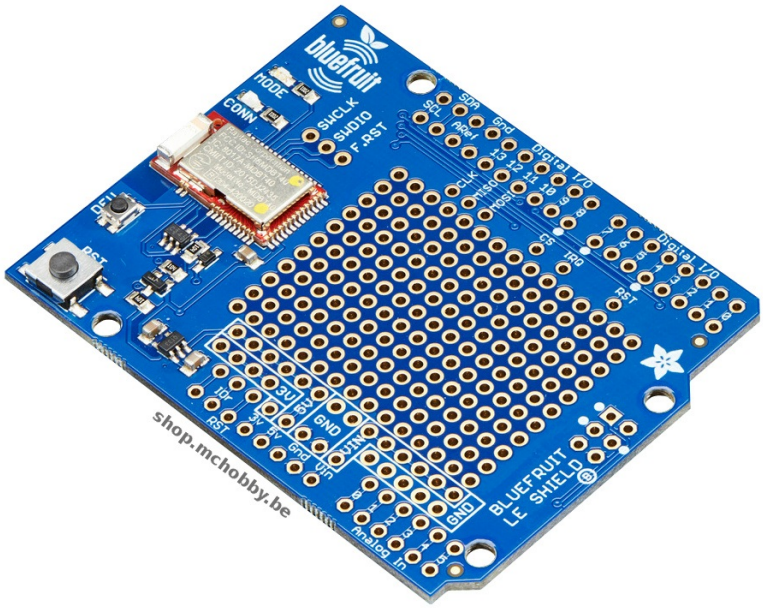


Guide utilisateur BlueFruit LE

Un guide complet et abordable pour utiliser et exploiter les modules Bluefruit LE d'Adafruit.

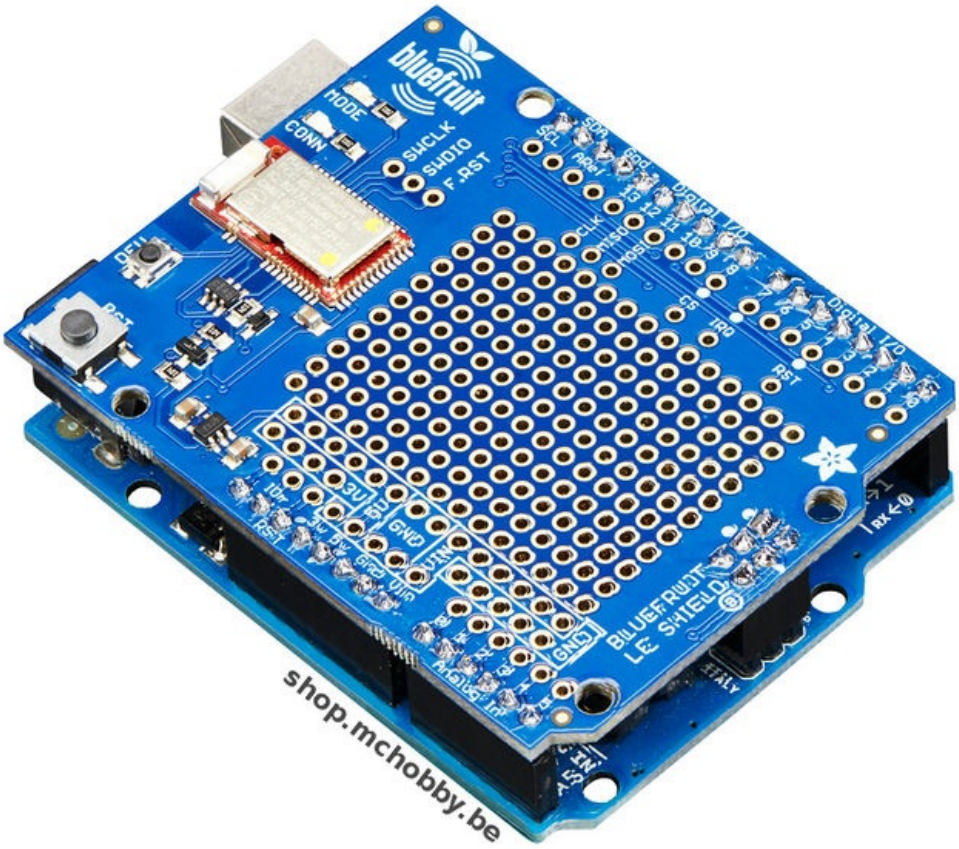
(version 0.1)

Traduit par MicroControleur Hobby (shop.mchobby.be)
Compilé depuis la traduction maintenue sur https://wiki.mchobby.be/index.php?title=Bluefruit-LE-Shield
Les hyperliens sont disponibles sur la version en ligne du document.
Translated by MicroControleur Hobby (shop.mchobby.be)
Compiled from online translation available at https://wiki.mchobby.be/index.php?title=Bluefruit-LE-Shield
Hyperlinks are available on the online version of this document.

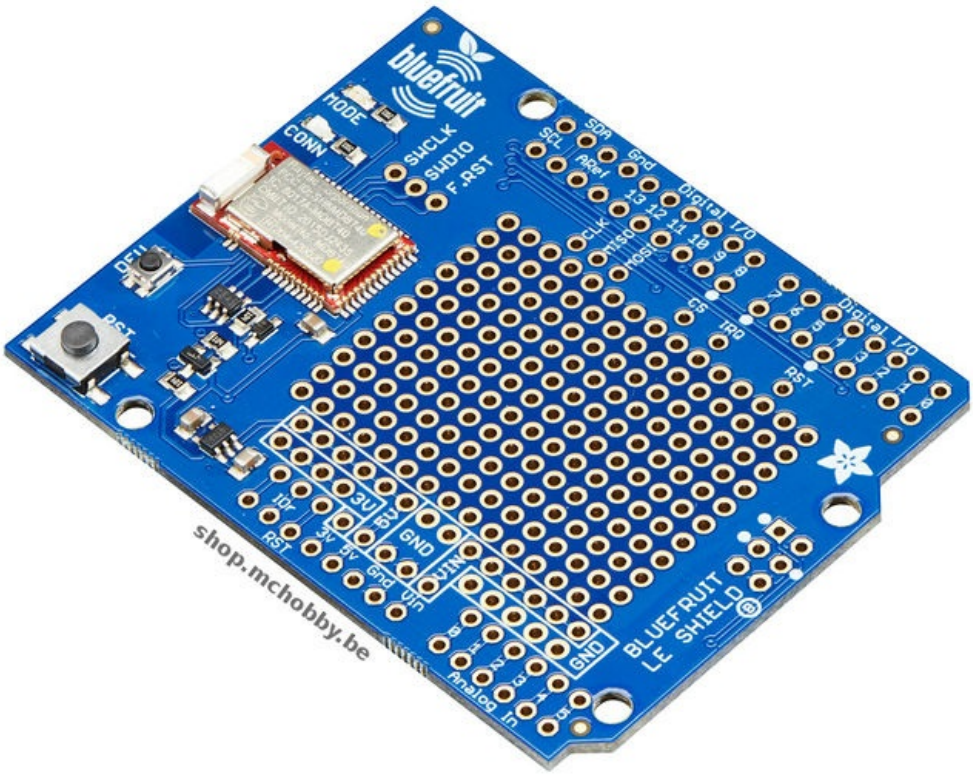
Introduction

Sommaire

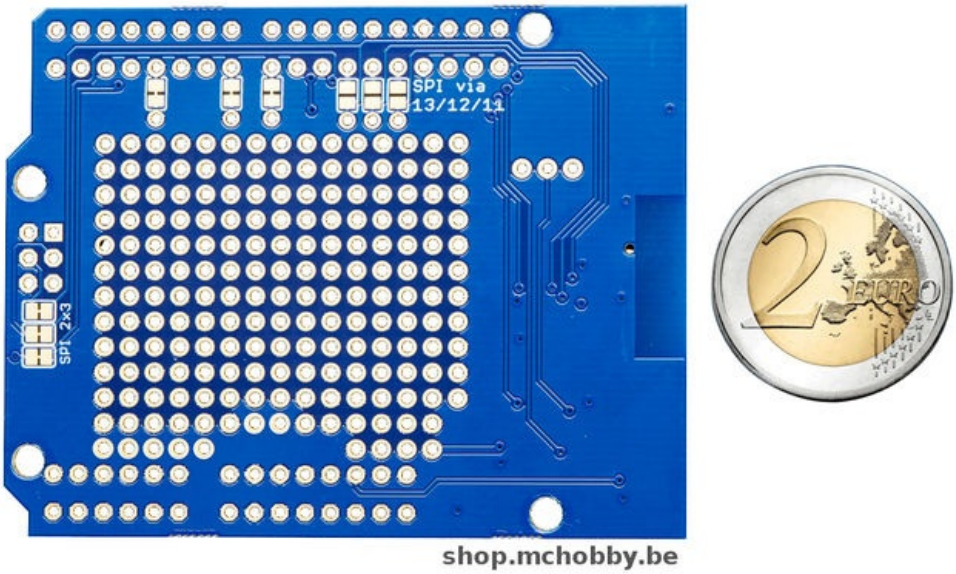
Présentation



Voudriez-vous ajouter une puissant interface Bluetooth Low Energy (et facile à mettre en oeuvre) sur votre robot, projet artistique ou autre projets électroniques? Maintenant que BLE est inclus dans la plupart des smartphones et tablette, disposer d'une connectivité Bluetooth sur son projet permet de créer des interfaces particulièrement pratique pour vos projets Arduino!



Le shield Bluefruit LE facilite l'ajout d'une connectivité Bluetooth Low Energy sur votre Arduino (ou compatible Arduino). Soudez les connecteurs sur la carte, insérez là sur votre Arduino et c'est parti. Le shield utilise l'interface SPI matériel (MISO, MOSI, SCK) plus un signal "chip select" (D8 par défaut), ligne d'interruption (D7 par défaut) et le signal (D4 par défaut). Vous pouvez réarranger n'importe quelle broche (ou toutes les broches) si vous en avez besoin. Il suffit de couper les pistes entre les cavaliers à souder (sous la carte) et souder des fils depuis le cavalier vers la broche que vous souhaiter utiliser.



Ce module multi-fonction peut faire bien plus encore! La plupart des utilisateurs apprécient l'utilisation du profile standard UART de Nordic. Dans ce profile, Bluefruit LE agit comme un tube transportant les données, qui transmet les données de façon 'transparente' dans les deux sens avec votre périphérique iOS ou Android. Vous pouvez utiliser l'App iOS ou l'App Android d'Adafruit pour commencer à envoyer rapidement des données depuis votre projet Arduino vers votre smartphone rapidement (et sans difficulté).

La carte est capable de faire bien plus que simplement envoyer des données dans les airs! Grâce à un ensemble de commande AT facile à apprendre, vous disposer d'un contrôle totale sur le comportement du périphérique. Cela inclus la possibilité de de définir et manipuler vos propres services GATT et ses caractéristiques, ou changer la façon dont le périphérique s'annonce auprès des autres périphériques Bluetooth Low Energy. Vous pouvez utiliser les commandes AT pour lire la température de la puce, vérifier la tension de l'accu, vérifier la connexion RSSI ou adresse MAC et bien plus encore.

Les options sont tellement nombreuses que la liste serait beaucoup trop longue pour être énumérée ici!

Bluefruit App: Démarrer rapidement avec le module BLE

En utilisant l'App iOS ou l'App Android BlueFruit d'Adafruit, vous pourrez rapidement interagir avec vos prototypes de projet en

utilisant une tablette/smartphone iOS (ou Android) comme contrôleur. Les Apps dispose d'un color picker (sélection de couleur), quaternion/accéleromèter/gyroscope/magnétomètre ou GPS (localisation), ainsi qu'un gamepad à 8 buttons. Après avoir connecté votre BlueFruit, vous pourrez envoyer des commandes -via Bluetooth- en moins de 10 minutes.

Les hackeurs seront très heureux d'utiliser le profile de connexion UART Rx/Tx standard (port série) de Nordic Télécom. Dans ce profile, Bluefruit agit comme un tube de transport de donnée, il transmet de façon 'transparente' toutes les données entre votre iOS (ou Android) et votre projet. Vous pouvez utiliser l'App iOS ou l'App Android, ou écrire votre propre App pour communiquer avec le service UART.

Plus de fonctionnalités

Avec BlueFruit LE vous pouvez faire bien plus encore!

- Le Bluefruit peut également agir comme un périphérique HID (clavier) (pour les périphériques/appareils supportant BLE HID)
- Devenir un monitor cardiaque BLE (BLE Heart Rate Monitor) (un profile standard de BLE) - vous aurez juste besoin d'ajouter l'électronique de détection de pulsation.
- Le transformer en UriBeacon, le standard Google pour les beacons Bluetooth LE. Alimentez le et votre "BLE Friend émettra une URL vers tous les périphériques alentour disposant de l'application UriBeacon.
- Dispose d'une fonctionnalité over-the-air bootloading de sorte qu'Adafruit peut proposer des mise-à-jours avec les derniers firmwares. Utiliser n'importe quel appareil Android ou iOS pour obtenir et installer les mise-à-jours!

Caractéristiques

- Coeur ARM Cortex M0 à 16MHz
- 256KB de mémoire Flash
- 32KB SRAM
- Transport: SPI à 4MHz avec interruption matériel (5 broches requises)
- Entrées tolérante 5V (compatible Arduino Uno, etc.)
- Inclus un régulateur 3.3V
- Bootloader avec mise-à-jour du firmware via OTA (Over The Air, via les airs)
- Commandes AT pour configurer rapidement le module
- **Fiche technique, schéma et élément Fritzing disponible dans le tutoriel**
- Dimensions du produit: 67.1mm x 51.7mm x 10.0mm
- Poids: 15.4g

Pourquoi utiliser un module Adafruit Bluefruit?

Il existe de nombreux modules BLE sur le marché, de qualité très variables et de conceptions matériels et du firmware très disparate.

Alors pourquoi choisir un produit Adafruit?

Un des grands avantages de la famille des modules Adafruit Bluefruit LE est qu' Adafruit a entièrement écrit le firmware fonctionnant sur le module. Adafruit contrôle chaque ligne de code qui fonctionne sur ses modules... et donc, Adafruit et ses clients, ne sont pas à la merci du firmware produit par une tierce personne (ou fournisseur de matériel) qui n'est pas forcement porté sur la mise-à-jour rapide, ajout de fonctionnalité de son module/firmware.

