

Bienvenue sur la nouvelle version de Connect... Professionnels, découvrez toutes nos offres dédiées sur notre boutique (<https://proboutique.ed-diamond.com/>) !



Déconnexion (</user/logout>)

## HACKABLE

Électronique numérique & embarqué



336 article(s)

Accueil (</>) > Hackable (</Hackable>) > HK-016 (</Hackable/HK-016>)  
> Contrôlez vos montages Bluetooth depuis votre Pi



([/Hackable/HK-](/Hackable/HK-016)

[016/Utiliser-](#)

[vos-](#)

[applications-](#)

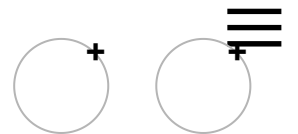
[graphiques-](#)

[Raspberry-](#)

[Pi-](#)

[depuis-](#)

[Windows](#)   



(</readlist>) (</get/a>)

# Contrôlez vos montages Bluetooth depuis votre Pi

Hackable n° 16 (</Hackable/HK-016>) janvier 2017 Par Bodor Denis (</auteur/Bodor-Denis>)

Embarqué (</search/node?domains%5B0%5D=72457>)

Radio et wireless (</search/node?domains%5B0%5D=72465>)

*Dans le précédent numéro, nous avons décrit comment ajouter une connectivité Bluetooth à un projet Arduino et ainsi pouvoir lui communiquer des ordres depuis un smartphone Android. Mais il peut également être très intéressant de remplacer le smartphone par une carte Raspberry Pi et ainsi gagner en souplesse et surtout ne plus être dépendant d'une application figée, peu configurable, difficile, voire impossible à modifier.*

Comme je l'ai dit dans le précédent numéro, s'attacher à la création d'une application pour smartphone, qu'il s'agisse d'Android ou iOS, n'est pas une tâche aisée. Il y a beaucoup à apprendre et cela demande un certain temps avant d'acquérir une confiance suffisante passé la première étape de l'application « Bonjour monde ».

Si l'on souhaite contrôler ou communiquer avec un montage à base d'Arduino et d'un module Bluetooth, il est bien plus simple de le faire avec un système plus facile à prendre en main : typiquement, une Raspberry Pi et quelques lignes de Python. Mais pour cela, encore faut-il connaître et savoir utiliser les bonnes commandes permettant de contrôler le Bluetooth sur une Pi.



([https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT\\_pi3.jpg](https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT_pi3.jpg))

*La carte Raspberry Pi 3 intègre de base un support pour le Wifi et le Bluetooth (Classic et Low Energy). Aucun accessoire complémentaire ne sera alors nécessaire, si ce n'est l'indispensable radiateur permettant d'éviter des plantages en cas de très forte charge du système. Le module enfiché à gauche du logo Pi est une horloge permettant de conserver l'heure entre deux redémarrages (module RTC).*

La Raspberry Pi 3 intègre le support du Wifi et du Bluetooth, *Classic* comme *Low Energy* (BLE), mais si vous ne disposez que d'un précédent modèle, l'ajout d'une simple clé USB règlera vite le problème et vous apportera les fonctionnalités souhaitées. Un modèle minuscule souvent recommandé est fabriqué par Plugable Technologies sous le nom USB-BT4LE. Celui-ci vous coûtera quelques 15€, se trouve facilement sur Amazon par exemple et vous fournira les mêmes fonctionnalités que celles disponibles sur une Pi 3 (à l'exception du Wifi bien sûr).

Les explications qui vont suivre sont relativement génériques et s'adapteront à n'importe quel montage. Le croquis Arduino correspondant au montage que nous cherchons à contrôler est disponible dans le dépôt GitHub du précédent numéro à l'adresse <https://github.com/Hackable-magazine/Hackable15>.

## 1. Prendre en main le Bluetooth et vérifier que tout marche

Il est possible de configurer et gérer le Bluetooth sur la Raspberry Pi en utilisant l'environnement graphique, mais ceci reste très limité et très orienté vers l'utilisation de périphériques comme un clavier ou une souris Bluetooth. Ce qui nous occupe ici est sensiblement différent puisqu'il s'agit non pas de faire en sorte que le système utilise le périphérique, mais de contrôler directement un montage « maison » communiquant en Bluetooth.

De plus, en cas de problème, il est important de connaître la mécanique qui se trouve sous le capot. Notre voie sera donc la ligne de commandes, offrant un accès complet à la configuration.



