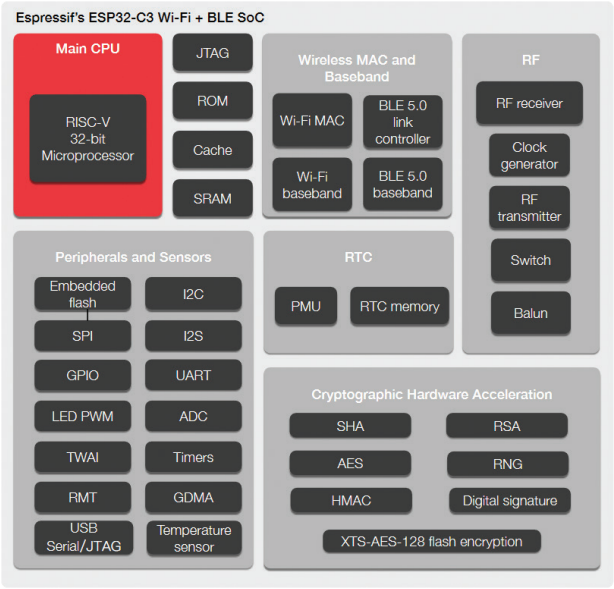


Figure 2. Blocs fonctionnels de l'ESP32-C3 (source : fiche technique de l'ESP32-C3).

nécessite deux kits ESP-C3-12F.
Un capteur et un actionneur sont également nécessaires. Le kit de capteurs 37-en-1 d'Elektor tombe à pic ! Il comprend 35 capteurs (au départ il y en avait 37, mais deux d'entre eux contenaient du mercure et ont été retirés pour des raisons de sécurité). La **figure 3** montre la boîte du kit, la **figure 4** et le document d'information [2] détaillent son contenu. Nous prendrons d'abord le module joystick et utiliserons sa fonction bouton-poussoir comme entrée de commande du système. Une fois connecté à l'autre carte à µcontrôleur, le module relais peut agir comme un actionneur dans le système. Des fils volants (femelle/femelle) sont nécessaires pour connecter les modules. Le kit de fabrication Pimoroni « Mini Breadboards & Jumpers » en contient. (Voir l'encadré **Produits**).
Il nous faut aussi un *serveur local* pour que les appareils IdO puissent échanger leurs données ; un Raspberry Pi sera parfait. Un Raspberry Pi original version 1 conviendrait, mais, pour cette application, nous conseillons une version 2 ou ultérieure. Le Wi-Fi ne fut pas intégré avant le Raspberry Pi 3B, donc pour utiliser les versions antérieures, il faut ajouter soit une simple clé Wi-Fi, soit un câble Ethernet. Si le petit, mais puissant, Raspberry Pi Zero vous séduit, jetez un coup d'œil au kit Raspberry Pi Zero 2 W (carte avec boîtier et câble). Pour ce projet, il n'est toutefois pas obligatoire d'utiliser un Raspberry Pi. N'importe quel PC sous Linux (distribution Ubuntu par ex. [3]) fera tout à fait l'affaire.
Avant d'entrer dans le vif du sujet, voyons de plus près comment se passent le contrôle et l'échange de données dans cette configuration.

MQTT

Tout dispositif IdO (capteur ou actionneur) doit transférer des données. Pour ce faire, nous pouvons réinventer la roue en développant notre propre protocole de communication, ou bien utiliser un protocole standard éprouvé. Très utilisé, MQTT répond à cette demande. Cet acronyme signifie *Message Queuing Telemetry Transport* (transport de télémétrie par file d'attente de messages), mais cela ne rend plus compte de la fonction actuelle du système qui a évolué. En 2013, il fut officiellement décidé que MQTT serait une norme [4].

Figure 3. Kit de capteurs Elektor 37-en-1.