Comme Adafruit contrôle l'entièreté de son module et firmware et Adafruit à toutes les cartes en main pour faire des correctifs et ajouter des fonctionnalités. Pour une fois, les Makers seront mieux servit que l'industrie!

A propos de MCHobby
Crédit de traduction

Sommaire

A propose de .: MC Hobby :.

MCHobby investi du temps et de l'argent dans la réalisation de traduction et/ou documentation. C'est un travail long et fastidieux réalisé dans l'esprit Open-Source... donc gratuit et librement accessible.
SI vous aimez nos traductions et documentations ALORS aidez nous à en produire plus en achetant vos produits chez MCHobby http://shop.mchobby.be .

MC Hobby SPRL, basé en Belgique, est le traducteur de ce manuel.

MC Hobby cherche à promouvoir, en français, la plateforme Open-Source Arduino, ses extensions et exemples de programmation afin de la rendre accessible au plus grand nombre.

License

Crédit de traduction

Toute référence, mention ou extrait de cette traduction doit être explicitement accompagné du texte de crédit de traduction.

Ce texte est repris en "pied de document" sur toutes les pages/documents, merci d'en prendre connaissance.

Limite de traduction

Anglicisme

Du fait que de nombreux anglicismes soient utilisés au jour le jour dans les milieux techniques, je me suis permis d'en préserver quelques-uns qui me semblaient avoir plus de sens dans la langue de Shakespeare que traduit dans la langue de Voltaire. Par ailleurs, ces anglicismes pourraient se révéler fort utiles lors de vos prochaines recherches sur Internet.

Ainsi, vous retrouverez les termes suivants :

- Pin : fait référence à une « broche » de raccordement d'un composant (Anglicisme fort répandu).
- Datasheet : fait référence à la « fiche technique » d'un composant.
- Pin Header : malheureusement intraduisible sans en perdre le sens mais vous aurez vite identifié les « Pinheaders ». La traduction courant est *connecteur*, ce terme restant beaucoup trop vague.

• LED : ce sont les diodes électroluminescentes (aussi appelées DEL). Cependant, le terme LED est si répandu qu'il a été préservé.
• RGB : Fait référence aux 3 couleurs fondamentales Rouge Vert Bleu (Red Green Blue). Si le terme RVB est d'usage fréquent en Belgique et en France, l'acronyme RGB reste encore le plus répandu.

Code source

Nous avons traduit les commentaires dans les programmes afin de les rendre plus intelligibles.

Par contre, nous n'avons pas modifié les codes sources (nom des variables, ...) qui, eux, restent ceux fournit par le concepteur et le fabricant du kit.

Autorisations

Le présent manuel a été traduit avec l'autorisation d'Adafruit (www.adafruit.com).

Signaler une erreur

Malgré tout le soin apporté à la réalisation de cette traduction, il n'est pas impossible qu'une erreur se soit malgré tout glissée dans ce document. N'hésitez pas à nous envoyez un e-mail si vous en constatiez une. Pour adresser vos remarques et commentaires relatifs à la traduction, ou pour demander une traduction complémentaire, contactez nous par e-mail à

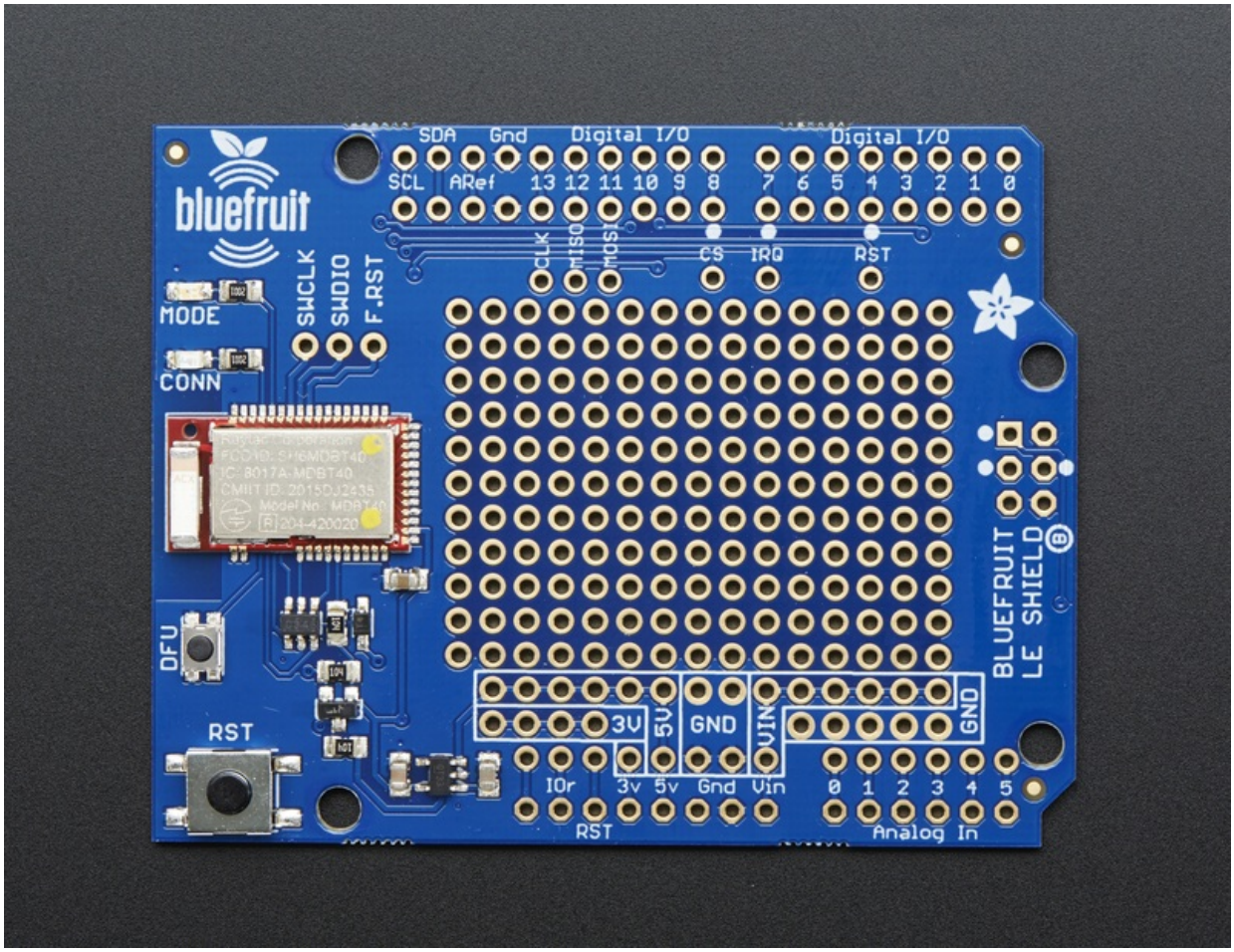
- support (arobase) mchobby.be.

N'oubliez pas de mentionner le manuel/page/lien en mentionnant l'erreur ;-)

Brochage

Sommaire

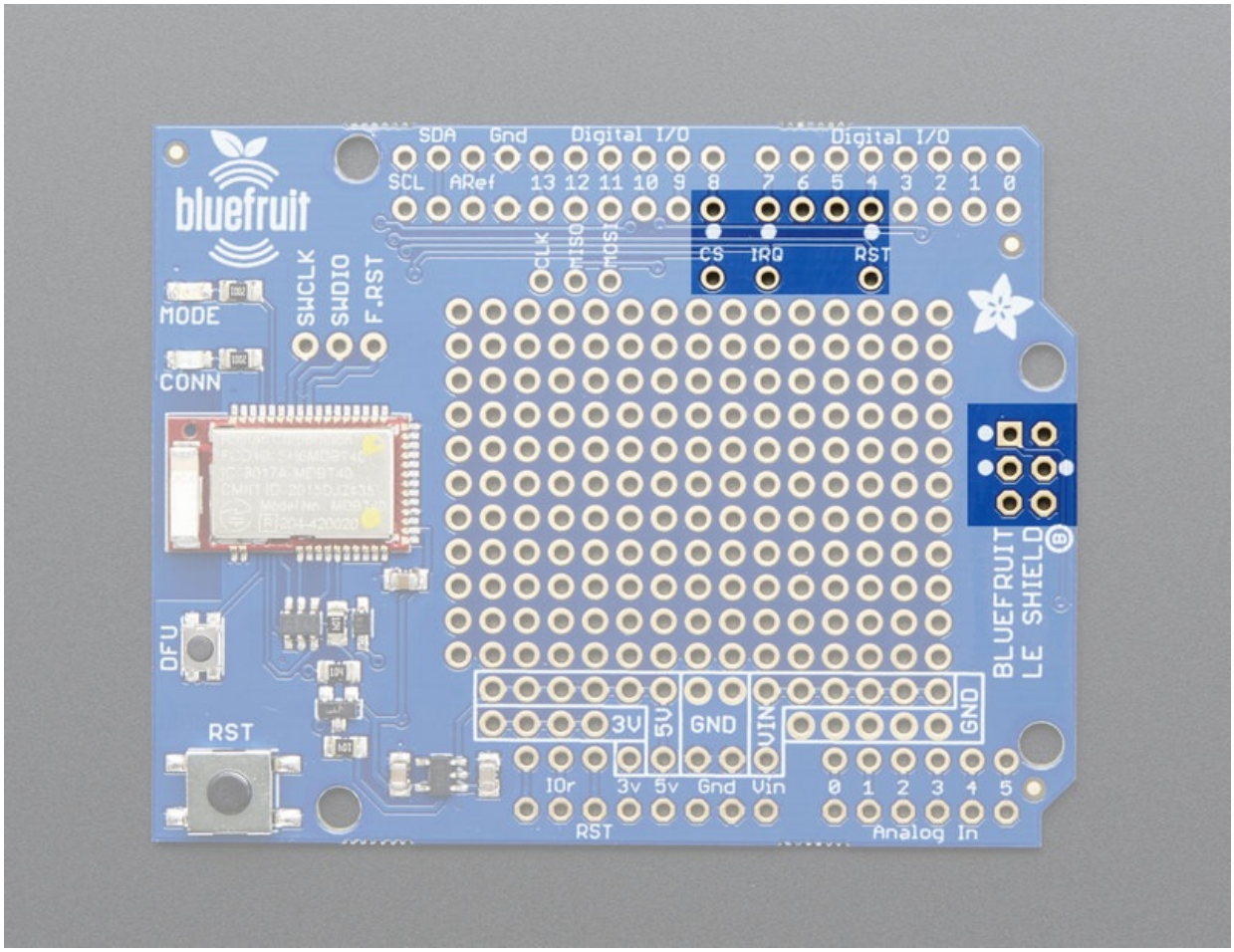
Brochage



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Broches d'alimentation

- **5V**: Ceci est l'alimentation du module. Alimentez la en entrée avec une source de 3.3 à 5V. La tension sera ensuite régulée à 3.3V pour faire fonctionner la puce.
- **GND**: La masse commune (alimentation et signaux logiques).

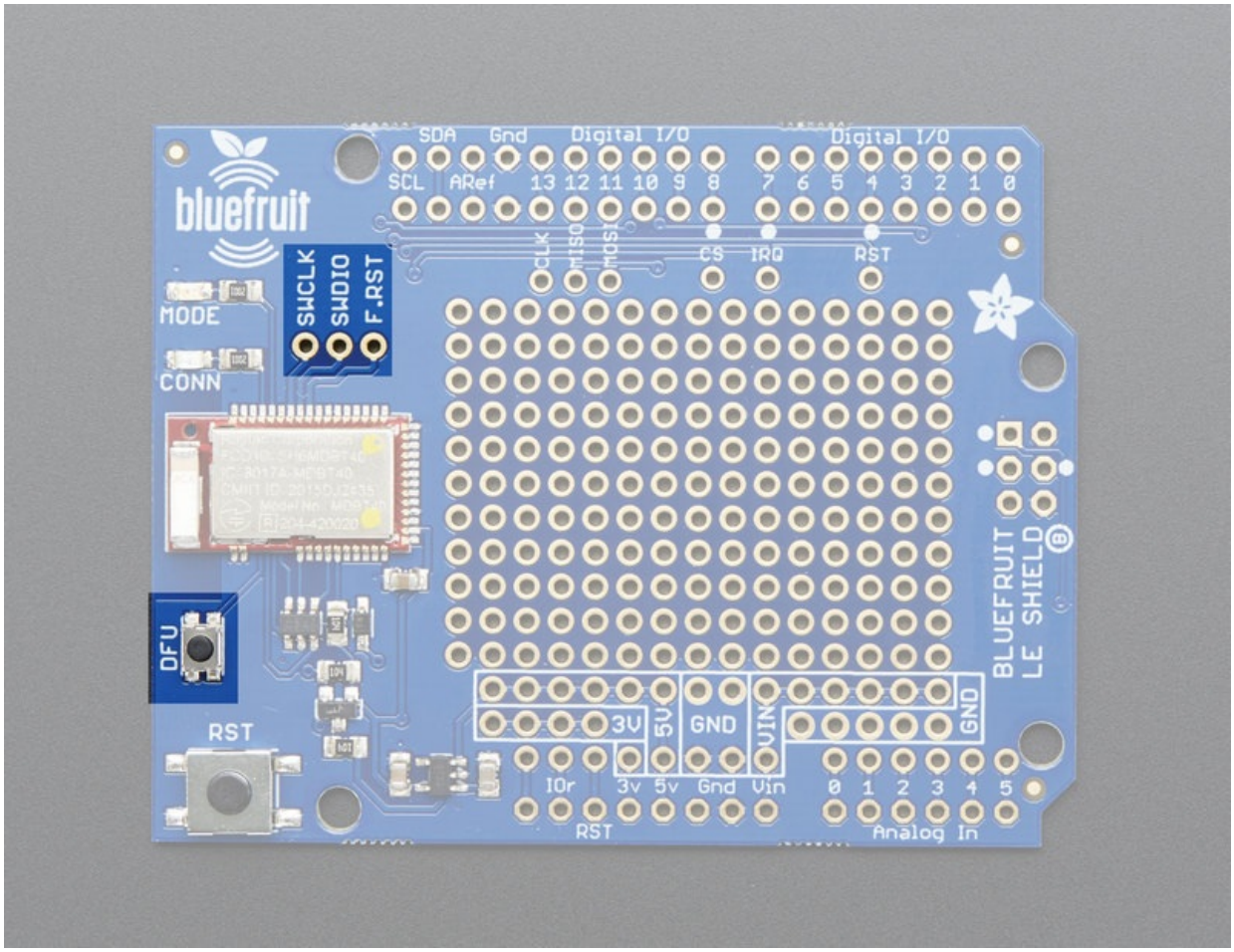


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Broches SPI

Le shield communique avec votre Arduino via le bus SPI.

