PROJET DU LABO

sonnette ESP32 par Telegram

« Le facteur sonne rarement une deuxième fois »

Luc Lemmens (Elektor Labs)

Cela nous arrive à tous : quelqu'un sonne à votre porte, mais vous n'entendez pas la sonnette ! Ou alors vous n'êtes pas à la maison. Si la personne à la porte connaît votre numéro de téléphone, elle pourra peut-être vous joindre. Et sinon, elle fait chou blanc et repart. C'est devenu particulièrement ennuyeux par exemple si vous êtes resté chez vous à attendre la livraison d'un colis.
Voici donc un projet de sonnette pas comme les autres, pour mettre fin à de tels désagréments.

Livreurs, facteurs et autres visiteurs ont rarement la patience d'attendre ou le temps de repasser. Plus tard, vous trouvez dans votre boîte aux lettres la trace de leur passage, une note indiquant éventuellement le point de collecte de votre colis ou l'annonce d'une nouvelle tentative de livraison le lendemain. C'est frustrant ! L'heure est venue de moderniser nos sonnettes.

INFOS SUR LE PROJET

ESP32
Arduino IoT
M5Stack

débutant
➡ connaisseur
expert

±3 h

Arduino IDE,
outils de labo courants

±85 €

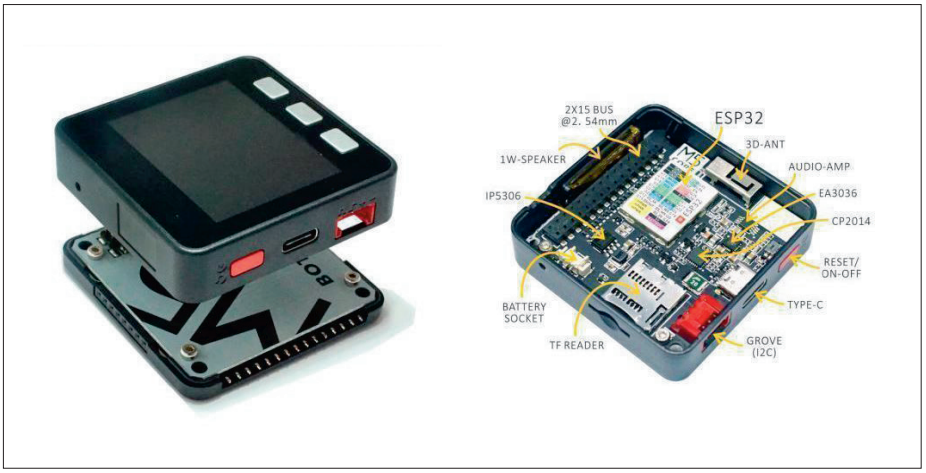


Figure 1 : Le module M5Stack ESP32 Basic Core disponible dans l'e-choppe d'Elektor.

Un télégramme pour vous !

Bientôt, dès que votre *nouvelle* sonnette retentira, ce circuit enverra un message à votre téléphone tactile où tourne une application appelée *Telegram Messenger.* Vous connaissez peut-être cette application de messagerie mobile, appréciée pour sa légèreté, sa rapidité, sa sécurité et sa vitesse. On en trouve des déclinaisons pour les trois plateformes de communication courantes : Android, Apple et Windows ainsi que des versions de bureau pour PC/Mac/Linux, MacOS... Comme *WhatsApp, Telegram* est lié à votre numéro de téléphone portable, mais vous pouvez également enregistrer votre nom d'utilisateur, ce qui vous permet de garder votre compte même si vous changez de numéro. Sur les autres différences plus ou moins subtiles entre ces deux messageries, vous trouverez sur l'internet des comparaisons, des campagnes de dénigrement et de glorification, des cohortes d'admirateurs, de détracteurs, et beaucoup d'interminables débats. Ce qui frappe du côté de *Telegram*, c'est la sobriété. Voyez leur site [**1**].

