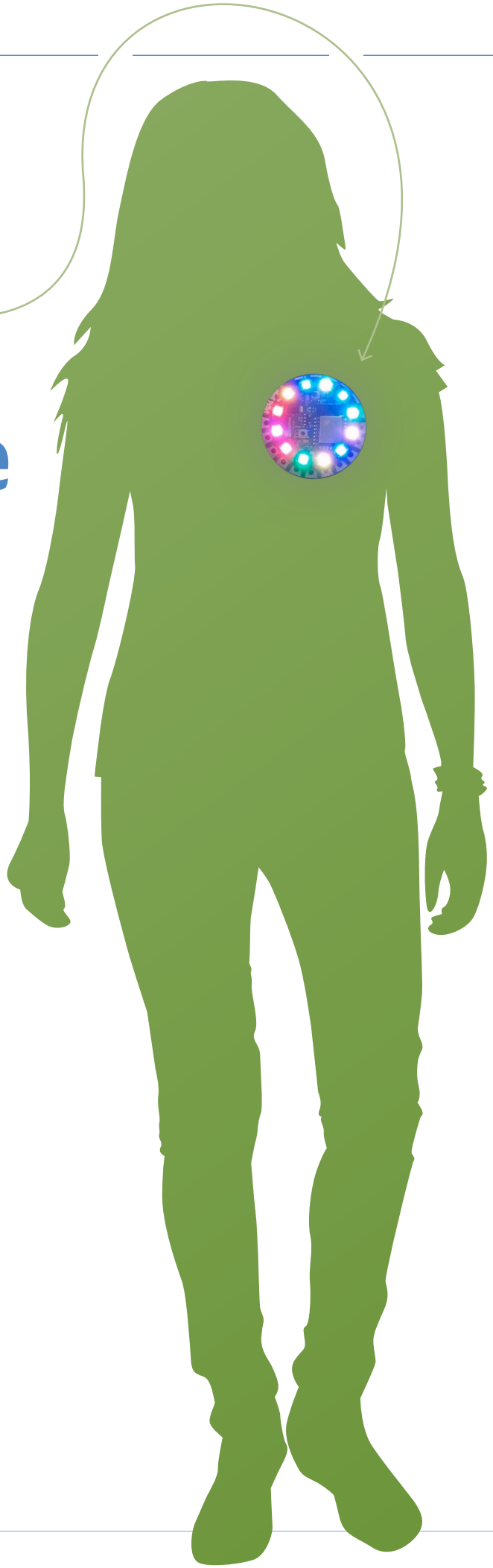
gadget Wi-Fi vestimentaire

ESPHome à nouveau à la manœuvre !



Clemens Valens (Elektor)

Vous connaissez ces caisses pleines de cartes que vous gardez pour un jour où vous aurez plus de temps ? Ayant besoin d'un module ESP8266 pour mes expériences de domotique, j'y ai jeté un coup d'œil et j'ai trouvé une carte « vestimentaire » à base d'ESP8266 que j'avais conçue il y a quelques années. Rien de spectaculaire, juste un module ESP-12E avec une passerelle USB-série et un pilote pour une chaîne de LED adressables WS2812. J'ai redonné vie à cette carte – bien sûr avec l'aide d'ESPHome !

Il y a quelques années, un collègue m'a demandé de concevoir une carte à microcontrôleur « prête à porter » intégrant un module Wi-Fi à base d'ESP8266 ; il s'occuperait de l'aspect micrologiciel, car il avait d'ambitieux projets pour un tel montage. Lorsque le prototype a été prêt, il a essayé la carte, puis a quitté l'entreprise. Ma conception l'avait-elle déçu ou découragé à ce point ? Je ne l'ai jamais su.
Le projet a été abandonné, et il aurait sombré dans l'oubli si, quelques

mois plus tard, je n'avais pas eu un besoin urgent d'un module ESP8266 pour mes expériences de domotique. En fouillant dans des caisses remplies d'objets qui pourraient m'être utiles un jour, je suis tombé sur mon prototype ESP8266 vestimentaire. Et, comme j'étais obsédé par ESPHome [1] à l'époque, j'ai immédiatement compris que j'avais maintenant les outils pour enfin créer le logiciel pour cette carte.