- **SCK**: Il s'agit du signal d'horloge du bus SPI. Connecté par défaut sur la broche d'horloge sur bus SPI matériel de votre Arduino (via le connecteur ICSP 2x3).
- **MISO**: Il s'agit du signal *Master In Slave Out* du bus SPI (communication du nRF51 --vers--> Arduino). Connecté par défaut sur la broche MISO du bus SPI matériel (via le connecteur ICSP 2x3).
- **MOSI'*: Il s'agit du signal* Master Out Slave In** du bus SPI (communication de l'Arduino --vers--> nRF51). Connectés par défaut sur la broche MOSI du bus SPI matériel (via le connecteur ICSP 2x3).
- **CS**: Il s'agit de la broche *Chip Select* (qui indique à la puce que les communications du bus SPI se font avec elle... et pas un autre composant éventuellement branché sur le même bus). Connectée par défaut sur la broche **#8**
- **IRQ**: C'est la broche d'interruption nRF51 --vers--> Arduino qui permet au nRF51 d'informer le microcontrôleur lorsque des données sont disponibles sur le nRF51. Indique à votre Arduino/microcontrôleur qu'une nouvelle transaction SPI doit être initiée. Connecté par défaut sur la broche digital **#7**
- **RST**: Placez cette broche au niveau bas (LOW) pour faire un reset du module Bluefruit. Branché par défaut sur la broche digital **#4**



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

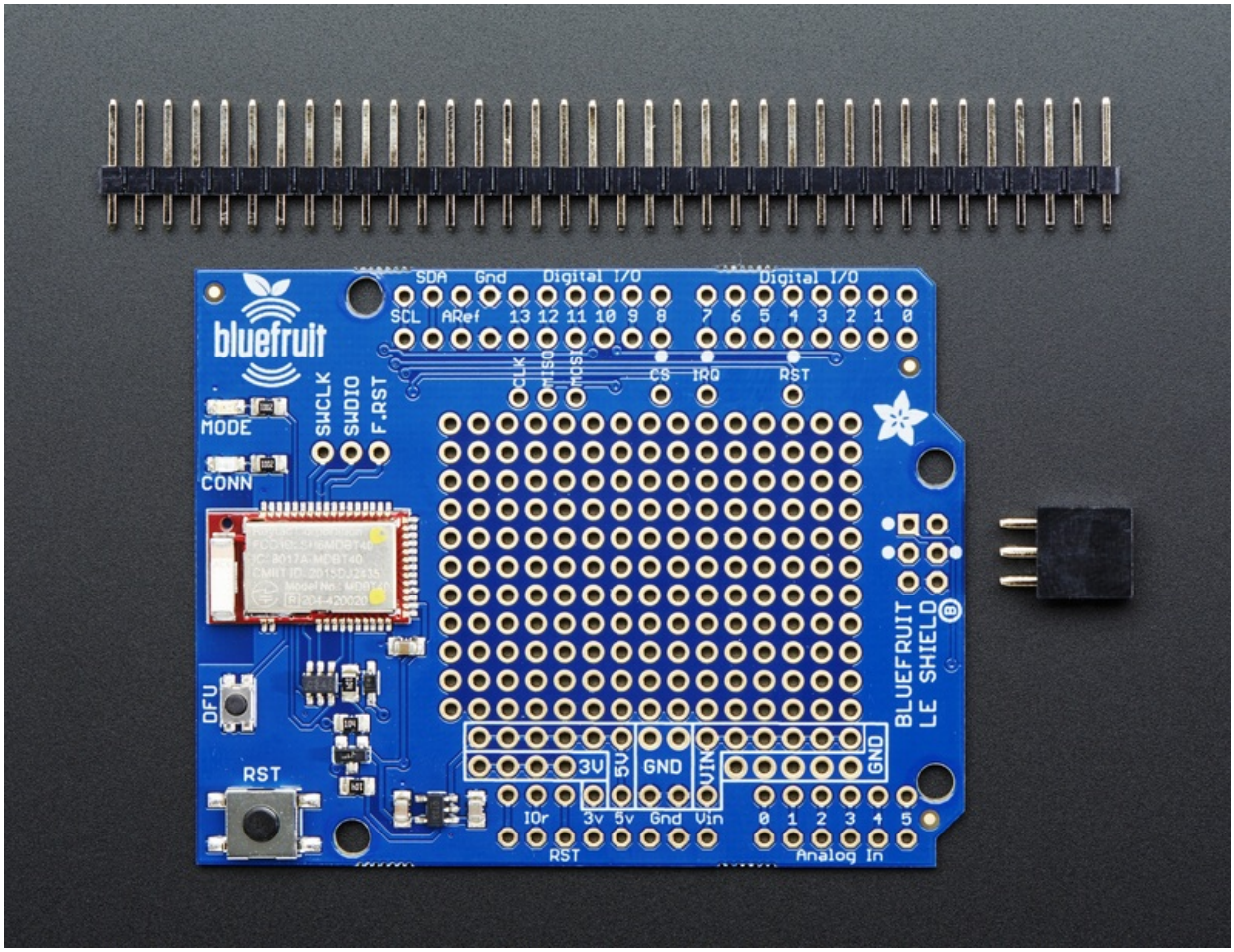
Autres broches

- **SWCLK**: est la broche d'horloge SWD (SWCLK), logique 3v - pour les hackers expérimentés!
- **SWDIO**: est la broche de donnée SWD (SWDIO), logique 3v - pour les hackers expérimentés!
- **F.RST**: broche *Factory Reset* (réinitialisation d'usine). Lorsque vous avez fait une opération qui a totalement planté le module et que vous ne savez plus rien faire avec lui ALORS branchez cette patte sur la masse pendant la mise sous tension du module (le module fera alors une réinitialisation d'usine). **Vous devriez d'abord essayer de faire une réinitialisation via DFU** (voyez cette page du tutoriel).
- **bouton DFU**: Presser ce bouton lorsque le shield est mis sous tension forcera le module Bluefruit LE à entrer dans un mode spécial de mise-à-jour du Firmware (appelé *Device Firmware Update* ou aussi DFU). Le mode DFU permet de faire une mise-à-jour via les airs (via Bluetooth) à l'aide d'un smartphone.
  Lorsque le périphérique est sous tension, cette broche peut **également être utilisée pour faire une réinitialisation d'usine**; pressez le bouton pendant >5 sec jusqu'à ce que les deux LEDs commencent à clignoter alors, seulement, libérez la broche (placez là au niveau haut ou sur 5V) et le processus de réinitialisation d'usine (*factory reset*) sera exécuté.

Assembler

Sommaire

Les éléments



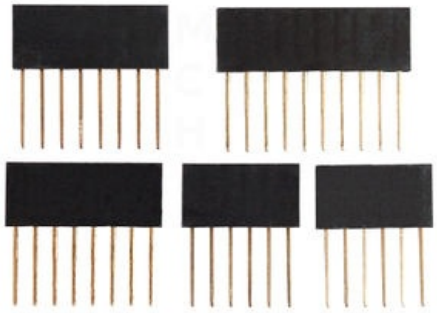
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Attention à l'empilement

Si vous désirez empiler un autre shield au dessus du shield Bluefruit LE, il sera nécessaire de vous procurer des "stacking headers" pour Arduino et les utiliser à la place des connecteurs fournit avec la carte!

Avec un Stacking Header

Besoin d'empiler? ce tutoriel se concentre sur l'utilisation d'un connecteur normal (pinHeader) pour Arduino. Ne suivez pas cette section du tutoriel si vous comptez utiliser des Stacking Headers http://shop.mchobby.be/product.php?id_product= !

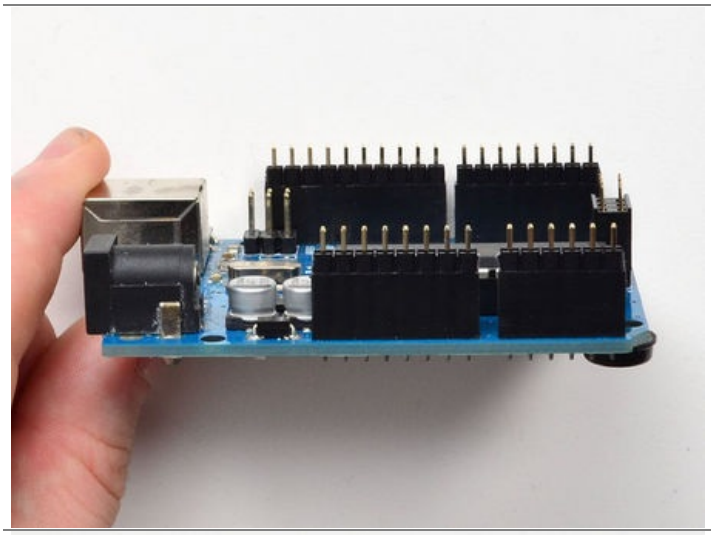


Branchez les connecteurs

Ces étapes concerne tous les Arduino et Arduino compatibles.

Soyez attentif au fait que c'est le port SPI matériel (le 2x3 broches) qui sera utilisé pour communiquer avec le module Bluefruit LE. Si ce port n'est plus disponible, il sera nécessaire de faire des pontages depuis les pastilles SCK/MOSI/MISO vers des autres broches de votre microcontroleur (et d'utiliser du "SPI logiciel")!

Toutes les images ci-dessous sont créditées à Adafruit Industries http://www.adafruit.com - All the images here under are credited to Adafruit Industries http://www.adafruit.com .

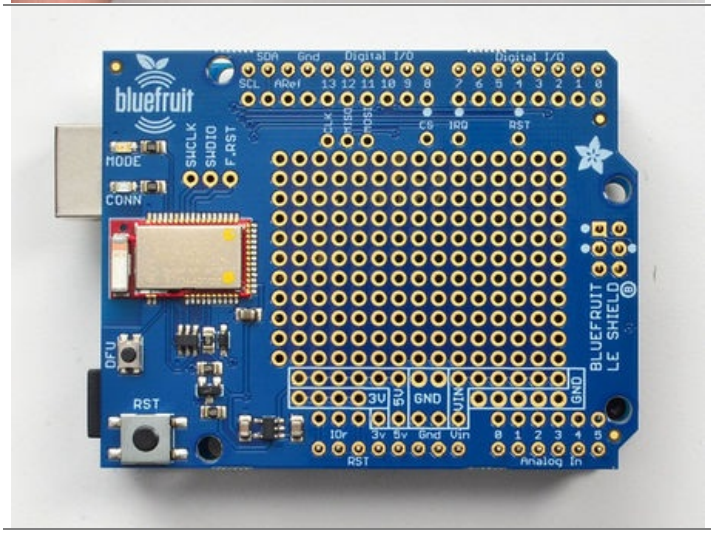


Nous allons commencer par briser les connecteurs 36 broches (pinHeader) en 4 sections:

- une section de 10 broches
- deux sections de 8 broches
- une section de 6 broches

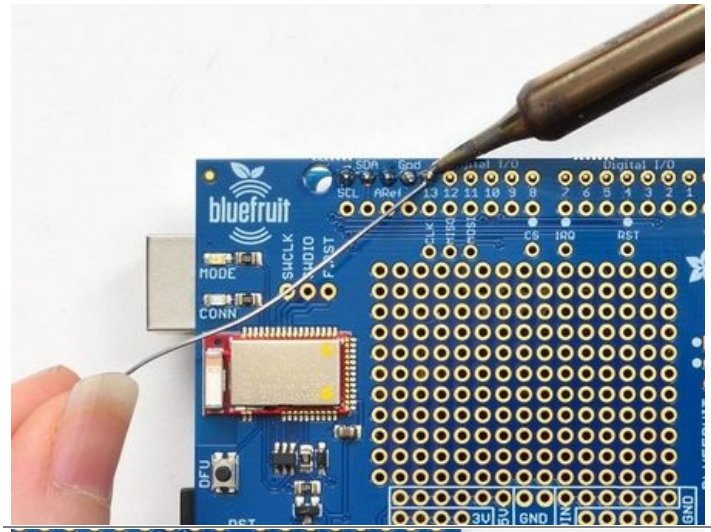
Placez la partie longue de ces section dans les connecteurs de votre Arduino.

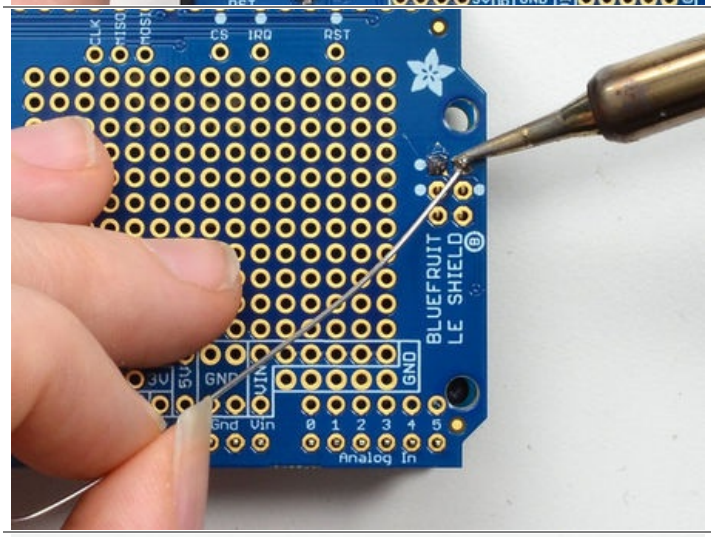
Placez également le connecteur femelle 2x3 sur le connecteur ICSP de votre Arduino (sur la droite de la carte)



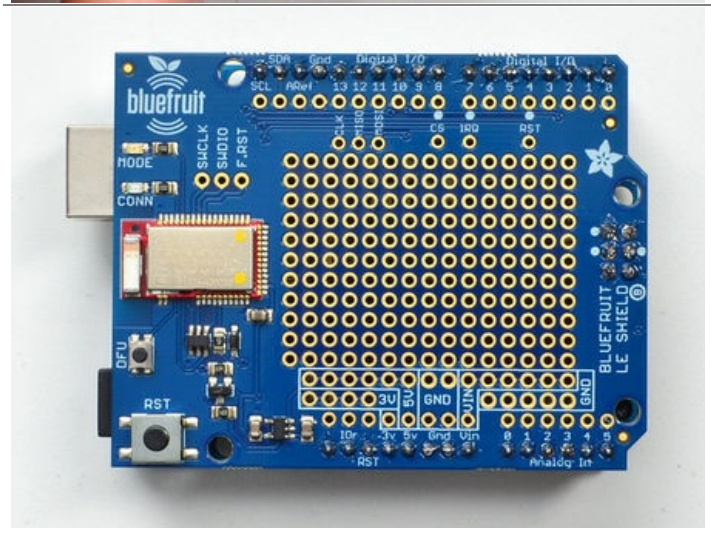
Placez le shield sur les connecteurs de sorte que les petites pointes des connecteurs passent dans tous les trous correspondant du shield. Cela doit parfaitement correspondre!