([https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT\\_plugable.jpg](https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT_plugable.jpg))

*L'adaptateur généralement recommandé et capable de fournir du Bluetooth à une Pi (ou tout autre système) occupera une place très réduite et ne vous*

*coûtera qu'une quinzaine d'euros. Une simple recherche de « Bluetooth Plugable » sur Amazon et vous trouverez votre bonheur...*

La première manipulation que nous allons effectuer consistera à nous assurer que le Bluetooth, aussi bien l'aspect matériel que logiciel, est effectivement bien pris en charge. Pour ce faire, utilisez simplement la commande **hciconfig**, qui devrait alors vous retourner quelque chose comme :

```
hci0: Type: BR/EDR Bus: UART
BD Address: B8:27:EB:10:1F:A8 ACL MTU: 1021:8 SCO MTU: 64:1
UP RUNNING
RX bytes:108313 acl:0 sco:0 events:4865 errors:0
TX bytes:28350 acl:0 sco:0 commands:3343 errors:0
```

Notre **adaptateur Bluetooth** est désigné dans le système par **hci0**. Il possède comme **adresse matérielle** « **B8:27:EB:10:1F:A8** », est actif et en marche (**UP RUNNING**). Cette commande provient du paquet **bluez**, mais il en existe d'autres, plus intéressantes fournies par le paquet **bluez-tools**. BlueZ est le nom du projet ayant en charge de fournir les fonctionnalités Bluetooth à GNU/Linux, que ce soit sous la forme de pilotes, de services ou d'outils, et ce aussi bien pour la Raspberry Pi que pour les distributions GNU/Linux sur PC.

**bluez-tools** nous fournit par exemple la commande **bt-adapter** nous permettant d'obtenir davantage d'informations :

```
$ bt-adapter -i
[hci0]
Name: raspberrypi
Address: B8:27:EB:10:1F:A8
Alias: raspberrypi [rw]
Class: 0x0
Discoverable: 0 [rw]
DiscoverableTimeout: 180 [rw]
Discovering: 0
Pairable: 1 [rw]
PairableTimeout: 0 [rw]
Powered: 1 [rw]
UUIDs: [PnPInformation, 00001800-0000-1000-8000-00805f9b34fb,
00001801-0000-1000-8000-00805f9b34fb, AVRemoteControl,
AVRemoteControlTarget]
```

On retrouve ici non seulement l'adresse matérielle, mais également le nom affiché du périphérique. Les produits disposant d'une connectivité Bluetooth sont identifiés par leur adresse qui est unique et par leur nom, qui peut être tantôt changé et qui, dans GNU/Linux, correspond par défaut au nom du système (nom d'hôte). Ici, ce nom pourra être modifié par l'édition du fichier **/etc/bluetooth/main.conf** et un redémarrage du service avec **sudo service bluetooth restart**.

On voit également dans le résultat de la commande que notre adaptateur n'est pas visible de l'extérieur (**Discoverable: 0**), n'est pas en train de chercher des périphériques (**Discovering: 0**), mais est appairable (ou associable). La notion d'appairage (*pairing* en anglais) est fondamentale en Bluetooth, car il s'agit d'une sécurité permettant de limiter les accès aux périphériques tout en n'ayant pas besoin, à chaque fois, de montrer patte blanche. Une fois deux périphériques appairés, généralement après vérification d'un code ou d'un mot de passe, ils peuvent communiquer librement. Les périphériques Bluetooth Smart ou Bluetooth Low Energy fonctionnent de façon un peu différente à ce niveau selon leur type, mais nous resterons concentrés ici sur le Bluetooth Classic.

Il est possible de manipuler notre adaptateur Bluetooth avec la commande **bt-adapter** et en utilisant l'option **--set** pour changer sa configuration. De la même manière, le paquet **bluez-tools** offre également la commande **bt-device** permettant de gérer les périphériques (détection, association, collecte d'informations, etc.). Je trouve cependant plus facile d'utiliser un autre outil, fourni par le paquet **bluez** : **bluetoothctl**.

## 2. Jouer avec les périphériques et les appairer

**bluetoothctl** est un programme interactif. Une fois lancé, vous obtenez une nouvelle ligne de commandes vous permettant de saisir des instructions propres à la gestion du Bluetooth. Vous pourrez quitter cette interface en utilisant simplement la commande **exit** ou avec le raccourci Ctrl+d. Nous lancerons la commande ainsi :

```
$ bluetoothctl -a
[NEW] Controller B8:27:EB:10:1F:A8 raspberrypi [default]
Agent registered
[bluetooth] #
```

Notez que celle-ci n'a pas besoin d'être lancée avec **sudo**, l'utilisateur par défaut ayant déjà les droits adéquats et la commande ne contrôlant pas directement le matériel, mais plutôt le service Bluetooth associé. L'option **-a** nous permet de faire en sorte de déclarer automatiquement un agent. En effet, le fait que le Bluetooth s'utilise également en mode graphique, directement sur le bureau, nécessite un mécanisme particulier.

Lors d'une association par exemple, il est nécessaire que l'utilisateur saisisse un code PIN ou un mot de passe. Or ceci ne peut pas être proposé par un service travaillant en tâche de fond dans le système. Il est alors fait appel à un agent, présentant une fenêtre et attendant la saisie dont le contenu sera ensuite communiqué au service qui procèdera à l'association Bluetooth avec le périphérique.