Revenons à notre nouvelle sonnette. Si votre application *Telegram* est active et connectée au serveur du même nom (par Wi-Fi ou réseau mobile), elle vous avertira de la présence d'un visiteur à votre porte, où que vous soyez, dès que cette personne sonnera. À la limite, vous recevrez le message alors que le visiteur aura peut-être encore le doigt sur la sonnette. Même pas besoin de vous précipiter vers la porte d'entrée ! Si vous savez qui est ce visiteur, vous pouvez aussitôt renvoyer un message *Telegram* qui actionnera un ouvre-porte électrique.

Telegram rencontre ESP32

Nous verrons qu'il est facile de mettre en place un traitement intelligent automatisé avec *Telegram*. Communication et commande à distance sont assurées par un ESP32, plus précisément pour envoyer un message lorsqu'on appuie sur un certain bouton et pour répondre (éventuellement) par l'activation d'un relais. Les possibilités ne se limitent pas à cette application simple. Le principe poura facilement être adapté et étendu pour répondre à d'autres besoins de commande à distance.

L'ESP32 (et son prédécesseur l'ESP8266) revient dans de nombreux projets et livres d'Elektor. Ici, nous utiliserons l'ESP32 comme module intelligent qui interagit avec le monde extérieur par Wi-Fi interposée. L'environnement de développement *Arduino* et la bibliothèque de (ro)bots *UniversalTelegram* simplifient *en principe* le développement d'applications comme cette sonnette. (pour la petite histoire énigmatique, voir le cadre ***Mauvaise rencontre*).**

Matériel : noyau M5Stack ESP32

Moyennant des modifications mineures du logiciel, ce projet pourra être utilisé avec n'importe quel module ESP32 ou 8266. Nous avons opté pour le module *M5Stack ESP32 Basic Core* de la **fig. 1,** disponible dans l'e-choppe d'Elektor (référence en fin d'article).Ce module compact réunit tout ce qu'il faut :

- affichage graphique
- boutons poussoirs
- haut-parleur
- batterie rechargeable intégrée

Les E/S du *M5Stack ESP32 Basic Core* sont accessibles de l'extérieur sur des connecteurs. Ouvrez l'arrière de la boîte et retirez une carte enfichable, vous trouverez un connecteur d'extension à 30 voies pour votre matériel.
Pour les petites extensions comme celle-ci, le matériel supplémentaire s'installe sur une carte de même taille que le module M5Stack lui-même. Il existe pour le boîtier de nombreux modèles pour l'impression 3D [**2**]. Ce qui vous fera un appareil bien fini. Avec un minimum d'efforts, vous passez du stade de prototype expérimental à celui d'un boîtier qui se laisse voir aussi en dehors de l'atelier de bricolage. En bref, ce module M5Stack offre une base excellente pour de nombreux projets ESP32, depuis les plateformes de développement de projets aux dispositifs «fermés» ou «clés en main». Pour commencer, vous pouvez utiliser le M5Stack et notre carte d'extension pour construire une installation complète de sonnette. En termes de volume, le haut-parleur incorporé peut rivaliser avec une *vraie* sonnette de porte. Un bouton de sonnette séparé peut être câblé en parallèle avec (ou à la place du) transistor de sortie de l'optocoupleur. Le signal de sonnerie doit être ajouté dans le *sketch* Arduino, et

le niveau logique sur GPIO17 doit être inversé si le bouton est connecté ainsi. Nous n'irons pas plus loin, et nous nous limiterons ici à une extension de sonnette à raccorder à une installation existante.