Un circuit de type NodeMCU

Le schéma de la carte (**fig. 1)** est en gros un NodeMCU où la passerelle USB-série CP2101 de Silicon Laboratories (Silabs) a été remplacée par le FT231XS de FTDI, beaucoup moins cher. De plus, j'ai ajouté un port pour une chaîne de LED adressables à base de WS2812 (NeoPixels si vous préférez).

Comme le gadget vestimentaire est quasiment un NodeMCU, tout ce qui suit est également valable pour un module NodeMCU normal. Le logiciel présenté fonctionne également très bien.

Les ports GPIO disponibles et l'alimentation sont amenés sur des *pads* spéciaux avec de grands trous placés tout autour de la carte ronde. Ces *pads* sont prévus pour être utilisés avec un filetage conducteur, mais vous pouvez bien sûr aussi y souder des fils ou utiliser des pinces crocodiles.

L'alimentation du circuit est fournie soit par le port micro-USB, soit par la connexion d'une alimentation externe de 5 V à l'une des broches de 5 V. C'est utile lorsque vous utilisez de longues chaînes de LED qui demandent plus d'énergie que ce qu'un port USB normal peut fournir. On peut aussi choisir une banque d'alimentation USB à haute capacité. Notez que l'interface est destinée à des chaînes de LED de 5 V. Notez également que l'entrée 5 V n'a pas de protection contre l'inversion de polarité.

Conception du logiciel avec ESPHome

Pour mes expériences, j'ai connecté une chaîne de douze LED WS2812 en forme d'anneau au port K2 (**fig. 2)**. Bien sûr, vous pouvez écrire le logiciel de cette carte en partant de zéro, comme mon ancien collègue avait prévu de le faire, et aurait dû probablement faire, car il n'y avait pas autant de code ESP8266 disponible qu'aujourd'hui, mais adopter un projet *open source* comme ESPHome vous épargne une énorme quantité de travail.

Je l'ai déjà dit et je le répète encore une fois, ESPHome permet de créer en quelques minutes une application connectée pour l'ESP8266 ou l'ESP32, avec une programmation *over-the-air* (OTA), un *hotspot*