Ici, via l'option **-a**, nous spécifions que nous n'avons pas besoin d'un agent externe, mais que la commande **bluetoothctl** elle-même servira d'interface et d'agent en même temps. Ainsi, lorsque nous demanderons une association nous pourrions saisir le code ou le mot de passe correspondant dans la foulée.



([https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT\\_adaptateurs.jpg](https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT_adaptateurs.jpg))

*Il existe des adaptateurs Bluetooth USB de toutes sortes. Cette petite collection présente en bas à droite un modèle récent intégrant Bluetooth Classic et Bluetooth Low Energy (BLE), à gauche un vieux modèle d'entrée de gamme et en rouge, un périphérique industriel de très bonne facture, mais ne supportant malheureusement pas le BLE.*

Une fois la commande **bluetoothctl** lancée, nous pouvons afficher des informations sur notre adaptateur avec **list** et **show**. Ceci nous présente peu ou prou la même chose que les commandes précédentes. Si plus d'un adaptateur est présent sur le système, il est également possible de choisir celui à utiliser avec un **list** puis une commande **select** suivie de l'adresse matérielle de l'adaptateur.

La commande **devices** nous permet de lister les périphériques disponibles, mais s'il s'agit de votre première utilisation du Bluetooth avec votre Raspberry Pi, celle-ci ne retournera probablement rien du tout. Il nous faut en effet tout d'abord provoquer une recherche des périphériques Bluetooth qui nous entourent et donc demander à l'adaptateur de passer en mode découverte (*discovering* à *yes*) avec la commande **scan on**.

Si cette commande vous retourne un message « *Failed to start discovery: org.bluez.Error.NotReady* », assurez-vous via un **show** que l'adaptateur est bien en marche (« **Powered: yes** »). Si ce n'est pas le cas, pliez-vous simplement d'un **power on** :

```
[bluetooth]# power on
Changing power on succeeded
[CHG] Controller B8:27:EB:10:1F:A8 Powered: yes
```

avant d'utiliser **scan on** :

```
[bluetooth]# scan on
Discovery started
[CHG] Controller B8:27:EB:10:1F:A8 Discovering: yes
```

Au bout de quelques instants, le ou les périphériques qui peuvent être recherchés (*discoverable*) dans votre voisinage vont commencer à être listés :

```
[NEW] Device 00:19:5D:EE:A4:24 Hackable15
[NEW] Device 78:CA:39:BF:E4:DA iMac de yann
[NEW] Device 00:19:5D:EE:A4:1D HackableBT2
[NEW] Device 49:83:39:88:BE:21 49-83-39-88-BE-21
[CHG] Device 49:83:39:88:BE:21 RSSI: -91
[NEW] Device 78:E9:94:D7:DC:41 78-E9-94-D7-DC-41
[CHG] Device 78:E9:94:D7:DC:41 RSSI: -84
[CHG] Device 78:E9:94:D7:DC:41 RSSI: -95
[CHG] Device 78:E9:94:D7:DC:41 RSSI: -86
[CHG] Device 78:CA:39:BF:E4:DA RSSI: -82
[NEW] Device 43:07:CA:F8:FB:A8 43-07-CA-F8-FB-A8
[CHG] Device 43:07:CA:F8:FB:A8 RSSI: -99
[CHG] Device 43:07:CA:F8:FB:A8 RSSI: -87
[CHG] Device 43:07:CA:F8:FB:A8 RSSI: -97
[NEW] Device 79:58:75:F5:00:07 79-58-75-F5-00-07
```

Notez que la commande **scan** change l'état de l'adaptateur qui se met à la recherche de périphériques, mais que celle-ci n'est pas bloquante. Les messages apparaissent au fur et à mesure alors que vous gardez la main sur l'outil. Les messages concernent aussi bien les découvertes de périphériques que les changements les concernant, comme la variation de la puissance du signal (RSSI) ou encore le fait d'obtenir le nom associé à l'adresse matérielle. Gardez simplement à l'esprit que ces messages ne sont pas le résultat de votre commande, mais juste une information sur ce qui se passe.

Après quelque temps, vous pourrez utiliser la commande **devices** pour lister vos découvertes alentour :

```
[bluetooth]# devices
Device 00:19:5D:EE:A4:24 Hackable15
Device 78:CA:39:BF:E4:DA iMac de yann
Device 00:19:5D:EE:A4:1D HackableBT2
Device 49:83:39:88:BE:21 49-83-39-88-BE-21
Device 78:E9:94:D7:DC:41 78-E9-94-D7-DC-41
```

Nous voyons ici que plusieurs périphériques ont été trouvés. Certains donnent leur nom d'autres pas. Certains sont à moi et d'autres à quelqu'un dans le voisinage. Celui qui nous intéresse ici est « Hackable15 », c'est le module Bluetooth utilisé dans le précédent numéro en compagnie d'une carte Arduino permettant de contrôler la couleur d'une led depuis un smartphone Android.