Souder ensuite tous les connecteurs sur votre shield.





N'oubliez pas souder le connecteur 6 broches! (c'est notre bus SPI matérie)



Vérifier toutes les soudures.

C'est OK? pas de soudure cassée ou soudure froide (en forme de boule qui ne mouille pas la pastille du shield)?

Alors nous pouvons passer à l'étape suivante

Brancher

Raccordement par défaut

Pour une mise en place rapide, Adafruit a déjà raccordé le Bluefruit LE sur les broches par défaut. Cela permet de suivre le tutoriel (qui exploite également les broches par défaut).

Le Bluefruit LE *SPI Friend* (comme le shield BlueFruit LE pour Arduino) ne devrait pas être modifié et utilisera le brochage le brochage suivant:

Broches SPI du Bluefruit LE	Broche Arduino
SCK	SPI Matériel: SCK
MISO	SPI matériel: MISO
MOSI	SPI matériel: MOSI
CS	8
IRQ *Interruption*	7
RST *Reset*	4

Par défaut, Adafruit utilise le bus SPI matériel https://www.arduino.cc/en/Reference/SPI . Ces broches SPI sont partagées avec d'autres broches digitales. Par exemple, sur un Arduino UNO, le bus SPI matériel utilise les broches #13, #12 et #11.

Si vous disposez d'un Arduino Uno (ou Atmega328 compatible) dont le connecteur 2x3 est manquant alors vous pouvez brancher des fils (sous le shield) pour réaliser le raccordement sur le bus SPI matériel de SCK/MISO/MOSI vers 13/12/11.

Si vous ne voulez pas utiliser le bus SPI matériel (connecteur 2x3 broches), il est toujours possible d'utiliser du SPI logiciel. Le SPI logiciel est plus lent mais permet d'utiliser n'importe quelle broche digital pour émuler le bus SPI logiciel (il faudra 3 broches). Utilisez simplement des fils pour réaliser des pontages entre les breakouts SCK/MISO/MOSI et les broches que vous souhaitez utiliser.

Si vous voulez modifier les broches SPI utilisés (le raccordement par défaut), il ne faut pas souder le connecteur SPI (2x3 broches) sur votre shield.

Modifier le brochage par défaut

Si vous voulez que les croquis/sketch d'exemple utilisent des broches SPI différentes (pour CS, IRQ ou RST) alors ouvrez le fichier **BluefruitConfig.h** disponible dans le répertoire exemple (de l'exemple que vous désirez tester).

Dans ce fichier .h, il faudra changer les broches pour qu'elles aient la nouvelle affectation souhaitée (Voyez la section "Logiciel" pour savoir comment installer la bibliothèque).

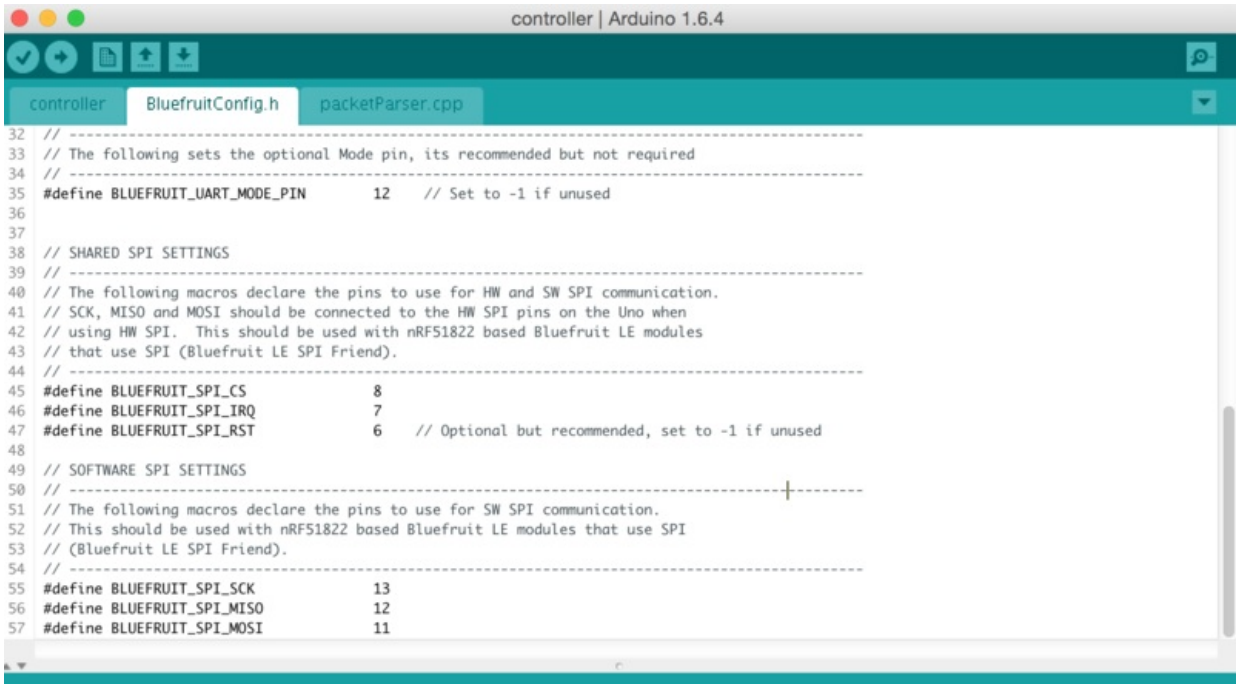
Si vous voulez utiliser **le SPI Logiciel (bitbang)**, vous pouvez utiliser les valeurs de broche suivantes pour SCK, MISO et MOSI:

```
// SPI Logiciel
#define BLUEFRUIT_SPI_CS          8
#define BLUEFRUIT_SPI_IRQ         7
#define BLUEFRUIT_SPI_RST         4
```

Pour revenir à la configuration SPI Matériel (sur un Arduino UNO), vous pouvez utiliser les valeurs de broches suivantes pour SCK, MISO et MOSI:

```
// SPI Materiel
#define BLUEFRUIT_SPI_SCK         13
#define BLUEFRUIT_SPI_MISO        12
#define BLUEFRUIT_SPI_MOSI        11
```

Le fichier **BluefruitConfig.h** peut être trouvé dans le volet dédicacé (voir ci-dessous):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Dans tous les codes d'exemple, Adafruit a ajouter une section en début de croquis/sketch reprenant les différentes manières de communiquer avec Bluefruit LE: port série matériel, port série logiciel, bus SPI matériel, bus SPI logiciel.

Il n'est pas possible d'utiliser la communication série sur un Bluefruit SPI. Il est cependant possible de choisir entre le bus SPI matériel ou SPI logiciel (bitbang).

Si vous désirez utiliser le bus SPI matériel, retirez le commentaire sur le bout de code suivant (et commentez les trois autres sections)

```
/* Traduction MCHobby: ...SPI Matériel utilisant les broches du SPI matériel SCK/MOSI/MISO (par défaut) et les broches CS/IRQ/RST définies par l'utilisateur */

/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

Si vous désirez utiliser du SPI logiciel (bitbang), retirez le commentaire sur le bout de code suivant. Vous pouvez utiliser 6 broches au choix (ou 5 si vous ne voulez pas utiliser RST)

```
/* Traduction MCHobby: ...SPI logiciel utilisant les broches SPI SCK/MOSI/MISO définies par l'utilisateur et les broches CS/IRQ/RST définies par l'utilisateur */

/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

La bibliothèque

Il sera nécessaire de télécharger la bibliothèque Adafruit BLE pour nRF51 d'Adafruit (connue comme Adafruit_BluefruitLE_nRF51) pour pouvoir essayer les démos BLE d'Adafruit.

Vous pourrez trouver le code ici sur github https://github.com/adafruit/Adafruit_BluefruitLE_nRF51 mais il est plus simple de cliquer sur le bouton de téléchargement ci-dessous:

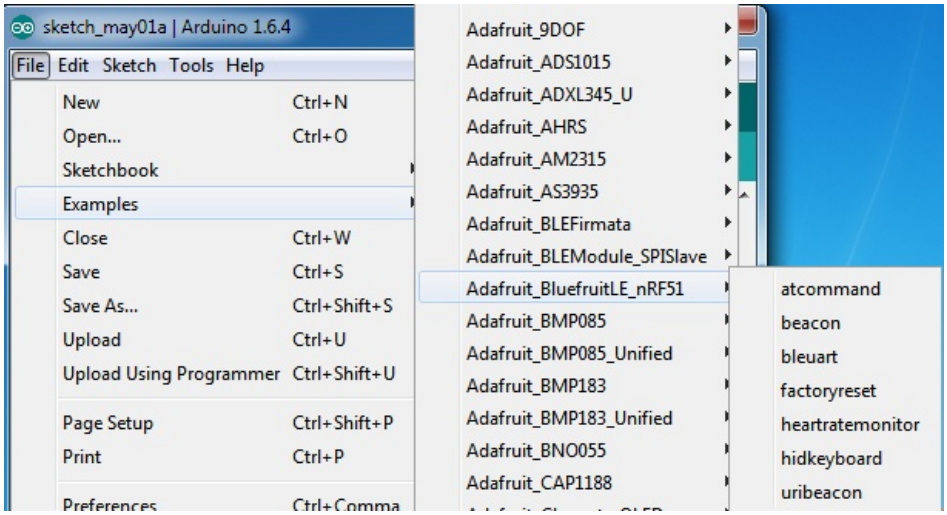
Après avoir décompressé l'archive, renommez le répertoire **Adafruit_BluefruitLE_nRF51** puis vérifier que le répertoire **Adafruit_BluefruitLE_nRF51** contient le fichier **Adafruit_BLE.cpp** et **Adafruit_BLE.h** (ainsi que de nombreux autres fichiers)

Placez le répertoire de la bibliothèque **Adafruit_BluefruitLE_nRF51** dans le répertoire `répertoire_des_croquis_arduino/libraries/` .

Vous pourriez avoir besoin de créer le sous-répertoire `libraries` si c'est la première bibliothèque que vous installez. Redémarrez votre Arduino IDE.

Nous disposons également d'un tutorial d'installation de bibliothèque Arduino.

Après avoir redémarré votre IDE, vérifiez que vous pouvez voir le répertoire de la bibliothèque dans les exemples Arduino:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

Sommaire

Attention

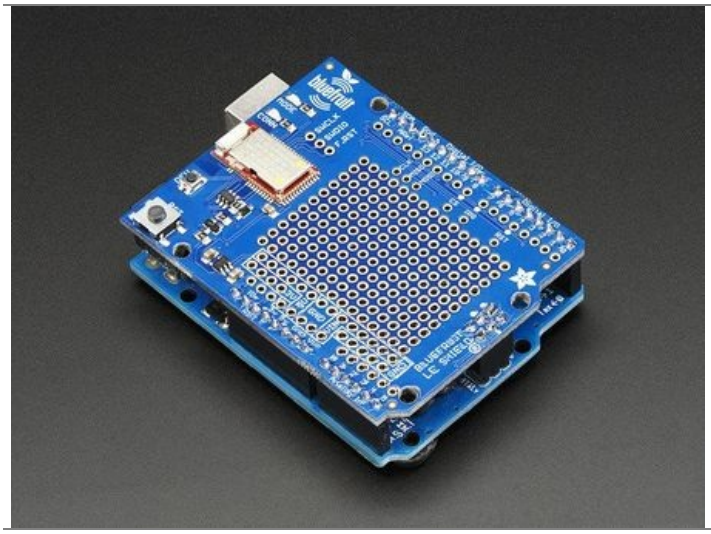
Avant de téléverser les croquis/sketchs d'exemple sur votre Arduino, vous aurez besoin de CONFIGURER l'interface du Bluefruit interface - soyez attentif car il y a de nombreuses options!

Quelle carte avez-vous?

Il y a plusieurs produits sous le nom Bluefruit.

Nous allons présenter ces produits et leur type d'interface dans la liste ci-dessous.

Toutes les images ci-dessous sont créditées à Adafruit Industries http://www.adafruit.com - All the images here under are credited to Adafruit Industries http://www.adafruit.com .