Schéma

Nous supposons sur le schéma de la **fig 2** que le module M5Stack est alimenté par son interface USB. Le rôle du matériel supplémentaire est donc la commande d'un ouvre-porte par T1 et RE1, ainsi que, par l'intermédiaire de l'optocoupleur IC1, la connexion au bouton de sonnette existant. Notre extension de sonnette peut donc être raccordée directement à une sonnette existante, dont les deux fils du bouton sont donc connectés à K1. Une sonnette électrique est habituellement alimentée par une tension alternative, qui n'a jamais fait l'objet d'une standardisation. On trouve de tout entre 6 V et 24 V, et on tombe parfois sur des variantes exotiques. Cette tension alternative est présente sur les fils du bouton de la sonnette tant qu'il n'est pas enfoncé, et sur K1 également. L'optocoupleur traduit la présence de cette tension en un niveau logique sur la ligne GPIO17 de l'ESP32 : bas au repos (on ne sonne pas), haut quand on sonne. Ensuite, il y a la connexion BT1, prévue en option pour une batterie au lithium de 3,7 V montée en série dans le module M5Stack, rouge au positif, noir au moins de BT1.

La carte d'extension ne comporte que des composants traversants, faciles à assembler. Attention à K2, monté d'une manière inhabituelle (**fig 3**). Normalement, c'est le côté court des broches que l'on insère dans le PCB côté composants pour les souder de l'autre côté. Ici ce sera l'inverse : les broches longues sont insérées côté cuivre (sous le PCB) pour les souder du côté des composants. Le côté court des broches est inséré dans le support du M5Stack et la partie en plastique de K2 sert également d'entretoise entre les deux circuits imprimés.

Installation de Telegram

L'étape suivante consiste à créer un compte *Telegram* sur votre téléphone, lequel commencera par télécharger l'application de l'*App Store* ou de *Google Play*. L'installation est simple, le compte lié au numéro de téléphone. Il est recommandé d'installer également l'application *Telegram* sur votre PC ou Mac, elle

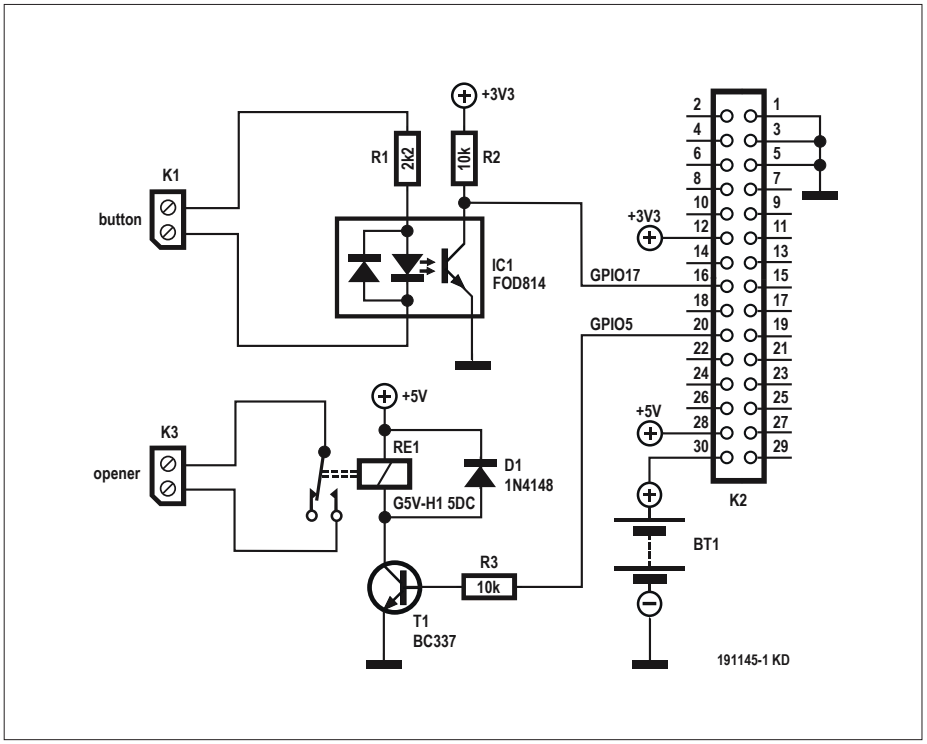


Figure 2 : Schéma de la sonnette de porte combinant les atouts d'*Arduino*, d'ESP32 et de l'application et du service de messagerie en ligne gratuit *Telegram*.