Nous pouvons obtenir des informations sur ce périphérique :

```
[bluetooth]# info 00:19:5D:EE:A4:24
Device 00:19:5D:EE:A4:24
Name: Hackable15
Alias: Hackable15
Class: 0x0c0800
Paired: no
```

```
Trusted: no
Blocked: no
Connected: no
LegacyPairing: yes
```

Nous voyons ici son nom et adresse, mais également le fait qu'il ne soit pas associé, ni de confiance, ni bloqué, ni connecté. Nous apprenons également que ce périphérique supporte le *legacy pairing*, correspondant à une méthode d'association propre au Bluetooth 2.0 (normal puisqu'il ne s'agit pas d'un périphérique Bluetooth Low Energy). Nous pouvons éventuellement tenter une connexion, qui est vouée à l'échec par définition sans association, afin d'obtenir tantôt plus d'informations et en particulier le ou les UUID permettant à l'outil de connaître les services disponibles et donc quelques caractéristiques supplémentaires. Exemple :

```
[
bluetooth]# connect 00:19:5D:EE:A4:1D
Attempting to connect to 00:19:5D:EE:A4:1D
[CHG] Device 00:19:5D:EE:A4:1D Connected: yes
[CHG] Device 00:19:5D:EE:A4:1D UUIDs:
00001101-0000-1000-8000-00805f9b34fb
Failed to connect: org.bluez.Error.NotAvailable
[CHG] Device 00:19:5D:EE:A4:1D Connected: no
[bluetooth]# info 00:19:5D:EE:A4:1D
Device 00:19:5D:EE:A4:1D
Name: Barre72
Alias: Barre72
Class: 0x0c0800
Paired: no
Trusted: no
Blocked: no
Connected: no
LegacyPairing: yes
UUID: Serial Port (00001101-0000-1000-8000-00805f9b34fb)
```

Cette dernière ligne nous indique que l'UUID rapporté par le périphérique décrit un port série (*Serial Port*). Parfois, en particulier avec des PC ou des Mac ayant la connectivité Bluetooth activée, il n'est pas même nécessaire de tenter une connexion et nous obtenons une liste bien plus complète de services :

```
UUID: Audio Source (0000110a-0000-1000-8000-00805f9b34fb)
UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)
UUID: Handsfree Audio Gateway (0000111f-0000-1000-8000-00805f9b34fb)
UUID: Serial Port (00001101-0000-1000-8000-00805f9b34fb)
UUID: Service Discovery Serve.. (00001000-0000-1000-8000-00805f9b34fb)
UUID: Headset AG (00001112-0000-1000-8000-00805f9b34fb)
UUID: GN (00001117-0000-1000-8000-00805f9b34fb)
```



Mais revenons à notre périphérique « Hackable15 »... La phase suivante consiste à associer ou appairer le périphérique en utilisant la commande **pair** suivie de l'adresse matérielle correspondante :

```
[bluetooth]# pair 00:19:5D:EE:A4:24
Attempting to pair with 00:19:5D:EE:A4:24
[CHG] Device 00:19:5D:EE:A4:24 Connected: yes
Request PIN code
[agent] Enter PIN code: 424242
[CHG] Device 00:19:5D:EE:A4:24 UUIDs:
00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 00:19:5D:EE:A4:24 Paired: yes
Pairing successful
[CHG] Device 00:19:5D:EE:A4:24 Connected: no
```

Ce mélange de messages et de résultats doit être vu d'une part en :

```
[CHG] Device 00:19:5D:EE:A4:24 Connected: yes
[CHG] Device 00:19:5D:EE:A4:24 UUIDs:
00001101-0000-1000-8000-00805f9b34fb
[CHG] Device 00:19:5D:EE:A4:24 Paired: yes
[CHG] Device 00:19:5D:EE:A4:24 Connected: no
```

et d'autre part en :

```
[bluetooth]# pair 00:19:5D:EE:A4:24
Attempting to pair with 00:19:5D:EE:A4:24

Request PIN code
[agent] Enter PIN code: 424242

Pairing successful
```