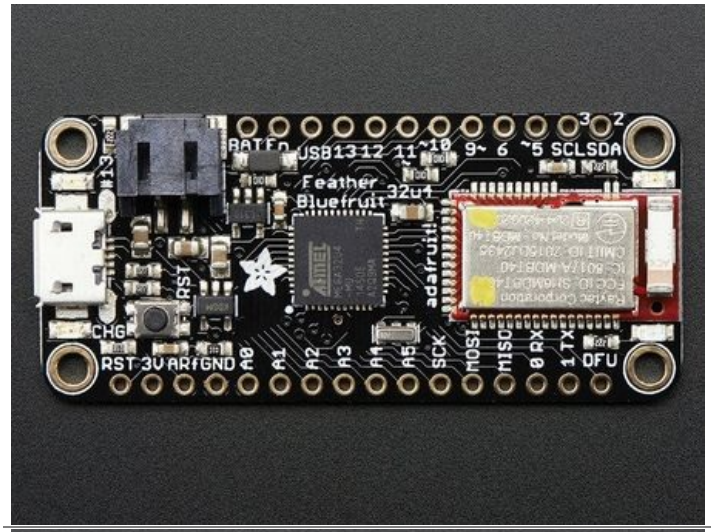


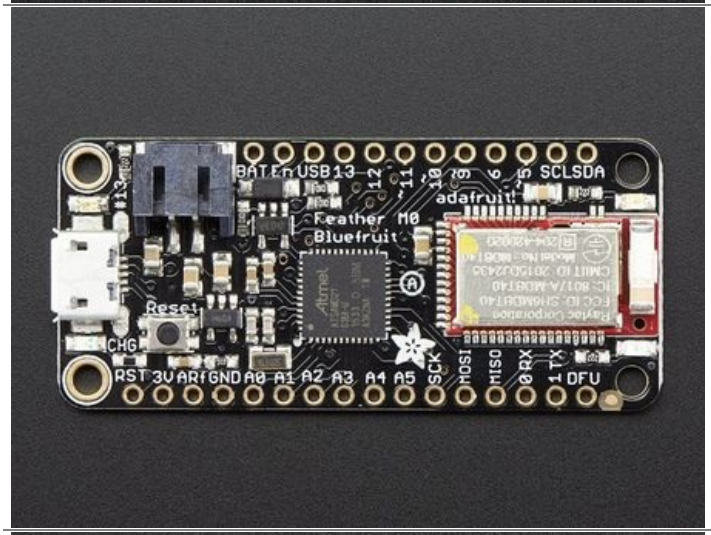
Si vous utilisez un Shield Bluefruit LE alors vous disposer d'un module **NRF51822 connecté en SPI**. Vous pouvez l'utiliser avec un **Atmega328** (Arduino UNO ou compatible), **ATmega32u4** (Arduino Leonardo, compatible) ou **ATSAMD21** (Arduino Zero, compatible). Et éventuellement d'autres plateformes.

Your pinouts are **Hardware SPI**, CS = **8**, IRQ = **7**, RST = **4**

Bluefruit Micro ou Feather 32u4 Bluefruit

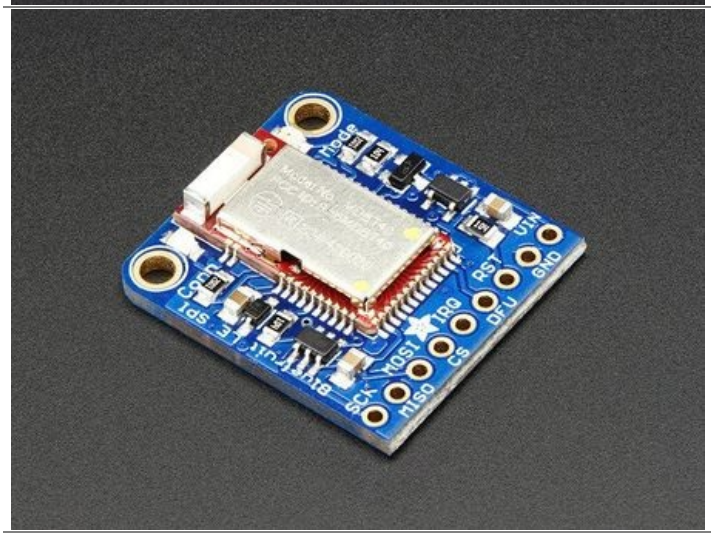
Si vous disposez d'un Bluefruit Micro ou Feather 32u4 Bluefruit LE alors vous disposez d'un ATmega32u4 avec un bus **SPI matériel**, CS = **8**, IRQ = **7**, RST = **4**





Feather M0 Bluefruit LE

Si vous disposez d'un Feather M0 Bluefruit LE alors vous avez un microcontrôleur ATSAMD21 avec un bus **SPI matériel**, CS = **8**, IRQ = **7**, RST = **4**



Bluefruit LE SPI Friend

Si vous disposez d'un module Bluefruit seul alors vous avez un peu plus de flexibilité avec vos raccordements. Adafruit recommande l'utilisation du bus **SPI matériel**, CS = **8**, IRQ = **7**, RST = **4**

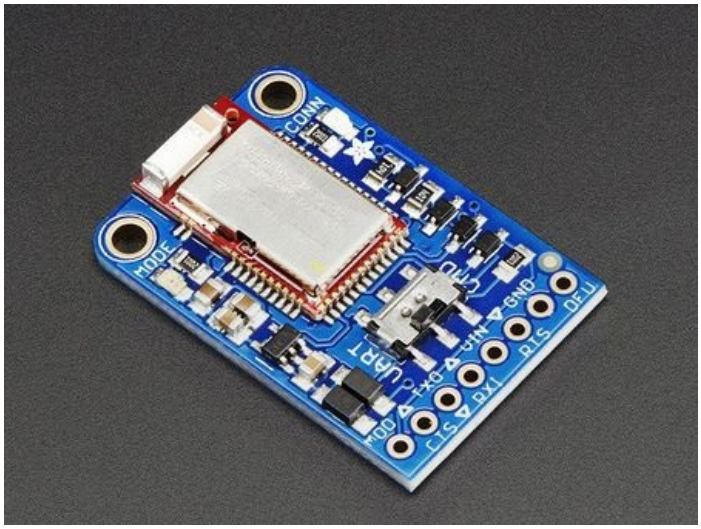
Vous pouvez utiliser ce module avec n'importe quel microcontrôleur disposant de 5 ou 6 broches (attention que dans ce cas, il vous faudra créer votre propre bibliothèque).

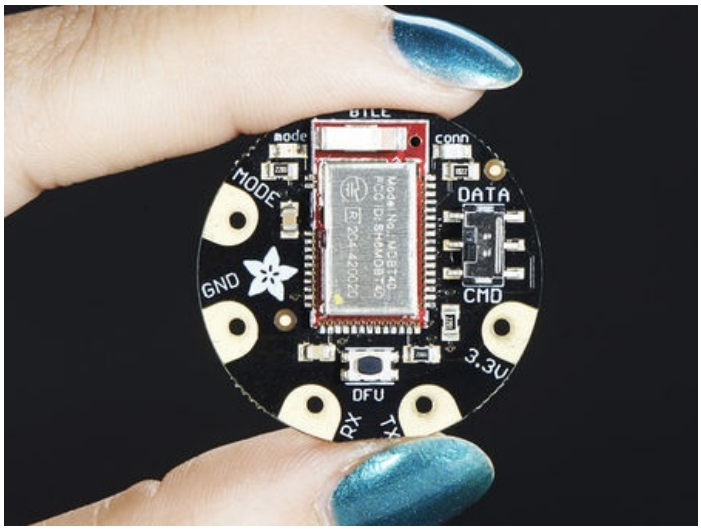
Bluefruit LE UART Friend -ou- Flora BLE

Si vous disposez d'un module Bluefruit LE UART vous avez également quelques flexibilités pour faire vis raccordements. Adafruit recommande l'utilisation **d'un UART matériel** si cela est possible. Si vous ne disposez pas d'un port UART matériel alors préférable d'utiliser le broche de contrôle de flux **CTS**. Ce module propose également une broche **MODE**.

Vous pouvez utiliser ce module avec n'importe quel microcontrôleur avec seulement 3 broches -mais le mieux reste encore d'utiliser un bus série/UART matériel!

Attention: sur une plateforme NON Arduino, il sera nécessaire de créer votre propre bibliothèque.

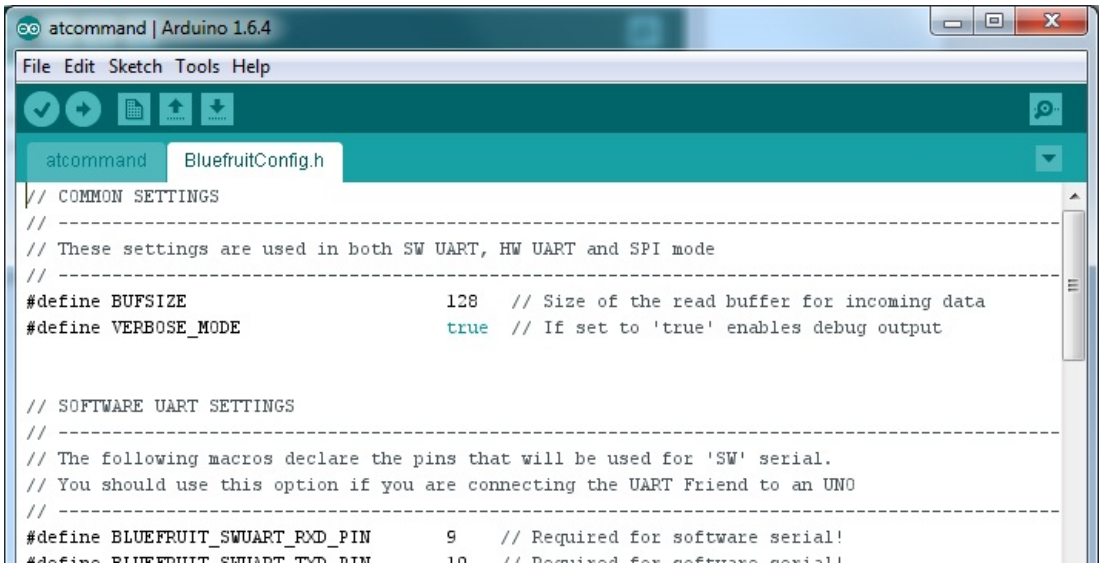




Configurer les broches utilisées

You'll want to check the Bluefruit Config to set up the pins you'll be using for UART or SPI

Each example sketch has a secondary tab with configuration details. You'll want to edit and save the sketch to your own documents folder once set up.



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Paramètres généraux

You can set up how much RAM to set aside for a communication buffer and whether you want to have full debug output. Debug

output is 'noisy' on the serial console but is handy since you can see all communication between the micro and the BLE

```
// -----------------------------------------------------------------------------
// These settings are used in both SW UART, HW UART and SPI mode
// -----------------------------------------------------------------------------
#define BUFSIZE                   128   // Size of the read buffer for incoming data
#define VERBOSE_MODE              true  // If set to 'true' enables debug output
```

UART logiciel

If you are using Software UART, you can set up which pins are going to be used for RX, TX, and CTS flow control. Some microcontrollers are limited on which pins can be used! Check the SoftwareSerial library documentation for more details

```
// SOFTWARE UART SETTINGS
#define BLUEFRUIT_SWUART_RXD_PIN      9   // Required for software serial!
#define BLUEFRUIT_SWUART_TXD_PIN      10  // Required for software serial!
#define BLUEFRUIT_UART_CTS_PIN        11  // Required for software serial!
#define BLUEFRUIT_UART_RTS_PIN        -1  // Optional, set to -1 if unused
```

UART Matériel

If you have Hardware Serial, there's a 'name' for it, usually Serial1 - you can set that up here:

```
// HARDWARE UART SETTINGS
#ifdef Serial1    // this makes it not complain on compilation if there's no Serial1
  #define BLUEFRUIT_HWSERIAL_NAME      Serial1
#endif
```

La broche MODE

For both hardware and software serial, you will likely want to define the MODE pin. There's a few sketches that dont use it, instead depending on commands to set/unset the mode. Its best to use the MODE pin if you have a GPIO to spare!

```
#define BLUEFRUIT_UART_MODE_PIN       12  // Set to -1 if unused
```

Broches SPI

For both Hardware and Software SPI, you'll want to set the **CS** (chip select) line, **IRQ** (interrupt request) line and if you have a pin to spare, **RST** (Reset)

```
// SHARED SPI SETTINGS
#define BLUEFRUIT_SPI_CS              8
#define BLUEFRUIT_SPI_IRQ             7
#define BLUEFRUIT_SPI_RST             4   // Optional but recommended, set to -1 if unused
```

Broches SPI logiciel

If you don't have a hardware SPI port available, you can use any three pins...its a tad slower but very flexible

```
// SOFTWARE SPI SETTINGS
#define BLUEFRUIT_SPI_SCK             13
#define BLUEFRUIT_SPI_MISO            12
#define BLUEFRUIT_SPI_MOSI            11
```

Refer to the table above to determine whether you have SPI or UART controlled Bluefruits!

Sélectionner le Bus Série

Once you've configured your pin setup in the BluefruitConfig.h file, you can now check and adapt the example sketch.

The Adafruit_BluefruitLE_nRF51 library supports four different serial bus options, depending on the HW you are using: **SPI** both hardware and software type, and **UART** both hardware and software type.

Carte de type UART (Bluefruit LE UART Friend & Flora BLE)

This is for Bluefruit LE UART Friend & Flora BLE boards. You can use either software serial or hardware serial. Hardware serial is higher quality, and less risky with respect to losing data. However, you may not have hardware serial available! Software serial does

work just fine with flow-control and we do have that available at the cost of a single GPIO pin.