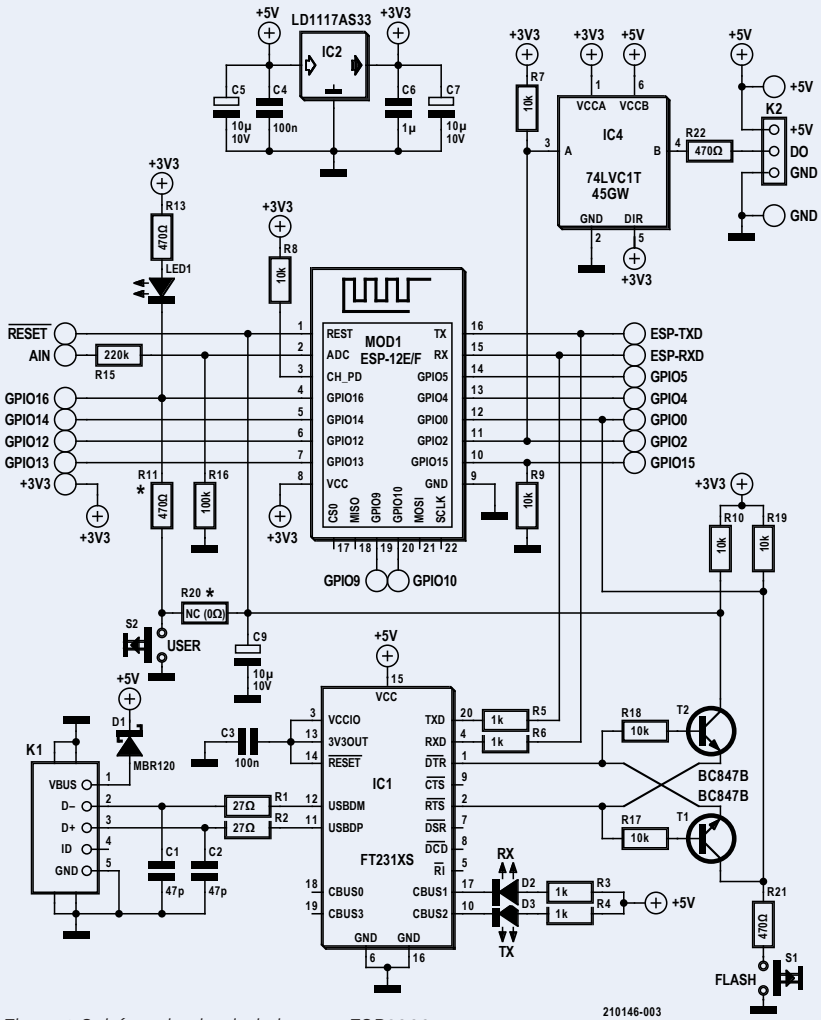


Figure 1. Schéma du circuit de la carte ESP8266.

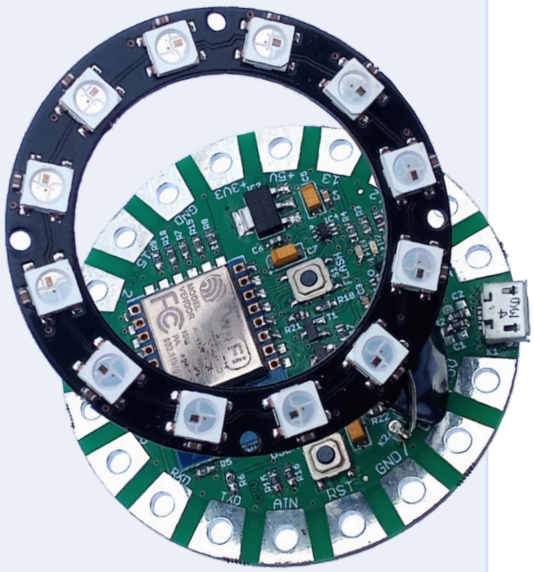


Figure 2. Le prototype du Gadget Wi-Fi vestimentaire, équipé d'un anneau de 12 LED RVB adressables (type NeoPixel).

Listage 1. Le fichier de configuration YAML [2].

```
# Elektor 160112 Wearable ESP8266
 # Configuration file for ESPHome

esphome:
   name: wearable
   platform: ESP8266
   board: nodemcu

wifi:
   ssid: "my_ssid"
   password: "my_passphrase"
  ap:
     ssid: "Wearable Fallback Hotspot"
     password: "12345678"

captive_portal:

# Enable logging
 logger:

# Enable Home Assistant API
 api:

# Enable Over-the-Air updates.
 ota:

output:
   - platform: gpio
     id: "blue_led"
     pin:
       number: GPIO16
       inverted: True

light:
   - platform: binary
     name: "Blue LED"
     output: "blue_led"
   - platform: neopixelbus
     name: "Light Ring"
     num_leds: 12
     type: GRB
     pin: GPIO2
     method: ESP8266_UART1
     effects:
       - addressable_color_wipe:
           name: "Color Wipe"
       - addressable_fireworks:
           name: "Fireworks"
       - flicker:
           name: "Flicker All"
       - addressable_flicker: # Doesn't work?
           name: "Flicker Individually"
       - addressable_rainbow:
           name: "Rainbow"
       - random:
           name: "Random All"
       - addressable_scan:
           name: "Scan"
       - strobe:
           name: "Strobe All"
       - addressable_twinkle:
           name: "Twinkle"
       - addressable_random_twinkle:
           name: "Twinkle Random"

# GPIO11 is somehow related to flash and should
not be used.
 switch:
   - platform: gpio
     name: "GPIO4"
     pin: GPIO4
   - platform: gpio
     name: "GPIO5"
     pin: GPIO5
   - platform: gpio
     name: "GPIO12"
     pin: GPIO12
   - platform: gpio
     name: "GPIO14"
     pin: GPIO14
   - platform: gpio
     name: "GPIO15"
     pin: GPIO15

# Pushbutton on GPIO0.
 binary_sensor:
   - platform: gpio
     name: "Flash"
     pin:
       number: GPIO0
       inverted: True

sensor:
   - platform: adc
     name: "Analog Input"
     pin: A0
     update_interval: 60s
     filters:
       - multiply: 3.2 # voltage divider is 100k/
(220k+100k)
```