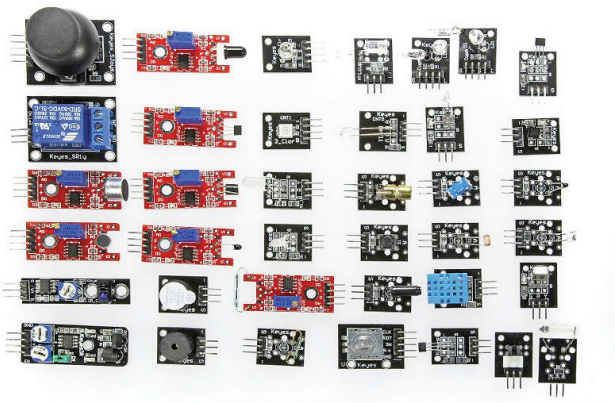


Figure 4. Toute une série de capteurs et d'actionneurs est incluse.

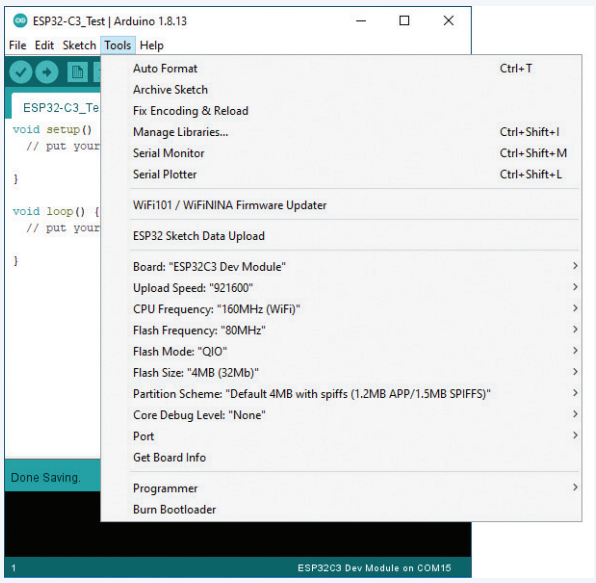


Figure 5. Configuration de l'ESP32-C3 dans l'EDI Arduino.

MQTT prend en charge l'échange de messages à l'aide d'un *broker* (courtier), notre serveur local, sans spécifier le contenu des messages. Cela peut se comparer à l'envoi d'une lettre : la logistique et le format de l'enveloppe sont spécifiés par l'entreprise postale, mais son contenu (message et langage employé) est du ressort exclusif de l'utilisateur. Quel « langage » choisir pour envoyer nos messages ? Il y a diverses possibilités, mais le format JSON (*JavaScript Object Notation*) s'est imposé (pas seulement pour MQTT).

JSON

C'est un format léger d'échange de données pour transférer des messages. Le code est facilement produit et interprété, même par de petits µcontrôleurs. En outre, toute personne comprend, mais rédige aussi facilement un script JSON. Le site de la norme JSON [5] vous donne un aperçu de ses spécifications. L'emploi de JSON ne se limite pas à MQTT, mais concerne bien d'autres domaines. JavaScript, le langage de programmation dont JSON est dérivé, est l'un des piliers sur lesquels repose aujourd'hui le World Wide Web. Le site de Mozilla [6] présente agréablement JSON avec des exemples pratiques.

Configuration de l'environnement IdO : le courtier MQTT

Comme dans tout projet, une préparation adéquate évite les mauvaises surprises. Pour traiter les messages MQTT, il nous faut un périphérique de courtage situé soit sur l'internet, soit sur notre réseau local. Un courtier local implique que la fonction IdO n'aura pas besoin de services en nuage car, dans notre cas, elle ne servira qu'à transférer des messages entre appareils connectés localement. Un PC retiré du service ou un Raspberry Pi pourront faire l'affaire. Node-RED nous fournit une boîte à outils complète pour développer des applications réseau non limitées au traitement des messages MQTT. Chez Elektor, nous utilisons fréquemment Node-RED [7] pour traiter les données MQTT. Des instructions détaillées permettent une installation rapide sur Raspberry Pi [8] ou sur PC [9].

L'EDI Arduino

L'EDI Arduino constitue un bon environnement de développement. Il n'est pas le meilleur de sa catégorie, mais offre actuellement la prise en charge de l'ESP32-C3 la plus stable. Gratuit, l'EDI Arduino [10] peut être téléchargé et installé depuis la page d'accueil Arduino. Installons ensuite la prise en charge de l'ESP32 par Arduino comme décrit dans la documentation Espressif [11]. Les réglages de la carte sont effectués comme indiqué à la **figure 5**.