For software serial (Arduino Uno, Adafruit Metro) you should uncomment the software serial contructor below, and make sure the other three options (hardware serial & SPI) are commented out.

```
// Create the bluefruit object, either software serial...uncomment these lines
SoftwareSerial bluefruitSS = SoftwareSerial(BLUEFRUIT_SWUART_TXD_PIN, BLUEFRUIT_SWUART_RXD_PIN);

Adafruit_BluefruitLE_UART ble(bluefruitSS, BLUEFRUIT_UART_MODE_PIN,
                      BLUEFRUIT_UART_CTS_PIN, BLUEFRUIT_UART_RTS_PIN);
```

For boards that require hardware serial (Adafruit Flora, etc.), uncomment the hardware serial constructor, and make sure the other three options are commented out

```
/* ...or hardware serial, which does not need the RTS/CTS pins. Uncomment this line */
Adafruit_BluefruitLE_UART ble(BLUEFRUIT_HWSERIAL_NAME, BLUEFRUIT_UART_MODE_PIN);
```

Carte de type SPI (Bluefruit LE SPI Friend)

For SPI based boards, you should uncomment the hardware SPI constructor below, making sure the other constructors are commented out:

```
/* ...hardware SPI, using SCK/MOSI/MISO hardware SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_CS, BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

If for some reason you can't use HW SPI, you can switch to software mode to bit-bang the SPI transfers via the following constructor:

```
/* ...software SPI, using SCK/MOSI/MISO user-defined SPI pins and then user selected CS/IRQ/RST */
Adafruit_BluefruitLE_SPI ble(BLUEFRUIT_SPI_SCK, BLUEFRUIT_SPI_MISO,
                             BLUEFRUIT_SPI_MOSI, BLUEFRUIT_SPI_CS,
                             BLUEFRUIT_SPI_IRQ, BLUEFRUIT_SPI_RST);
```

Commandes AT

Sommaire

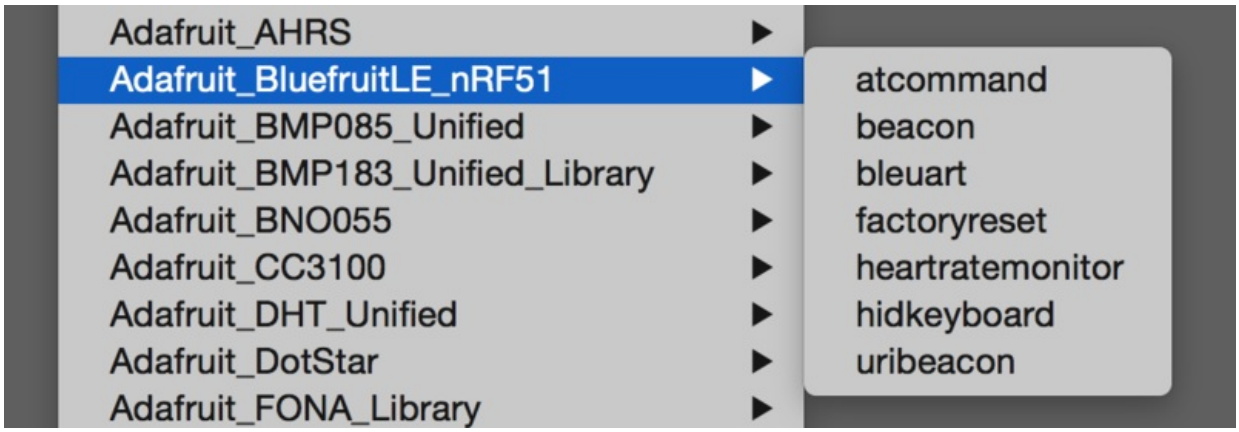
Croquis ATCommand

Le croquis d'exemple ATCommand vous permet d'envoyer des commandes AT vers le module. Ce croquis permet également de voir les résultats sur le moniteur série d'Arduino.

Cela peut être très utile pour faire du débogage, ou juste tester différentes commande pour voir comment cela fonctionne. C'est un excellent module pour débuter!

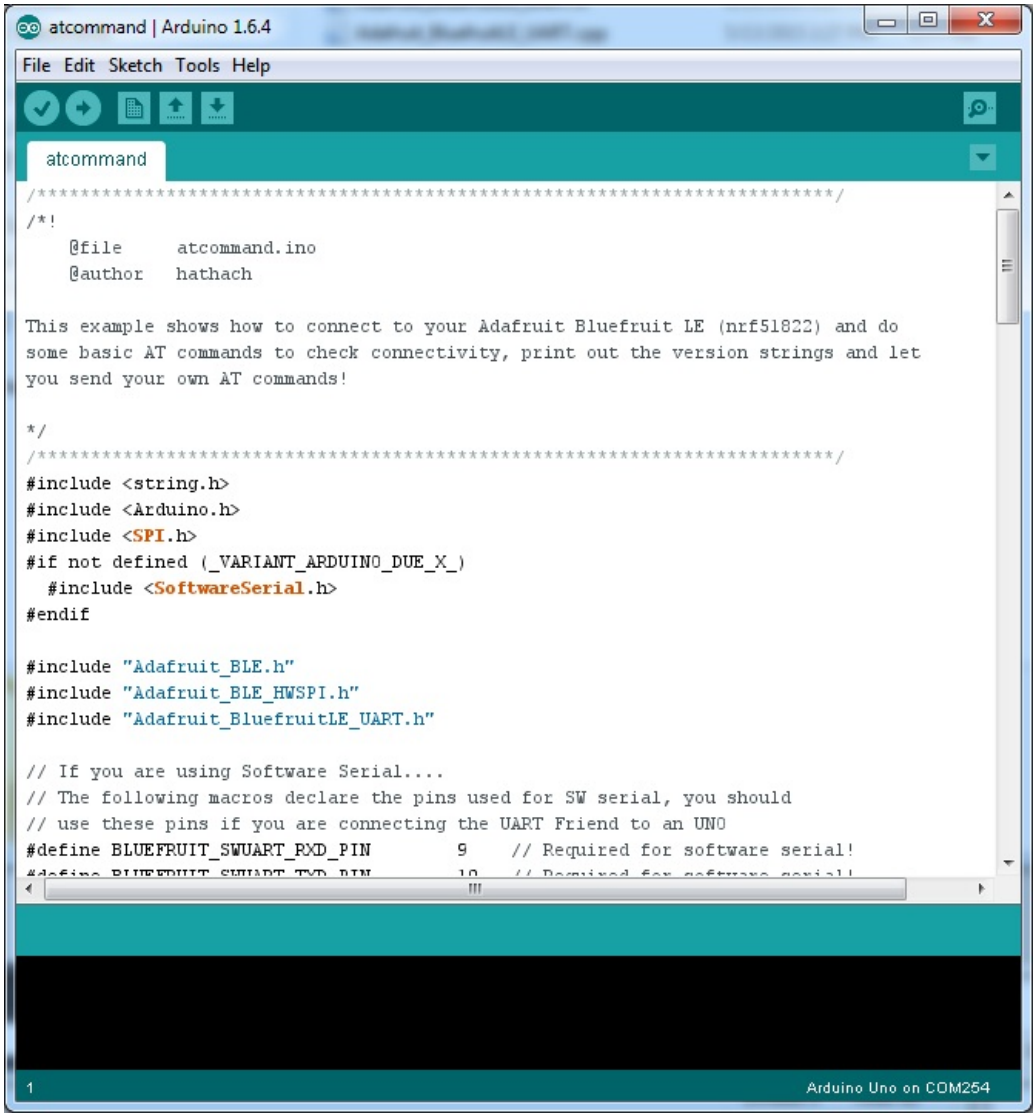
Ouvrir le croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **atcommand**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

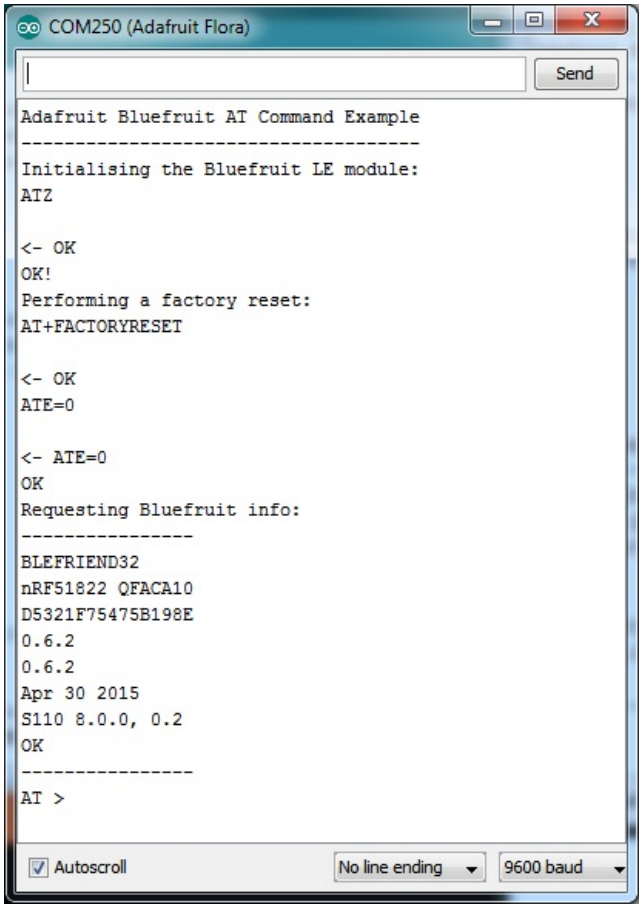
Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel

Si vous utilisez le port série/UART logiciel ou matériel:

- Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.
- N'oubliez pas de **connecter la broche CTS sur la masse/GND du Bluefruit si vous n'utilisez pas le signal CTS!** (Le Flora l'a déjà branché à la masse)

Exécuter le croquis

Une fois le croquis/sketch téléversé sur votre carte (à l'aide de l'icone de téléversement en forme de flèche), vous pouvez ouvrir un moniteur série à l'aide du menu **Outil > Moniteur Série** et fixer le débit (baud rate, en bas à droite) sur **115200**:

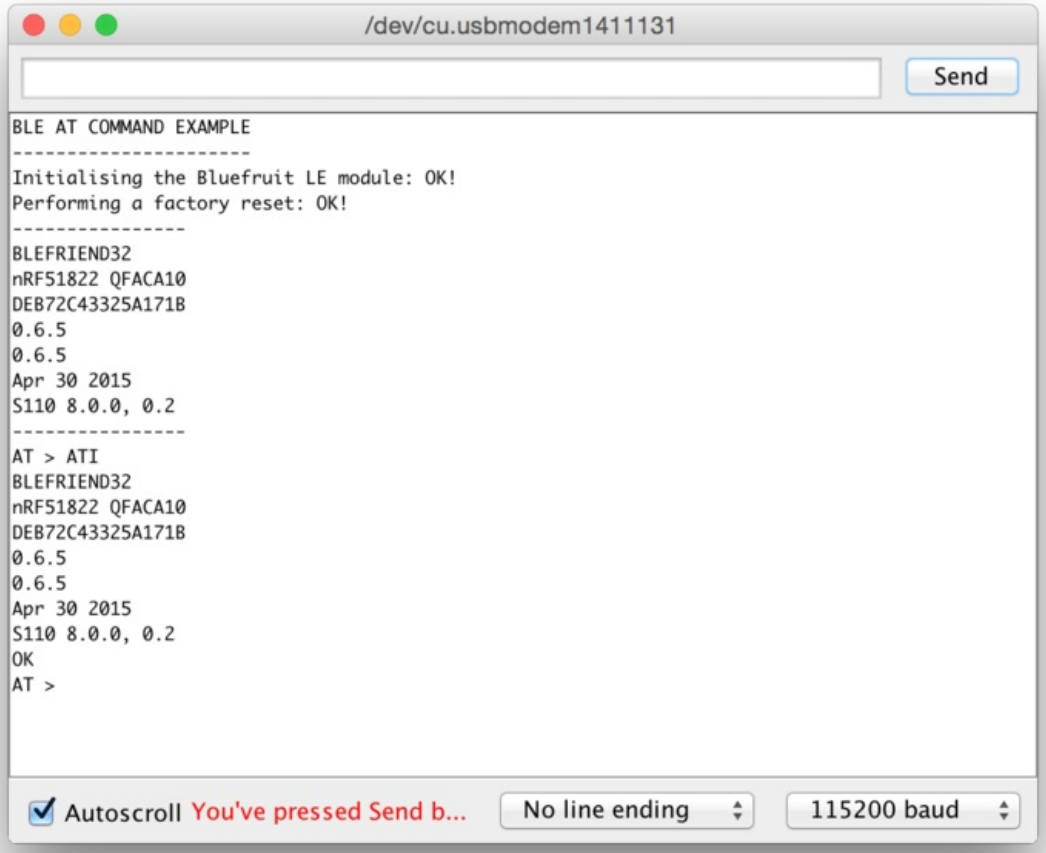


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Pour envoyer une commande AT sur le module Bluefruit LE: saisissez la commande dans le zone de texte en haut du moniteur série puis cliquez sur le bouton "Envoyer" (*Send*)

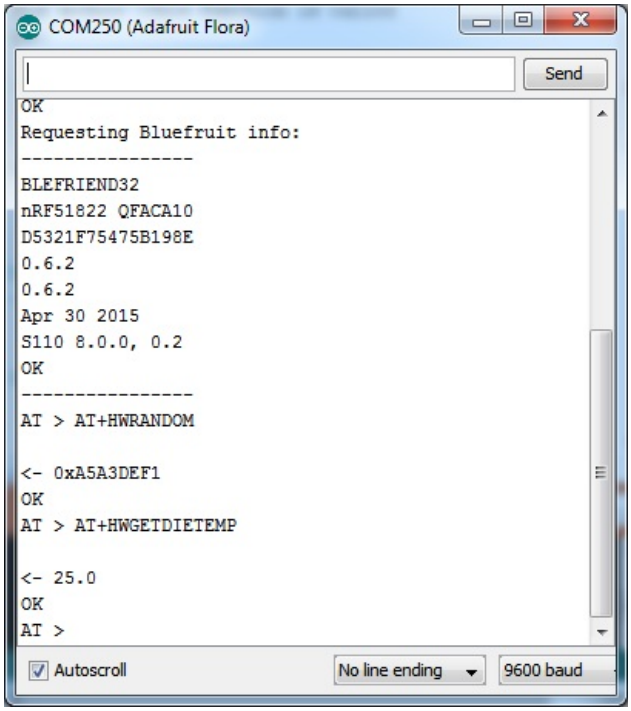
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

La réponse à la commande AT sera affichée dans la partie principale du Moniteur Série. La réponse à la commande "**ATI**" est visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Vous pouvez saisir n'importe quelle commande AT sur cette invite de commande. Essayez **AT+HELP** pour obtenir la liste de toutes les commande et essayer une commande telle que **AT+HWGETDIETEMP** (obtenir la température interne du nRF51822) ou **AT+HWRANDOM** (générer un nombre aléatoire)



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

BLE Uart

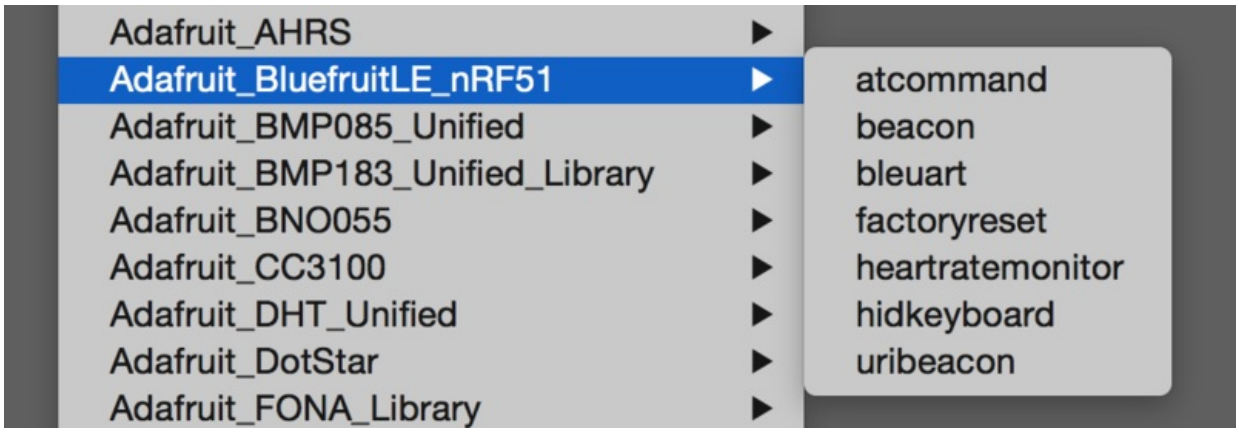
Sommaire

Croquis BLEUart

L'exemple **BLEUart** permet d'échanger des données texte (envoi et réception) entre un Arduino et un périphérique Bluetooth Low Energy connecté à l'autre bout (comme un téléphone mobile utilisant l'application **Adafruit Bluefruit LE Connect** pour Android https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect ou iOS https://itunes.apple.com/app/adafruit-bluefruit-le-connect/id830125974?mt=8 . Le transfert se fait en mode UART).

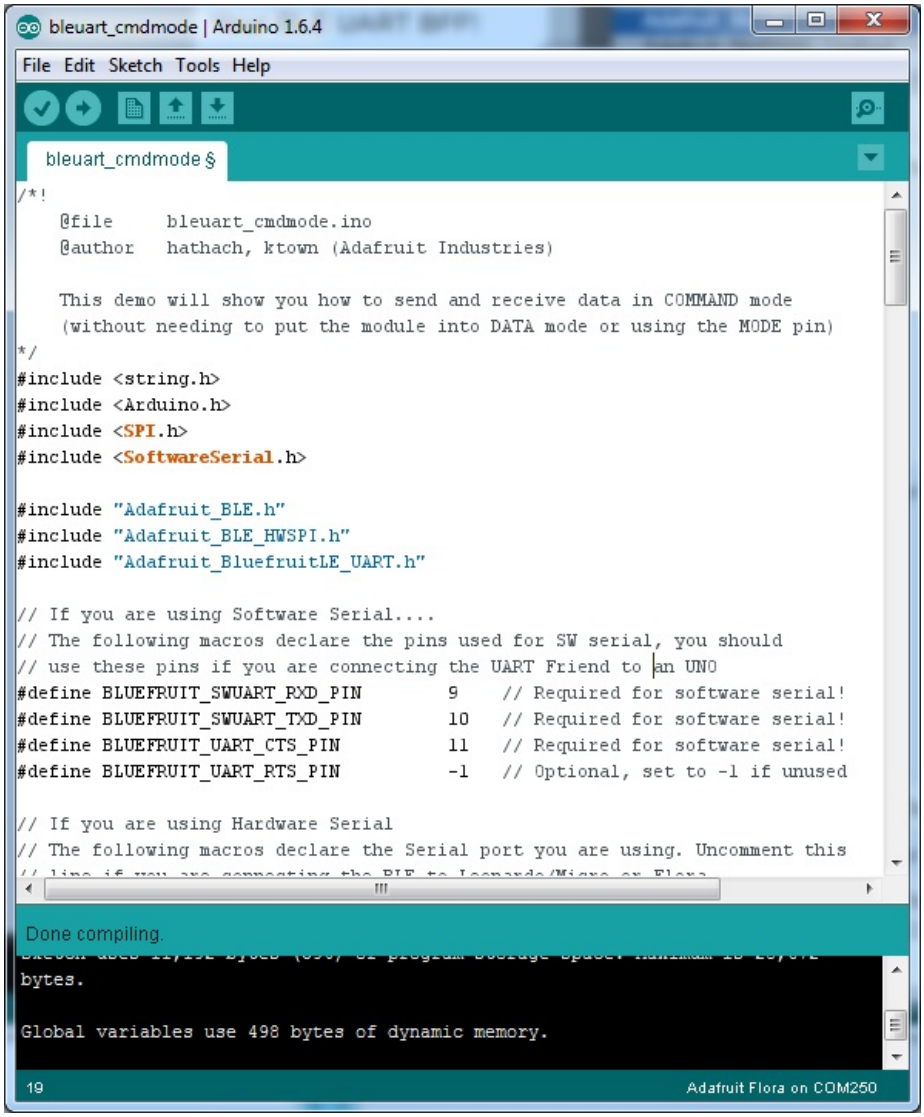
Ouvrir le croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **bleuart_cmdmode**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

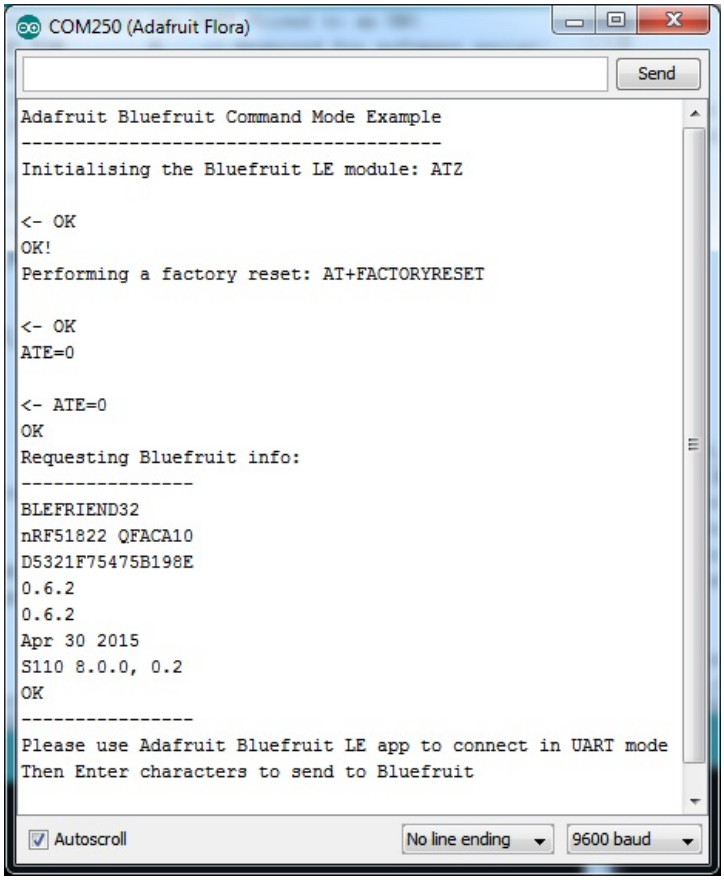
Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel

Si vous utilisez le port série/UART logiciel ou matériel:

- Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.
- N'oubliez pas de **connecter la broche CTS sur la masse/GND du Bluefruit si vous n'utilisez pas le signal CTS!** (Le Flora l'a déjà branché à la masse)