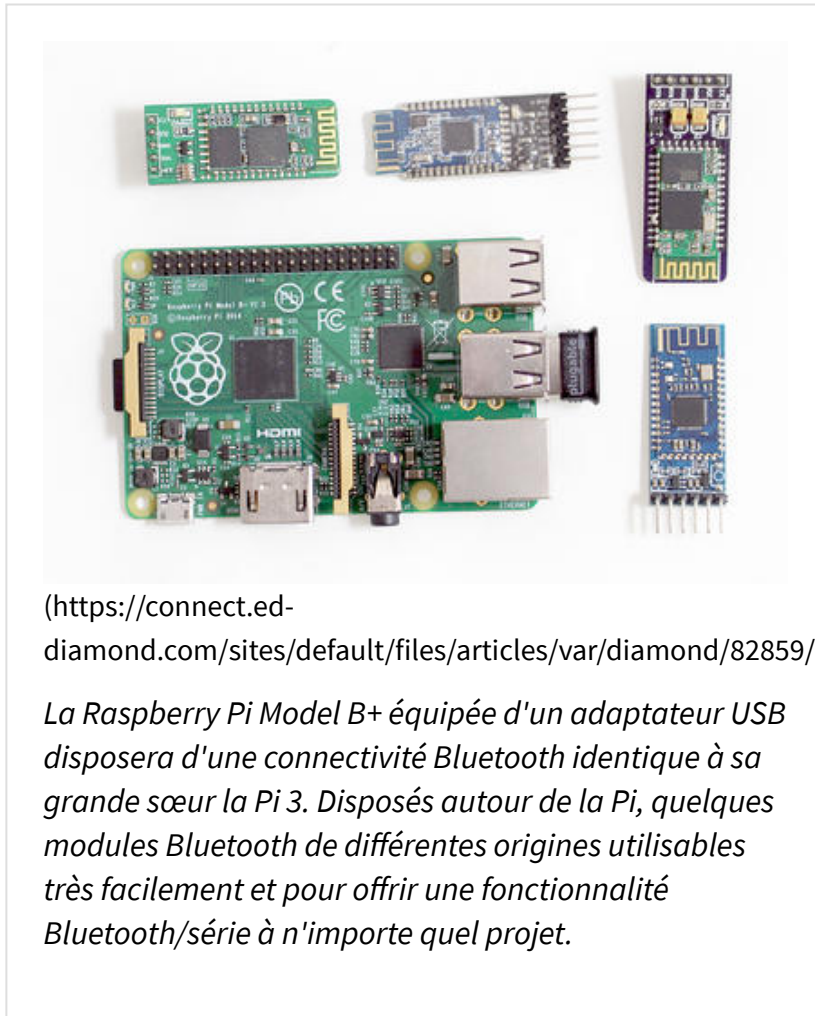
Suite à la commande, l'outil a besoin du code permettant l'association. Celui-ci nous est alors demandé, puisque **bluetoothctl** lancé avec l'option **-a** embarque un agent. Nous entrons **424242** correspondant au code que nous avons configuré au moment de paramétrer le module dans le numéro précédent. Suite à cela, la tentative d'association a lieu et réussit avec le message « Pairing successful ».

Notre périphérique est maintenant associé, ce qui signifie qu'un lien de confiance existe entre notre système et lui, et qu'il n'est plus nécessaire de procéder à d'autres authentifications via un code. Notre périphérique apparaît maintenant aussi bien dans le résultat de la commande **devices** que dans celui de la commande permettant de lister les périphériques associés :

```
[bluetooth]# paired-devices
Device 00:19:5D:EE:A4:24 Hackable15
```

Bien entendu l'association, une fois en place, perdure après avoir quitté **bluetoothctl** et même entre deux redémarrages. Attention cependant, celle-ci est liée à l'adaptateur et sa propre adresse matérielle. En d'autres termes, si vous utilisez un adaptateur USB et le remplacez par un autre (d'un

modèle strictement similaire ou non), les associations ne seront plus en place et il faudra recommencer.



Pour « désassocier » un périphérique, le cas échéant, il vous suffira d'utiliser la commande **remove** suivie de l'adresse du périphérique. Notez à ce propos que pour réassocier ce même périphérique, il faudra au préalable qu'il soit à nouveau découvert (**scan on**) et présent dans la liste des périphériques (**devices**) dont les entrées expirent quelque temps après la découverte si la recherche est désactivée (**scan off**).

### 3. Communiquer en Bluetooth avec Python

À présent que notre montage Arduino est associé, nous pouvons communiquer avec lui. Il existe plusieurs techniques comme par exemple le fait d'utiliser la commande **rfcomm** du paquet **bluez** pour initier la connexion et ajouter un pseudo port série la représentant. On doit alors ensuite utiliser un outil de communication série ou un programme « maison » pour utiliser ce port et échanger des données avec le périphérique Bluetooth distant.

Une solution plus simple consiste à écrire un script Python se chargeant d'initier la connexion puis de communiquer avec notre montage. Pour cela, il faut disposer du module Python adéquat : **python-bluez**. Une fois installé avec **sudo apt-get install**, nous pouvons faire un premier essai :

```
#!/usr/bin/python
```

```

# -*- coding: utf-8 -*-

import bluetooth
import time
import sys

# adresse du périphérique Bluetooth
bd_addr = "00:19:5D:EE:A4:24"
port=1

# tentative de connexion
print "Connexion..."
try:
sock=bluetooth.BluetoothSocket(bluetooth.RFCOMM)
sock.connect((bd_addr, port))
except:
print "Erreur lors de la connexion !"
sys.exit(1)

# Ca marche, sock représente la connexion désormais
print "Connecté."
time.sleep(2)

# Tentative de déconnexion
print "Déconnexion..."try:
sock.close()
except:
print "Erreur lors de la déconnexion !"

print "Déconnecté."