Pour nos premiers pas, outre la prise en charge de l'ESP32, nous aurons besoin de quelques bibliothèques. Pour que l'ESP32-C3 puisse envoyer des données via MQTT/JSON, notre exemple fait appel aux bibliothèques *PubSubClient* de Nick O'Leary et *ArduinoJson* de Benoit Blanchon. Installons-les à l'aide du gestionnaire de bibliothèques de l'EDI Arduino (**fig. 6** et **7**).

Figure 6. PubSubClient dans le gestionnaire de bibliothèque Arduino.

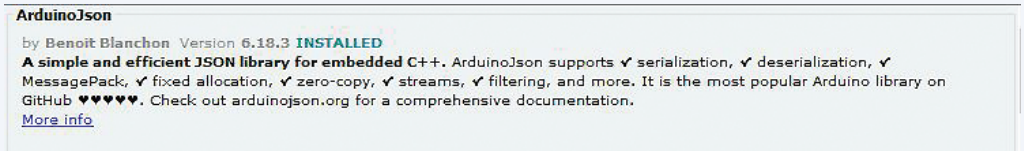


Figure 7. ArduinoJson dans le gestionnaire de bibliothèque Arduino.

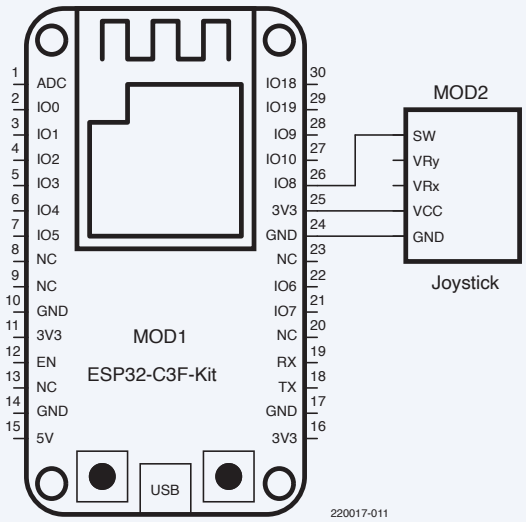


Figure 8. Schéma de l'ESP32-C3 et du joystick.

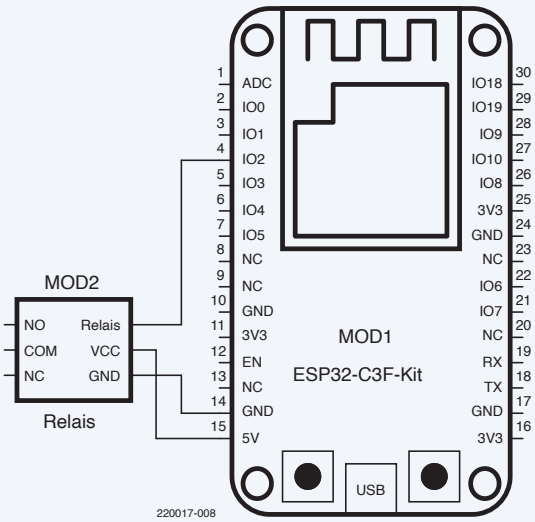


Figure 9. Schéma de l'ESP32-C3 et du relais.

Assemblage du matériel

Les **figures 8** et **9** donnent le schéma de connexion des deux cartes à ESP32-C3. Il n'y a que trois fils de liaison entre le module joystick ou relais et la carte ESP32-C3 correspondante. Notez que le relais s'alimente sur le 5 V de l'USB de la carte ESP32. La **figure 10** montre les modules et cartes câblés.