fonctionne différemment une fois que la connexion entre *Telegram* et le *sketch* ESP32 sera faite.
Connectez-vous à l'application *Telegram* sur votre ordinateur en utilisant votre numéro de téléphone pour vous identifier. Aussitôt l'application sur le téléphone affichera un code de vérification à saisir dans l'application sur l'ordinateur pour y autoriser *Telegram*. Le programme sur l'ESP32 est un robot répondeur. Il peut répondre à des commandes reçues par un message *Telegram* ou envoyer des messages toujours par *Telegram*, déclen-

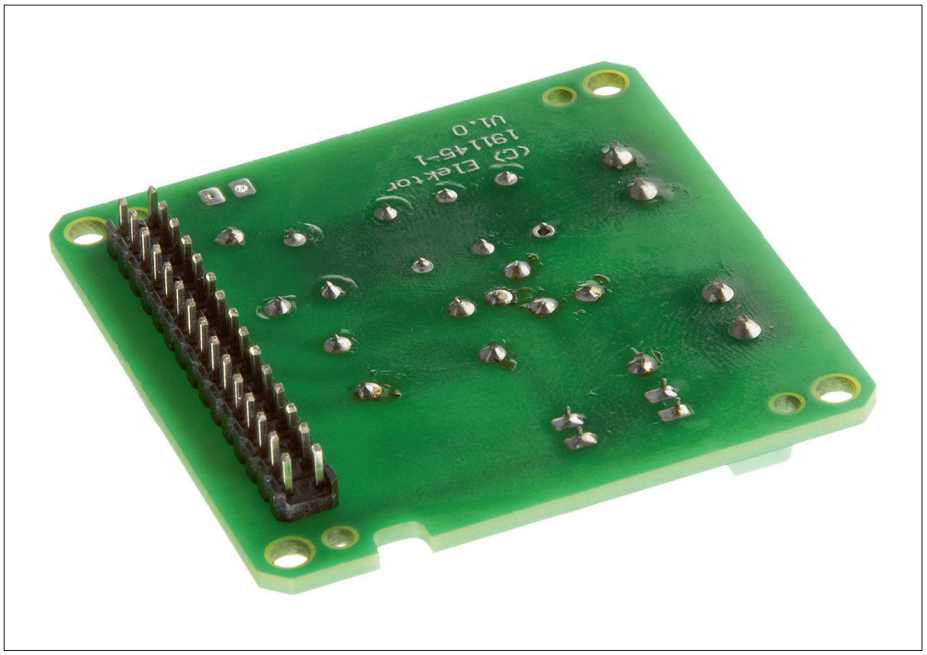


Figure 3 : Méthode recommandée pour le montage de K2 : la partie **longue** des broches est insérée par le côté cuivre (sous le PCB), pour être soudée du côté des composants.

Mauvaise rencontre (furtive)

Tout va bien, votre code source soigneusement étudié est prêt, il se compile sans erreur, il est téléchargé sur l'ESP32 sans problème, et le circuit fonctionne comme prévu. C'est bien parti, mais... vous constatez que le µC est initialisé chaque fois qu'un message arrive par Telegram ! Vous vérifiez et revérifiez. L'erreur persiste. Vous recompilez et testez le programme sur un autre ordinateur et là, à votre grande surprise, l'ESP32 fait ce qu'on attend de lui ! Les deux ordinateurs ont la même version de l'EDI Arduino et des bibliothèques. Aucune anomalie. Alors vous supprimez complètement 'Arduino' sur le premier ordinateur. Pas une simple désinstallation, qui laisse des traces dans des dossiers cachés. Vous procédez à une nouvelle installation d'Arduino et tous les problèmes ont disparu.