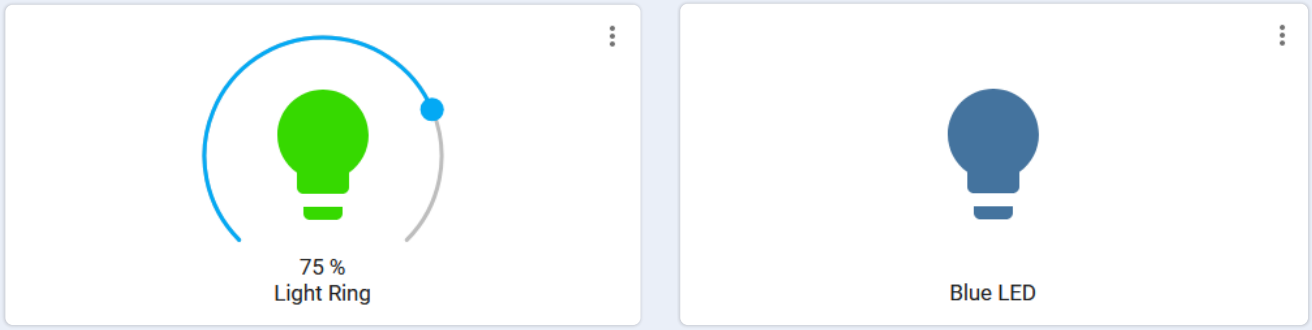


Figure 3. Ces cartes dans Home Assistant permettent de contrôler l'anneau de LED et la LED bleue du gadget Wi-Fi vestimentaire.

de secours, une interface utilisateur de serveur web et des interfaces pour plus de 200 appareils. Vraiment ! Reportez-vous à [1] pour plus de détails.

ESPHome utilise une approche modulaire où des blocs de code prêts à l'emploi sont combinés pour former une application. Les blocs requis par l'application sont répertoriés dans un fichier de configuration dit YAML (pour « *Yet another markup language* », en français « Encore un autre langage avec balises ») ; ce n'est pas un langage, mais un ensemble de règles de formatage de fichiers texte pour spécifier des paramètres et des valeurs [1]. Chaque bloc est configuré individuellement, pour spécifier par ex. le ou les ports GPIO qu'il doit utiliser, ou le protocole de communication ou son type.

Le fichier de configuration est lu par ESPHome et transformé en code C++, qui est ensuite compilé en un exécutable qui peut être programmé dans la mémoire flash du module. Une fois que vous avez compris comment composer un fichier de configuration, c'est parti. Voyez [1] pour plus de détails.