Configuration du logiciel

Le code source de ce projet est également disponible sur GitHub [12]. Vous y trouverez aussi les croquis pour les deux contrôleurs ESP32-C3. Avant de les transférer sur les ESP32, il faudra entrer quelques informations sur votre réseau local dans ces fichiers. Tout doit être réglé correctement pour que les deux contrôleurs puissent échanger des données avec le courtier MQTT local. C'est le rôle des directives `#define` présentes au début des deux croquis Arduino :

```
#define WIFI_SSID "changeme"
#define WIFI_PASS "changeme"
#define MQTT_SERVER "test.mosquitto.org"
```

Ces trois `#define` doivent refléter votre propre réseau. Le SSID et le PASSWORD de votre réseau doivent être saisis entre guillemets. L'adresse IP interne au réseau de l'ordinateur Node-RED est spécifiée pour le courtier MQTT. Une fois les deux croquis (relais et bouton) modifiés, ils peuvent être *téléversés* sur leur ESP32-C3 respectif. On peut alors mettre les deux ESP32 sous tension et la grande LED de chacun doit commencer à clignoter en blanc. Cela indique que l'ESP32-C3 tente de se connecter au réseau Wi-Fi. Dès qu'une carte y parvient, sa LED s'allume en continu. La couleur de la LED dépend de la fonction de la carte : celle de la carte du relais connecté s'allume en blanc ; selon que le relais est activé ou non, celle de la carte du

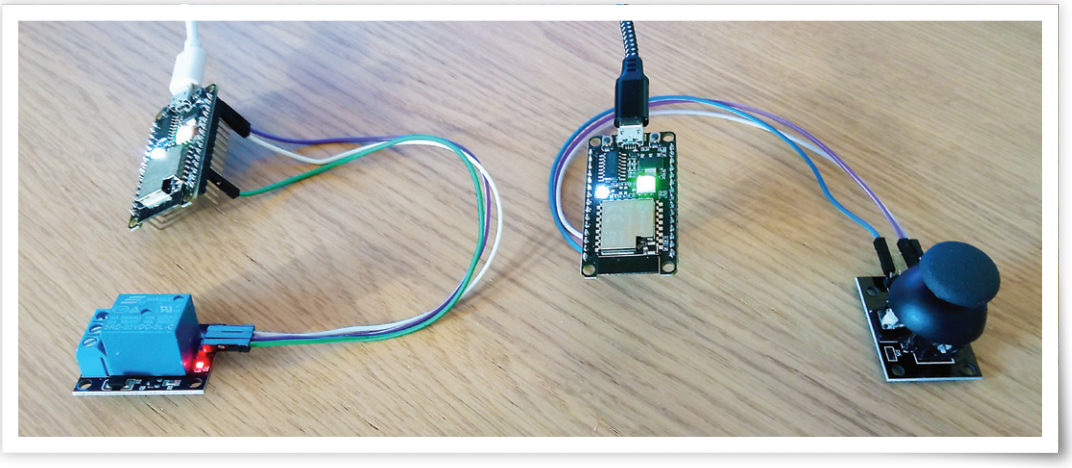


Figure 10. Câblage complet du matériel.

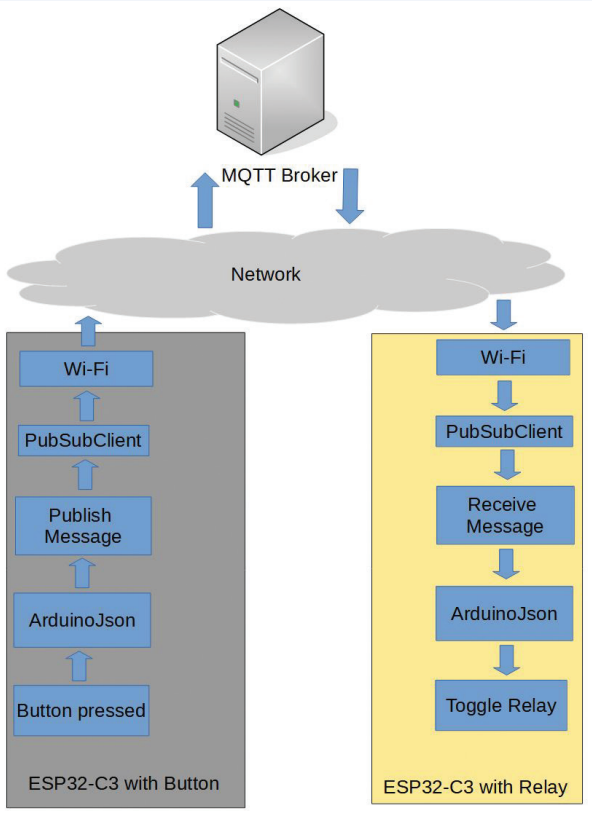


Figure 11. Transfert de données bouton-courtier-relais.