Quelle est donc la nature exacte de ce problème ?
Aucune idée. Ce qu'il faut en retenir c'est que l'IDE Arduino _peut_ se comporter de manière erratique. Au labo d'Elektor, je crois que c'est la première fois que nous avons rencontré un tel problème.
Espérons que ce fut aussi la dernière !

chés par des événements dans ou autour de l'ESP ; par exemple pour signaler une pression sur une touche ou envoyer une valeur de tension après conversion analogique-numérique ADC. Dans notre projet, les commandes commandent un ouvre-porte. Il faut d'abord créer ce (ro)bot dans *Telegram* en cherchant dans la liste de contacts un certain *BotFather*.

Celui-ci est lui-même un robot pour créer votre bot de sonnette en utilisant une discussion (*chat*). Après avoir saisi la commande /newbot le dialogue commence en demandant un nom (qui apparaîtra plus tard dans la liste des contacts *Telegram*) et un nom d'utilisateur pour le (ro)bot. Ce dernier doit toujours se terminer par les caractères «bot». Notez la barre oblique (/) au début de chaque commande du bot. Lorsque ce processus est réussi, le *BotFather* vous en informe et affiche un code (*token*) nécessaire pour autoriser le nouveau (ro)bot à envoyer des requêtes à l'API.

Gardez ce jeton en lieu sûr, il pourrait permettre à n'importe qui de prendre le contrôle de votre (ro)bot de sonnette. C'est pourquoi nous l'avons flouté dans le dialogue de configuration de notre propre robot (**fig 4**).

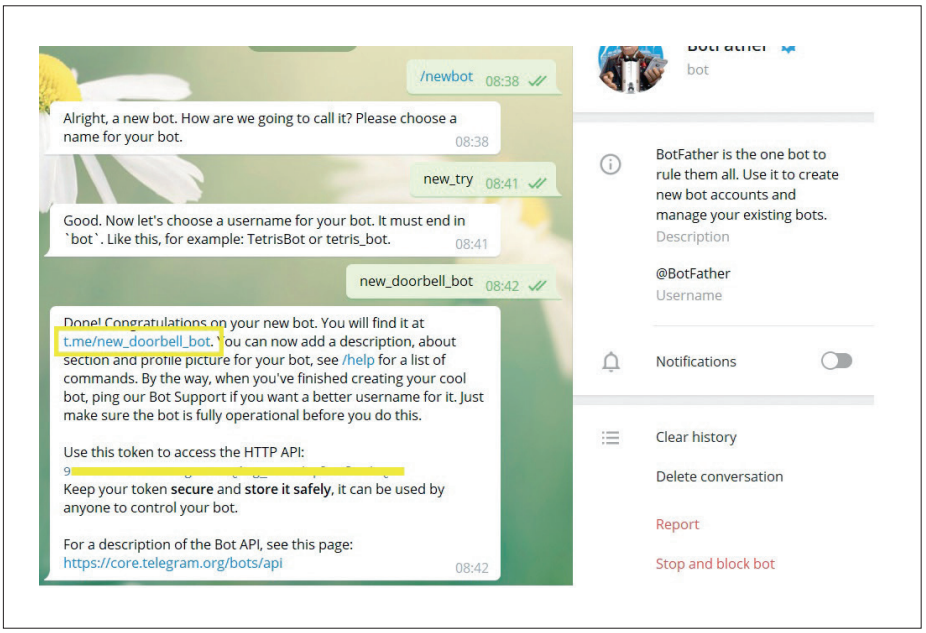


Figure 4 : Cette copie d'écran montre comment nous nous sommes débrouillés au labo pour l'utilisation de *BotFather*. Les jetons ont été cachés pour brouiller l'écoute.

Dans ce dialogue avec *BotFather*, cliquez sur le lien encadré ('t.me/...') pour ouvrir une fenêtre de discussion pour le nouveau robot, également ajouté à la liste de contacts. Ce (ro)bot est maintenant prêt à l'emploi, il ne reste qu'à le relier au *sketch* Arduino pour *le prolongateur* de sonnette.