```

Ce script ne fait pas grand-chose, mais nous permet de vérifier le fonctionnement de la connexion. On utilise le module Python **bluetooth** en compagnie de l'adresse matérielle du périphérique auquel se connecter (et qui est désormais appairé). On crée alors un objet **sock** retourné par **BluetoothSocket()**. La syntaxe **try/except** nous permet de détecter un problème et d'y réagir facilement.

La liaison effectuée est une liaison série reposant sur le protocole Bluetooth RFCOMM, typique des modules qu'on peut utiliser avec des montages Arduino. Une fois la communication établie, notre script marque une pause de deux secondes puis invoque la méthode **close** pour mettre un terme à la liaison. Là encore, le **try/except** est bien pratique.

Le script une fois lancé nous affiche normalement simplement :

```

Connexion...
Connecté.
Déconnexion...
Déconnecté.

```

Ce script forme un squelette qui peut nous servir de base pour tous nos projets. Dans le cas qui nous intéresse ici, nous voulons envoyer un message au montage permettant de choisir la couleur que doit prendre la led qui s'y trouve connectée. Ce message prend la forme de quelques octets : 0, 0, la valeur de rouge, de vert, de bleu et le caractère « ; ». Pour simplifier l'utilisation, notre outil va prendre en argument une valeur de teinte entre 0 et 359, puis transformer celle-ci en rouge, vert et bleu, avec une saturation et une valeur maximum.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import bluetooth
import sys
import colorsys

def hsv2rgb(h,s,v):
    return tuple(int(i * 255) for i in colorsys.hsv_to_rgb(h,s,v))

bd_addr = "00:19:5D:EE:A4:24"
port=1
msg = bytearray("\x00\x00\x00\x00\x00;")

if len(sys.argv) != 2:
    print "Donnez-moi une teinte (0-359) !"
    sys.exit(1)

couleur = int(sys.argv[1])

if not (0 <= couleur <= 359):
    print "Mauvaise couleur/teinte !"
    print "Entre 0 et 359 svp."
    sys.exit(1)

print "Envoi..."

try:
    sock=bluetooth.BluetoothSocket(bluetooth.RFCOMM)
    sock.connect((bd_addr, port))
except:
    print "Erreur lors de la connexion !"
    sys.exit(1)

msg[2],msg[3],msg[4] = hsv2rgb(couleur/360.0,1,1)
sock.send(str(msg))