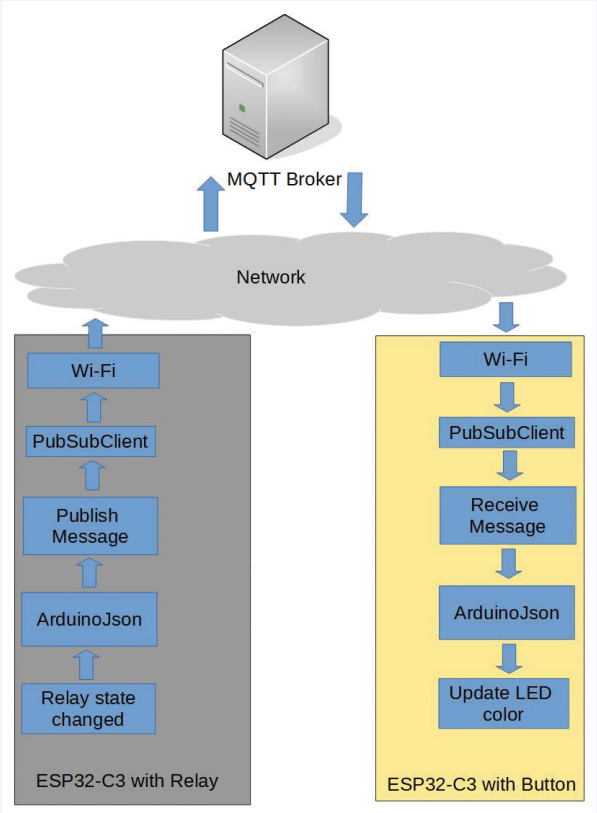


Figure 12. Retour relais-courtier-bouton.

bouton-poussoir connecté s'allume en rouge ou en vert.
Si c'est le cas, les deux ESP32-C3 fonctionnent correctement. En appuyant sur le bouton, l'état du relais change et la couleur de la LED passe du rouge au vert et vice versa. Le bouton d'un ESP32-C3 peut donc commander le relais de l'autre ESP32-C3, et il reçoit également des informations sur l'état du relais. Il est temps de fêter ça. Votre première application IdO fonctionne ! Mais comment l'échange de données se fait-il précisément ?

Aller-retour relais-bouton

Commençons par le chemin suivi jusqu'au relais quand on presse le bouton. La **figure 11** illustre comment le message est emballé couche par couche, puis envoyé au courtier par Wi-Fi. L'instruction `client.publish(MQTT_TOPIC_OUT, (const uint8_t*)buffer, n, true);` du code source s'en charge. Pourquoi cette fonction s'appelle-t-elle `publish` (publier) et non `send` (envoyer) ? Cela vient du mode ultérieur de distribution des données dans MQTT. Le courtier MQTT distribue les messages selon un `TOPIC` (sujet) ; ici, le sujet `(MQTT_TOPIC_OUT)` est « BUTTON ». En se connectant au courtier MQTT, le client (c'est-à-dire l'ESP32-C3 de notre relais) peut indiquer le sujet qui l'intéresse (c.-à-d. s'abonner à ce canal d'information).
Chaque participant ayant informé le courtier MQTT qu'il est intéressé par un sujet donné recevra les messages publiés sur ledit sujet. L'expéditeur, quant à lui, ne s'occupe pas de la distribution. Il ne fait qu'envoyer (« publier ») ses messages au courtier.
L'ESP32-C3 du relais s'abonne au sujet *BUTTON* dans son code en utilisant `client.subscribe(MQTT_TOPIC_IN);` et ici `MQTT_TOPIC_IN` est « BUTTON ». Quand le bouton déclenche l'envoi d'un message, il arrive au courtier MQTT, puis l'ESP32-C3 du relais le reçoit et fait basculer ce relais.
Lorsque le relais change d'état, son contrôleur envoie au courtier MQTT un message dans lequel figure cet état (activé ou non) sous le sujet « RELAIS » *(RELAY)*. L'ESP32-C3 du bouton, qui s'est abonné au sujet « RELAIS » auprès du courtier MQTT reçoit donc le message avec le nouvel état, comme illustré à la **figure 12**. La couleur de la LED passe alors au rouge ou au vert.
L'intérêt de cette structure est qu'un 2e couple bouton-contrôleur peut être mis en réseau pour publier lui aussi des messages sous le sujet « BUTTON ». L'ESP32-C3 du relais répondra alors aux données de commande de l'un ou l'autre bouton et effectuera la commutation correspondante. Si vous souhaitez expérimenter davantage avec MQTT, regardez les autres projets d'Elektor utilisant MQTT pour véhiculer des données. Par ex. la station météo [13] et l'horloge à LED géante avec capteur de température externe [14]. Pour en savoir plus sur MQTT