Sketch Arduino

Dans le téléchargement du logiciel de ce projet, vous trouverez le *sketch* `DOORBELL.INO`, c'est-à-dire le code source du micrologiciel ESP32 [**3**]. Ce programme peut être retouché et, dans sa forme actuelle, il est destiné à montrer surtout comment Arduino/ESP32 et *Telegram* travaillent ensemble pour simplifier les systèmes de commande.

L'utilisation d'un ouvre-porte électrique est une brèche potentielle, car un intrus pourrait pirater le (ro)bot et ouvrir votre porte avec son téléphone (ou son ordinateur). Elektor décline donc toute responsabilité en cas de dommage résultant de l'application de ce projet !

Dans le *sketch*, notre module M5Stack nécessite une connexion au réseau local Wi-Fi, et l'accès au bot *Telegram* que vous venez de créer. Les identifiants de votre réseau Wi-Fi sont dans les lignes 17 et 18, tandis que le jeton attribué par *BotFather* est à la ligne 21. Vous comprenez pourquoi on vous conseillait de créer le bot sur votre ordinateur, afin de copier-coller facilement le jeton dans le *sketch*. Une fois l'installation de l'EDI Arduino terminée avec succès, l'esquisse se compilera sans aucun problème. Dans le menu Outils — lorsque le M5Stack est connecté à un port USB de l'ordinateur — la carte correcte (M5Stack-Core-ESP32) et le port COM virtuel doivent être réglés afin de télécharger le *sketch* sur l'ESP32.

Cela fait, le *sketch* se lance, se connecte au réseau, et le nom du réseau Wi-Fi et l'adresse IP du prolongateur de sonnette apparaissent sur le LCD du M5Stack. Le (ro)bot doit être activé après chaque redémarrage en envoyant la commande `/start` via la fenêtre de discussion dans *Telegram*. Le (ro)bot fournit alors un retour d'information avec vue d'ensemble des commandes reconnues.
Lorsque la sonnette retentit, un message est envoyé par l'application *Telegram* à votre ordinateur ou votre téléphone.

LISTE DES COMPOSANTS

Résistances

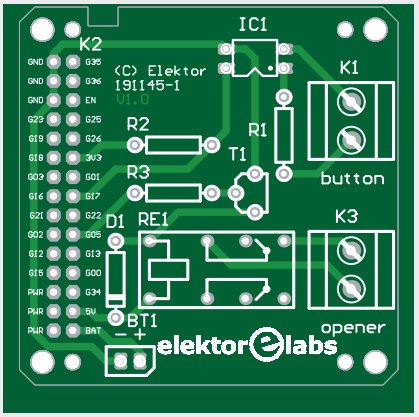
R1 = 2,2 kΩ 5%, 0,25 W, 250 V
R2,R3 = 10 kΩ 5%, 0,25 W, 250 V