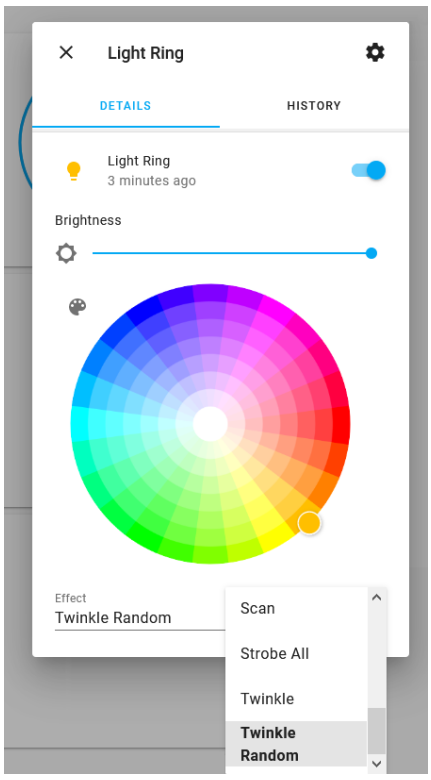


Figure 4. Les effets lumineux déclarés dans le fichier de configuration YAML de ESPHome s'affichent dans Home Assistant sous forme d'une liste dans laquelle vous pouvez choisir celui qui vous convient.

Gros plan sur le fichier de configuration

Le fichier de configuration YAML (**listage 1)** commence par la section `esphome:` obligatoire pour spécifier un nom, le MCU utilisé (ESP8266) et le type de carte (NodeMCU).

Vient ensuite la section Wi-Fi pour indiquer le réseau auquel se connecter et les options de secours en cas de problèmes de réseau. Notez que l'ordre des sections n'a aucune importance.

La spécification de l'option `logger:` active la sortie d'état sur le port série. L'option `api:` permet une intégration facile avec le logiciel de contrôle domotique gratuit et *open source* Home Assistant (voir [1]). L'option `ota:` permet de programmer le dispositif sans fil (à partir de Home Assistant, par ex.), c'est très pratique, car il n'est plus nécessaire d'avoir une connexion physique avec le dispositif.

Vient ensuite l'essentiel de l'application, à commencer par la spécification que le port GPIO16 doit être une sortie. Il le faut si vous voulez utiliser la LED qui y est connectée comme un dispositif *light* (éclairage), ce qui est intéressant – car les éclairages ont différentes options comme des *switches* (interrupteurs) (**fig. 3**).

Éclairages

Comme *light*, j'ai défini la LED bleue sur le port GPIO16 et la chaîne de LED. Cette dernière est gérée par la plateforme *neopixelbus*, où l'on doit spécifier quelques options comme la longueur de la chaîne et le port auquel elle est connectée. Dans ce cas, le port est GPIO2, ce qui permet d'utiliser la méthode *ESP8266_UART1*. En fait, lorsque vous spécifiez cette méthode, vous n'avez pas besoin de spécifier GPIO2, car cela est implicite.

Les éclairages peuvent avoir des effets, et ESPHome en a intégré quelques-uns que vous pouvez utiliser (si votre éclairage les supporte, bien sûr). Il va sans dire que vous pouvez également programmer vos propres effets lumineux. J'ai spécifié la plupart des effets intégrés pour l'anneau de LED. Les effets peuvent avoir des paramètres, mais sans eux, les valeurs par défaut seront utilisées. Home Assistant vous permet de choisir quel effet est actif (**fig. 4**). J'aime l'effet de scintillement aléatoire.

Autres entrées et sorties

Les ports GPIO inutilisés sont déclarés comme des interrupteurs afin que vous puissiez les activer et les désactiver depuis Home Assistant. En domotique, un interrupteur est un dispositif qui est contrôlé par le

système (par ex. un relais). Un interrupteur commandé par l'utilisateur (ou l'occupant) est un capteur binaire. Un bouton-poussoir est connecté à GPIO0 et il est donc spécifié comme un capteur binaire.
Enfin, la carte possède une entrée analogique sur la broche A0 avec un diviseur de tension devant elle (R15 & R16), ce qui explique le facteur de multiplication de 3,2 pour reconvertir la tension d'entrée divisée en volts. La tension d'entrée maximale est de 3,3 V.