et sur son utilisation dans l'envoi de données à des plateformes du nuage, consultez par ex. la série « Mon voyage dans le nuage » [15].

Une plateforme idéale pour l'IdO

L'IdO ça peut être simple ! Comme exposé ici, la technologie est assez mature, et vous pouvez rapidement développer des dispositifs IdO à l'aide d'outils conviviaux et de composants appropriés. Bien sûr, les applications peuvent être plus complexes qu'un bête bouton-poussoir et un relais. Du pilotage du chauffage domestique à la sonnette d'entrée, de nombreuses applications pratiques peuvent exploiter la connectivité IdO. L'ESP32-C3 est une carte puissante et peu coûteuse qui constitue une plateforme idéale pour donner vie à vos idées !

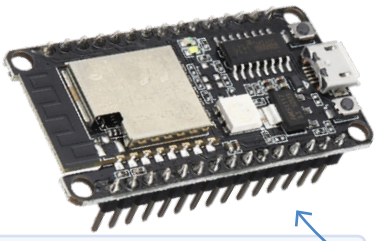
220017-04

Contributeurs

Idée et texte : **Mathias Claußen**
Rédaction : **Jens Nickel**
Traduction : **Yves Georges**
Mise en page : **Giel Dols**

Des questions, des commentaires ?

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



PRODUITS

- **Carte de développement ESP-C3-12F avec 4 Mo de flash**
  www.elektor.fr/19855
- **Kit de capteurs Elektor 37-en-1**
  www.elektor.fr/16843
- **Kit Raspberry Pi Zero 2 W**
  www.elektor.fr/19952
- **Les indispensables du DIY, Pimoroni - Miniplaques d'essai et câbles de liaison**
  www.elektor.fr/18430

LIENS

[1] « Prise en main du microcontrôleur ESP32-C3 RISC-V », M. Claußen, Elektor 01-02/2022 : www.elektormagazine.fr/210466-04
[2] Documentation (en anglais) du kit de capteurs Elektor 37-en-1 : www.elektor.com/amfile/file/download/file/1170/product/6171/
[3] Distribution Linux Ubuntu : https://ubuntu.com/
[4] Procès-verbal du TC OASIS MQTT du 25/04/2013 : www.oasis-open.org/committees/download.php/49028/OASIS_MQTT_TC_minutes_25042013.pdf
[5] JSON.org : www.json.org/json-fr.html
[6] « Travailler avec JSON », Mozilla Web Docs : https://developer.mozilla.org/fr/docs/Learn/JavaScript/Objects/JSON
[7] Node-RED : https://nodered.org/
[8] Installation de Node-RED sur le Raspberry Pi : https://nodered.org/docs/getting-started/raspberrypi
[9] Installation de Node-RED sur un PC : https://nodered.org/docs/getting-started/local
[10] Téléchargement de l'EDI Arduino : www.arduino.cc/en/software
[11] Instructions d'installation Espressif Arduino-ESP32 : https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html
[12] Dépôt GitHub d'Elektor : https://github.com/ElektorLabs/220017-ESP32-C3-and-IdO_First-steps
[13] « Station météo à ESP32 », R. Aarts, Elektor 01-02/2019 : www.elektormagazine.fr/180468-04
[14] « Horloge à LED géante avec Wi-Fi et mesures météo », M. Claußen, Elektor 05-06/2019 : www.elektormagazine.fr/180254-04
[15] « Mon voyage dans le nuage IoT », J. Nickel : www.elektormagazine.fr/search?query=mon+voyage+dans+le+nuage