Semi-conducteurs

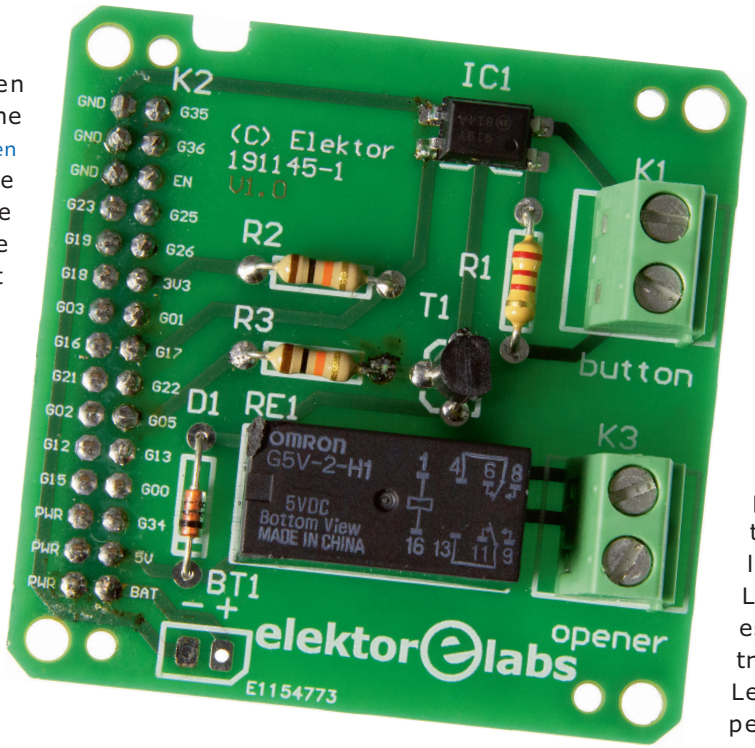
D1 = 1N4148
T1 = BC337
IC1 = FOD814A, optocoupleur AC avec sortie à transistor, 1 canal, DIP, 4 broches, 50 mA, 5 kV

Divers

RE1 = G5V-2-H1 5DC, bobine 5 V_{CC}, DPDT, 1 A (Omron)
K1,K3 = bornier à vis pour circuit imprimé à 2 voies (pas de 5,08 mm), 630 V
K2 = connecteur à 30 broches (2x15), snap-off, vertical
module M5Stack ESP32 Basic Core, dans l'e-choppe d'Elektor (ou module ESP32 équivalent)
BT1 = batterie en option avec support
circuit imprimé 191145-1 V1.0 dans l'e-choppe d'Elektor



Vous pouvez répondre en ouvrant la porte vous-même ou en utilisant le bouton `/Open` pour actionner un ouvre-porte électrique (s'il y en a un). Le bouton `/Ignore` permet de renoncer à cette option, et le bouton `/status` (occupé à vérifier la connexion au prolongateur de sonnette) apparaît à nouveau en bas de l'écran de discussion. Ces commandes peuvent également être tapées et envoyées via la fenêtre de discussion, mais attention, elles sont sensibles à la casse ! C'est la bibliothèque de (ro)bots `UniversalTelegram` [4], qui fait la plus grande partie du boulot dans ce projet. Grâce à sa documentation et aux exemples fournis, ce n'est pas bien sorcier. Notre *sketch* de sonnette s'appuie sur l'exemple du «clavier personnalisé», qui montre comment envoyer des boutons de commande à *Telegram*, une option pratique, qui évite d'avoir à taper les commandes dans la fenêtre de discussion. La routine indispensable est `handleNewMessages` qui traite les messages entrants. Les quatre instructions `If` permettent au (ro)bot de comprendre :



- start
- status
- Ignore
- Open

Pour pouvoir envoyer des messages, le *sketch* a besoin de l'ID de discussion enregistré dans la variable `ThisChat` après avoir reçu la première commande `/start`. Le bouton de commande (c'est-à-dire la sonnette) est capturé au cours d'une interruption ; notez l'attribut `IRAM_ATTR` requis pour la déclaration d'une routine de service d'interruption ESP, tout cela pour un traitement correct bien sûr.

(191145-03 VF)

@ WWW.ELEKTOR.FR

→ **M5Stack ESP32 Basic Core**
www.elektor.fr/m5stack-esp32-basic-core-development-kit

→ **Circuit imprimé n° 191145-1 V1.0**
www.elektor.fr/191145-1

Liens

[1] Telegram Messenger : https://telegram.org/
[2] modèles 3D pour le boîtier M5Stack : https://github.com/m4k3r-net/M5Stack-3DPrintFiles
[3] sketch Arduino : www.elektormagazine.com/191145-01
[4] bibliothèque de (ro)bots UniversalTelegram : https://Github.com/witnessmenow/Universal-Arduino-Telegram-Bot