try:
    sock.close()
except:
```

```
print "Erreur lors de la déconnexion !"
```

Pour la conversion de couleur, on utilise le module **colorsys** qui est livré en standard avec Python et ajoutons une fonction simple, **hsv2rgb**, qui nous retourne les valeurs au bon format (entre 0 et 255 et non entre 0,0 et 1,0).

Nous préparons notre message sur la forme d'un **bytearray**, un tableau d'octets puis, après avoir vérifié que la valeur passée en argument correspond bien à nos attentes, nous le modifions avec les valeurs de couleur calculées. Le tout est enfin envoyé avec la méthode **send()** de notre objet **sock** sous la forme d'une chaîne de caractères à destination du montage Arduino.

Pour utiliser ce script et envoyer une couleur au montage, il nous suffit maintenant de l'appeler en lui passant un nombre entre 0 et 359. Immédiatement, la led connectée prend la couleur demandée. Mission accomplie !

## 4. Pour finir... ou presque

Étant d'un naturel à préférer la ligne de commandes aux interfaces graphiques, j'ai une tendance à tout naturellement écrire des scripts et des programmes pour ce type d'interface. Python permet cependant de développer rapidement des outils graphiques (Qt, Tk, Gtk+, etc.) qui peuvent offrir plus de souplesse avec, par exemple ici, un sélecteur de couleur similaire à l'application Android que nous avons vu dans le précédent numéro.

Mais hors du cadre de la liaison avec notre montage, une telle communication Bluetooth entre Arduino et Raspberry Pi pourra satisfaire bien des besoins. On imagine ainsi aisément quelques exemples :

- sonde de température ou autre ;
- afficheur LCD distant ;
- imprimante thermique déportée ;
- télécommande Bluetooth de moteurs ou servo ;
- contrôle de prises de courant ou de luminaires pour une solution domotique ;
- etc

Il est également possible de voir en la Raspberry Pi non pas un maître, mais un esclave, attendant des connexions de périphériques. Ceci demande un peu plus de configuration au niveau système et nous reviendrons peut-être sur le sujet dans un prochain numéro. Dans ce genre de scénario, il devient alors possible d'obtenir une ligne de commandes de la part de la Pi depuis un smartphone via une application faisant office d'émulateur de terminal.



([https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT\\_SD1000.jpg](https://connect.ed-diamond.com/sites/default/files/articles/var/diamond/82859/BT_SD1000.jpg))

*Les adaptateurs USB Bluetooth industriels se déclinent tantôt en des modèles assez originaux. Celui-ci, le Parani SD1000U de SENA, n'est pas à proprement parler un périphérique Bluetooth, mais apparaît comme un port série fournissant une liaison transparente sans avoir à utiliser de pilote Bluetooth spécifique.*

Le Bluetooth est un terrain de jeu très intéressant qui offre une vaste compatibilité. Le Wifi aussi, me direz-vous, mais à la différence du Wifi, le Bluetooth fonctionne avec des équipements plus anciens ou peu puissants, ne pouvant supporter le Wifi. De plus, je trouve que le Bluetooth, du fait de prendre ici la forme d'une liaison série, est bien plus simple à mettre en œuvre. Mais à chaque projet son cahier des charges et ses besoins. La clé du succès c'est avant tout d'utiliser les bons outils pour les bonnes tâches...

[!\[\]\(d3fb9f94af8b26d1c844efa9a98805b0\_img.jpg\)](#) [!\[\]\(78eb1652b591ce460bbb1a853a52e223\_img.jpg\)](#) [!\[\]\(73569cd2fc0daa66f7f79a4aa32515bb\_img.jpg\)](#)

## Article rédigé par



(/auteur/Bodor-Denis)

### **Bodor Denis (/auteur/Bodor-Denis)**

Chef des rédactions, Rédacteur en chef du magazine Hackable - Éditions Diamond (/auteur/Bodor-Denis)

619 articles

# Par le(s) même(s) auteur(s)

Édito (/Hackable/HK-035/Edito)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

« *Alerte de sécurité critique* »

Voilà qui a de quoi effrayer et c'est clairement l'objectif. Je dis bien « effrayer » et non « sensibiliser », car, « *Quelqu'un vient d'utiliser votre mot de passe pour essayer de se connecter à votre compte à partir d'une application n'appartenant pas à Google* » (notez avec quelle condescendance il est précisé qu'il ...

**Motoriser une antenne directionnelle avec un ESP8266 (/Hackable/HK-035/Motoriser-une-antenne-directionnelle-avec-un-ESP8266)**

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

**Électronique (/search/node?domains%5B0%5D=72452)**

**Radio et wireless (/search/node?domains%5B0%5D=72465)**

Cet article aurait pu s'intituler « pointez les choses dans le ciel avec un ESP8266 », car en réalité, l'application de ce qui va suivre à la réception de signaux venus de l'espace n'est qu'une utilisation parmi tant d'autres. Notre objectif ici sera de motoriser une antenne de façon à la pointer automatiquement en direction d'un émetteur mobile, et plus exactement, un satellite en orbite basse (< 2000 km). Le tout, bien entendu, en le suivant alors qu'il se déplace. ...

**Le module du moment : Décodeur DTMF (/Hackable/HK-035/Le-module-du-moment-Decodeur-DTMF)**

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

DTMF, pour *dual-tone multi-frequency* est un encodage où chaque symbole d'un alphabet de 16 correspond à un couple de deux fréquences audibles, parmi une collection de 8, utilisées de concert. Initialement créé pour la téléphonie fixe, ce système permet historiquement d'encoder les pressions sur les touches d'un téléphone (« 0 » à « 9 », « A » à « D », plus « \* » et « # ») et de transmettre le signal correspondant sous forme de sons.

...