Construire une alarme de distanciation sociale

ESPHome permet l'automatisation et vous pouvez ajouter un capteur de proximité à la carte pour changer la couleur de l'anneau de LED en fonction de ce que le capteur voit. De cette façon, la carte pourrait servir d'alarme de distanciation sociale. On peut en faire un simple bijou électronique décoratif ou une broche ; laissez libre cours à votre imagination.
Une possibilité intéressante ici est d'utiliser Home Assistant. Avec quelques-unes de ces cartes ESP8266 prêtes-à-porter intégrées à Home Assistant, il est possible de créer des effets lumineux fantaisistes (par ex. pour Noël). Mais, même si Home Assistant est optimisé pour la domotique, il peut aussi faire d'autres choses, comme animer un jeu à la fête d'anniversaire de votre enfant. Épinglez une carte sur chaque enfant et utilisez Home Assistant pour créer et commander des équipes ou décider quel candidat peut répondre à une question ou qui est qui lorsqu'on joue à chat perché. Je suis sûr qu'avec un peu de créativité, il est possible d'inventer de nombreuses applications amusantes.
Les fichiers de conception peuvent être téléchargés en [2].

(210146-04)

Produits

- **ESP-12F, module Wi-Fi basé sur l'ESP8266**
  www.elektor.fr/esp-12f-esp8266-based-wi-fi-module-160100-92
- **Carte microcontrôleur NodeMCU ESP8266**
  www.elektor.fr/nodemcu-microcontroller-board-with-esp8266-and-lua
- **H. Henrik Skovgaard,** *IoT Home Hacks with ESP8266*, **Elektor, 2020**
  www.elektor.fr/iot-home-hacks-with-esp8266

Des questions, des commentaires ?
Envoyez un courriel à l'auteur (clemens.valens@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Contributeurs
Idée, conception, texte et photographies : **Clemens Valens**
Schéma : **Patrick Wielders**
Rédaction : **Jens Nickel, C. J. Abate**
Mise en page : **Giel Dols**
Traduction : **Denis Lafourcade**

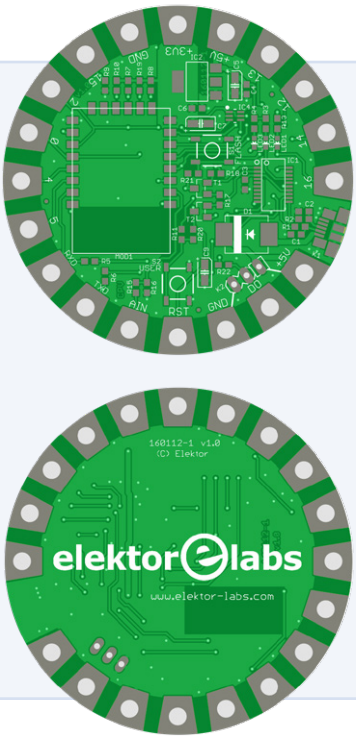
LISTE DES COMPOSANTS

Résistances
Toutes 5%, 50 V, 0,1 W, 0603
R20 = 0 Ω
R1, R2 = 27 Ω
R11, R13, R21, R22 = 470 Ω
R3, R4, R5, R6 = 1 kΩ
R7, R8, R9, R10, R17, R18, R19 = 10 kΩ
R16 = 100 kΩ
R15 = 220 kΩ

Condensateurs
C1, C2 = 47 pF, 0603
C3, C4 = 100 nF, 0603
C6 = 1 µF, 0603
C5, C7, C9 = 10 µF, 16 V, boîtier A

Semi-conducteurs
D1 = MBRS540
IC4 = 74LVC1T45GW
IC1 = FT231XS
IC2 = LD1117AS33
LED1 = LED, bleue, 0603
LED2 = LED, jaune, 0603
LED3 = LED, rouge, 0603
T1, T2 = BC847C

Divers
K1 = connecteur micro-USB de type B, montage en surface
K2 = connecteur à 3 broches, pas de 2,54 mm
S1, S2 = interrupteur tactile, 5,1 × 5,1 mm
MOD1 = ESP-12F
Circuit imprimé réf. 160112-1



LIENS

[1] **C. Valens, « la domotique, c'est facile avec... », Elektor 09-10/2020** : www.elektormagazine.fr/200019-04
[2] **Gadget Wi-Fi vestimentaire, Elektor Labs** : www.elektormagazine.fr/labs/4382