**Pilotez de manière optimale vos afficheurs LED (/Hackable/HK-034/Pilotez-de-maniere-optimale-vos-afficheurs-LED)**

Hackable n° 34 (/Hackable/HK-034) juillet 2020

Par Bodor Denis (/auteur/Bodor-Denis)

**Code (/search/node?domains%5B0%5D=72464)**

Électronique (/search/node?domains%5B0%5D=72452)

Trop souvent, dans les forums et/ou sur les sites web, certains ont tendance à conseiller l'approche « facile » plutôt que l'approche « efficace ». Qui n'a jamais vu un jour quelqu'un répondre « mais utilises donc xxx(), ça marche et c'est plus simple » en réponse à une problématique précise ? C'est là, généralement, le fait de personnes qui n'ont que peu d'expérience ou ne comprennent simplement pas la motivation du demandeur. Voici une petite ...

**Le module du moment : afficheur matrice led 8x32 (/Hackable/HK-034/Le-module-du-moment-afficheur-matrice-led-8x32)**

Hackable n° 34 (/Hackable/HK-034) juillet 2020

Par Bodor Denis (/auteur/Bodor-Denis)

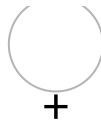
Électronique (/search/node?domains%5B0%5D=72452)

La bibliothèque MD\_MAX72XX de Marco Colli (alias MajicDesigns) permet de piloter un ou plusieurs modules en configuration linéaire (les uns après les autres, sur une ligne), mais elle forme également la base de la bibliothèque MD\_MAXPanel supportant des agencements en panneaux et fournissant des primitives graphiques intéressantes (points, lignes, rectangles, cercles, etc.).

...







## Articles qui pourraient vous intéresser... (/search/node?authors%5B0%5D=70942)

### Fonctionnement d'un téléphone mobile sous l'eau (/Hackable/HK-035/Fonctionnement-d-un-telephone-mobile-sous-l-eau)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Carry Emile (/auteur/Carry-Emile), Friedt Jean-Michel (/auteur/Friedt-Jean-Michel)

#### Radio et wireless (/search/node?domains%5B0%5D=72465)

Parmi les idioties que nous lisons quotidiennement dans la presse grand public, la technologie, et les liaisons radiofréquences en particulier, est souvent en haut du palmarès, faute de compétence des auteurs et du moindre esprit critique sur leurs affirmations, sans néanmoins atteindre le niveau des statistiques qui sont l'apothéose de l'art de cacher le mensonge [1]. ...

### Simulation d'un ordinateur mécanique en scriptant sous FreeCAD (/Hackable/HK-035/Simulation-d-un-ordinateur-mecanique-en-scriptant-sous-FreeCAD)

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Friedt Jean-Michel (/auteur/Friedt-Jean-Michel), Testault Olivier (/auteur/Testault-Olivier), Carry Emile (/auteur/Carry-Emile)

#### Radio et wireless (/search/node?domains%5B0%5D=72465)

#### Code (/search/node?domains%5B0%5D=72464)

L'évolution du traitement du signal est une histoire fascinante largement déroulée par David Mindell dans ses divers ouvrages [1] et citations [2]. Partant de l'ordinateur mécanique avec ses rouages, poulies, bielles et crémaillères, le passage à l'électrique au début du 20ème siècle, puis à l'électronique intégrée avec l'avènement du transistor et des circuits intégrés (VLSI) nous ont fait oublier les stades initiaux qui ont amené à notre statut actuel d'ordinateurs infiniment puissants, ...

### MiSTer : La solution rétro ultime ? (/Hackable/HK-035/MiSTer-La-solution-retro-ultime)


Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

**Rétro (/search/node?domains%5B0%5D=72471)**

**Embarqué (/search/node?domains%5B0%5D=72457)**

Imaginez, un matériel unique capable de simuler fidèlement n'importe quel ordinateur, console et borne d'arcade rétro, sans émulation et vous donnant accès à une collection presque infinie de programmes, de démos et de jeux. Le tout compatible avec du matériel moderne (HDMI, USB, etc.), activement développé et accessible pour un budget raisonnable (mais pas économique pour autant). Cette solution existe, c'est MiSTer ! ...

 **Identification automatique de signaux grâce à la fonction d'autocorrélation (/MISC/MISCHS-022/Identification-automatique-de-signaux-grace-a-la-fonction-d-autocorrelation)**

MISC HS n° 22 (/MISC/MISCHS-022) octobre 2020

Par Kuchly Denis (/auteur/Kuchly-Denis)

**Sécurité (/search/node?domains%5B0%5D=72467)**

**Radio et wireless (/search/node?domains%5B0%5D=72465)**

Vous connaissiez la chasse au trésor... Initiez-vous maintenant à la chasse au signal (signal hunting en anglais). La chasse au signal consiste à rechercher les signaux qui nous entourent dans l'espace invisible du spectre électromagnétique. Mais plus besoin de rester l'oreille collée sur un haut-parleur à tourner le bouton pour régler la fréquence. La SDR (Software Defined Radio) a révolutionné tout cela : une radio numérique et un PC est vous voilà armé pour découvrir ce ...

**Motoriser une antenne directionnelle avec un ESP8266 (/Hackable/HK-035/Motoriser-une-antenne-directionnelle-avec-un-ESP8266)**

Hackable n° 35 (/Hackable/HK-035) octobre 2020

Par Bodor Denis (/auteur/Bodor-Denis)

Électronique (/search/node?domains%5B0%5D=72452)

Radio et wireless (/search/node?domains%5B0%5D=72465)

Cet article aurait pu s'intituler « pointez les choses dans le ciel avec un ESP8266 », car en réalité, l'application de ce qui va suivre à la réception de signaux venus de l'espace n'est qu'une utilisation parmi tant d'autres. Notre objectif ici sera de motoriser une antenne de façon à la pointer automatiquement en direction d'un émetteur mobile, et plus exactement, un satellite en orbite basse (< 2000 km). Le tout, bien entendu, en le suivant alors qu'il se déplace. ...



(/search/node?  
domains%5B0%5D=72457&domai

**DOMAINES**



**NOS PUBLICATIONS**



**AIDE & CONTACT**



**ABONNEMENTS**



**À PROPOS**



Tél. : 03.67.10.00.20