Exécuter le croquis

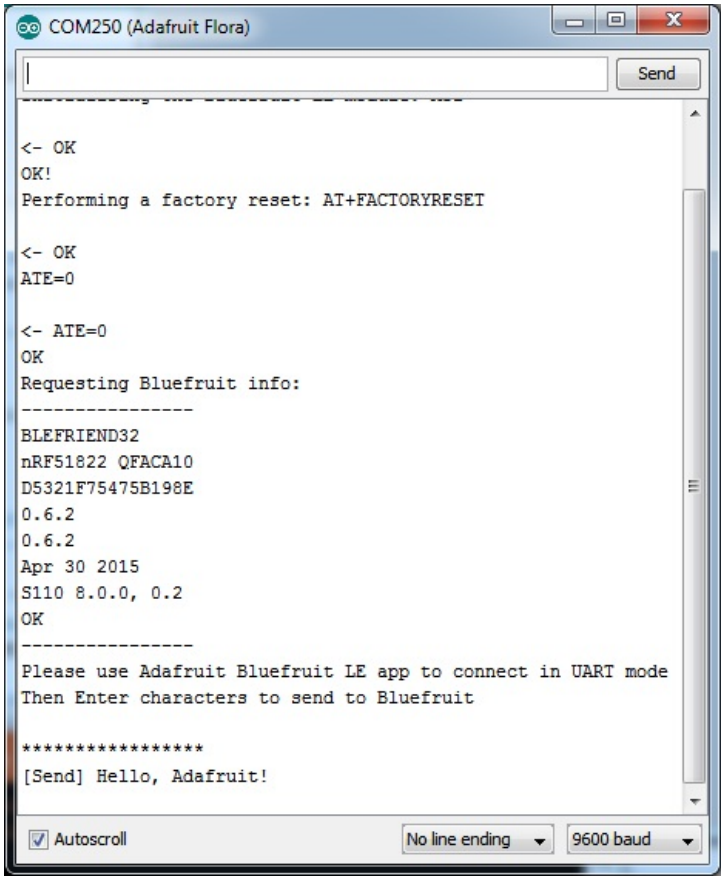
Une fois le croquis téléversé sur votre carte Arduino, vous pouvez ouvrir le moniteur série via le menu **Outils > Moniteur série**. Assurez-vous que le débit (Baud rate) soit configuré sur **115200** bauds (en bas à droite):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

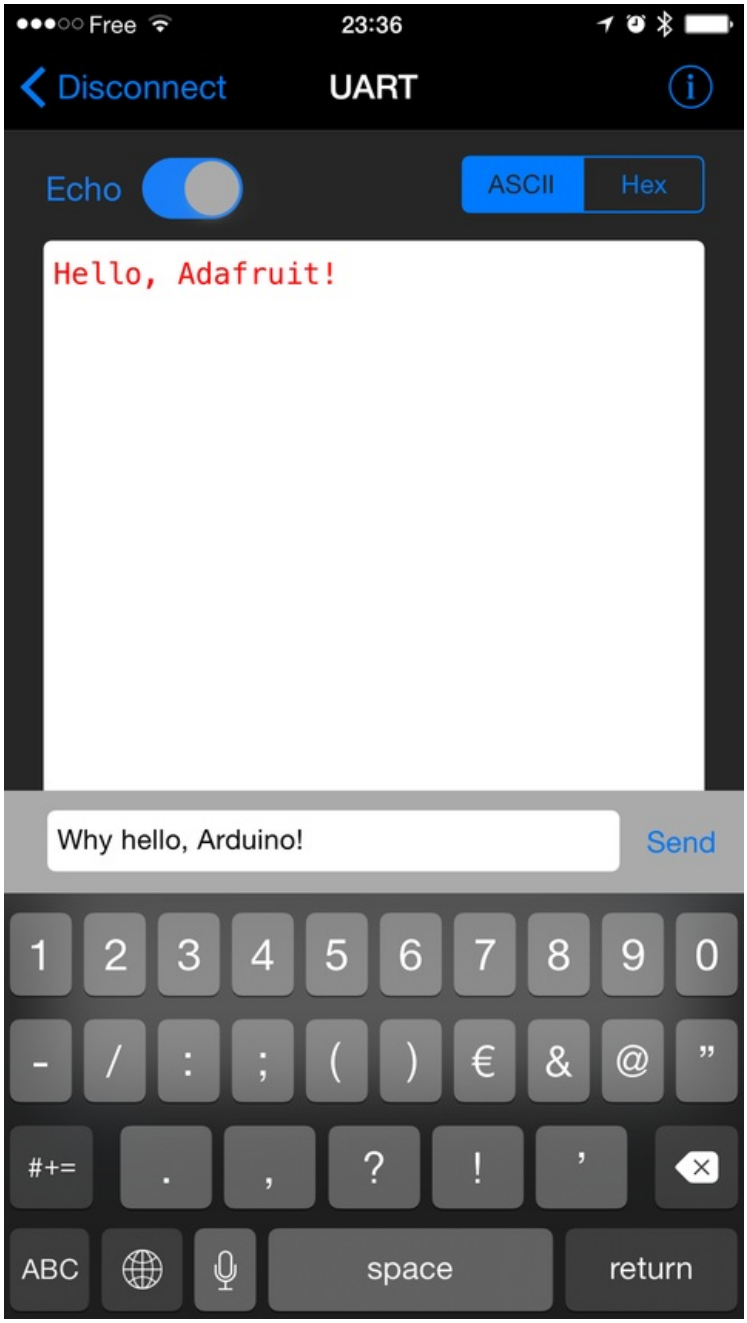
Une fois que vous pouvez voir les requêtes envoyées au module BleFruit, utiliser l'App Adafruit sur votre smartphone pour vous connecter sur le module Bluefruit LE en mode **UART** (vous obtiendrez donc une zone texte sur votre téléphone.

Tout le texte que vous saisissez dans la zone de saisie du moniteur série (la zone en haut) sera envoyé vers le smartphone... et toutes les donnée envoyé depuis le smarphone sera affiché dans le moniteur série:



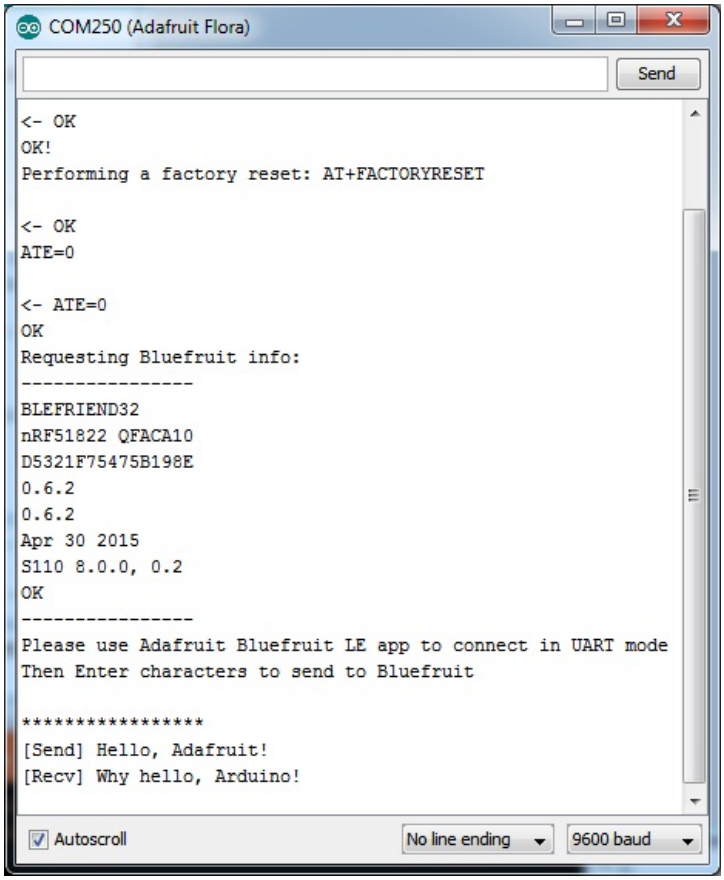
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Vous pouvez voir, ci-dessous, les chaînes de caractères arrivant sur l'application "Adafruit Bluefruit LE Connect" (App iOS en l'occurence):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Le texte de réponse ('Why hello, Arduino!') renvoyé depuis le Smartphone est visible dans le moniteur série Arduino (voir ci-dessous):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Clavier HID

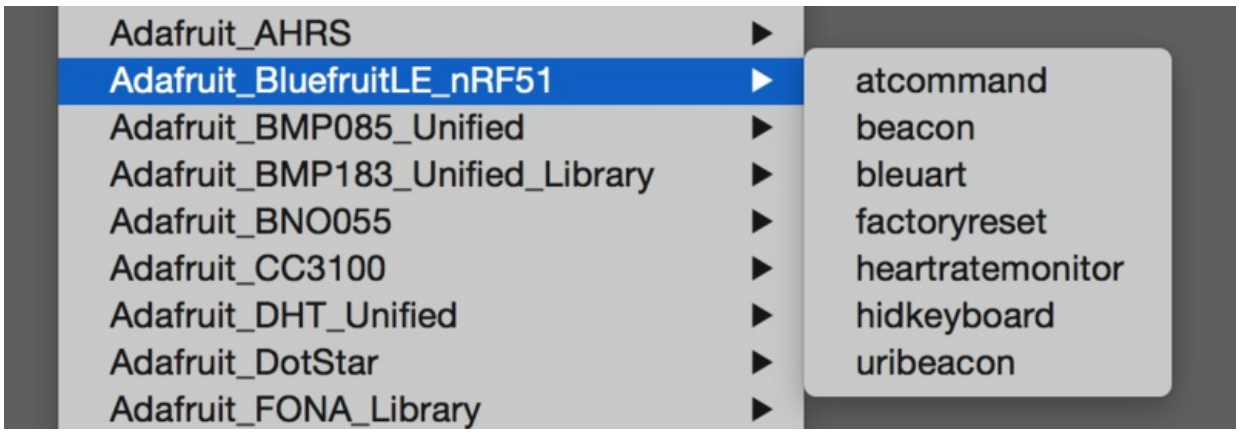
Sommaire

Le croquis HIDKeyboard

L'exemple **HIDKeyboard** vous montre comment utiliser les commandes AT pour la fonctionnalité HID keyboard afin d'envoyer des données clavier vers n'importe quel périphérique Android ou iOS (supportant BLE). Cela fonctionne également avec les autres périphériques supportant la spécification "BLE HID peripherals".

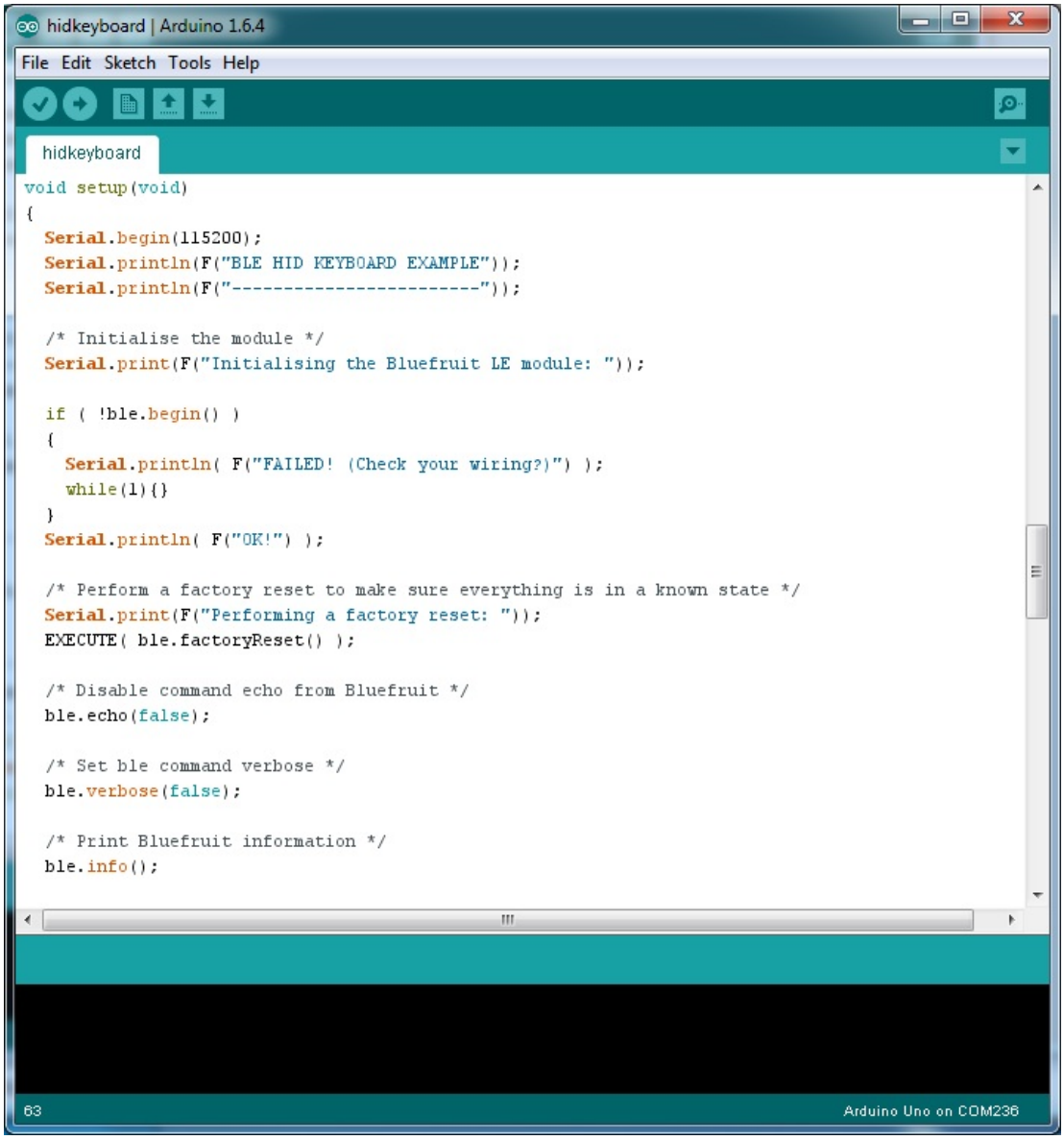
Ouvrir le croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **hidkeyboard**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

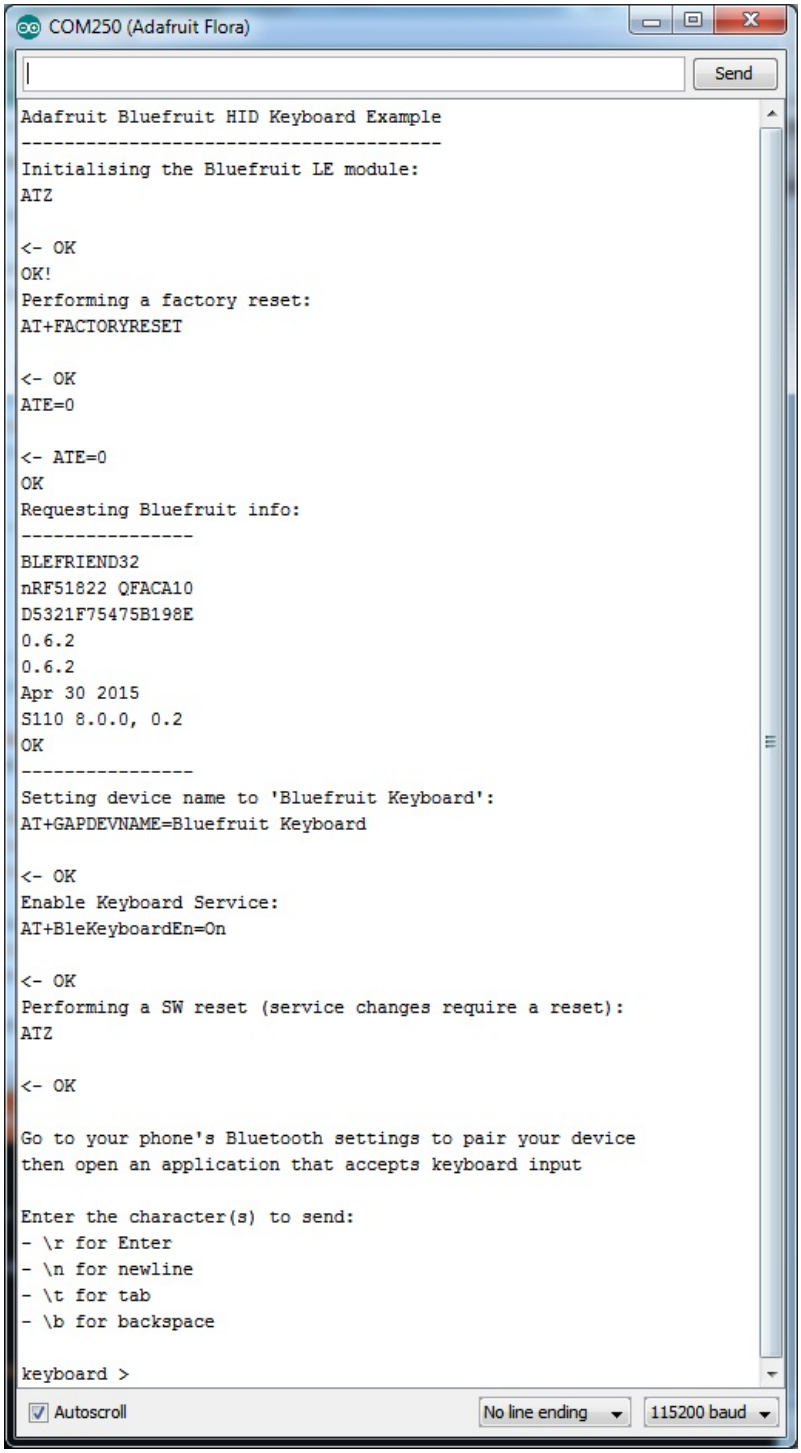
Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel

Si vous utilisez le port série/UART logiciel ou matériel:

- Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.
- N'oubliez pas de **connecter la broche CTS sur la masse/GND du Bluefruit si vous n'utilisez pas le signal CTS!** (Le Flora l'a déjà branché à la masse)

Exécuter le croquis

Une fois le croquis téléversé sur votre carte Arduino, vous pouvez ouvrir le moniteur série via le menu **Outils > Moniteur série**. Assurez-vous que le débit (Baud rate) soit configuré sur **115200** bauds (en bas à droite):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Lier le clavier HID

Avant de pouvoir utiliser le clavier "HID keyboard", il sera nécessaire de le lié/l'appairer avec votre téléphone ou PC. Le mécanisme d'appairage (*bonding* en anglais) permet d'établir une connexion permanente entre les deux périphérique. Cela signifie que dès que votre téléphone ou PC redétecte ce module Bluefruit LE module bien précis alors il est connecté automatiquement.

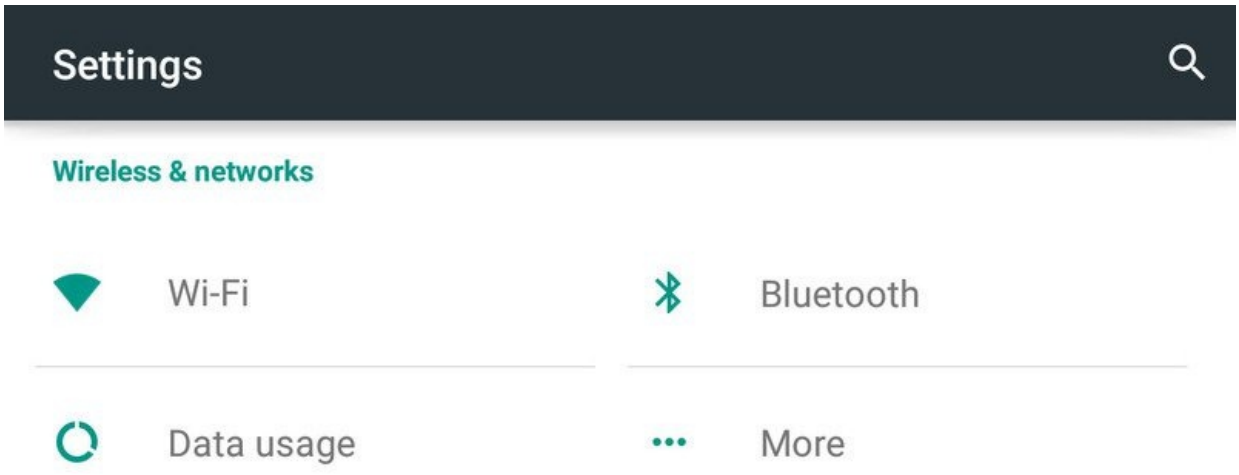
La procédure exacte à utilisé pour faire l'appairage varie d'une plateforme à l'autre.

> ***Lorsque vous n'avez plus besoin de l'appairage d'un module, ou si vous désirez re-appairer le périphérique Bluefruit LE sur un autre appareil, alors il faut effacer les informations d'appairage stocké sur le téléphone ou ordinateu... sinon il ne sera pas possible de le connecter avec un nouveau périphérique!***

Android

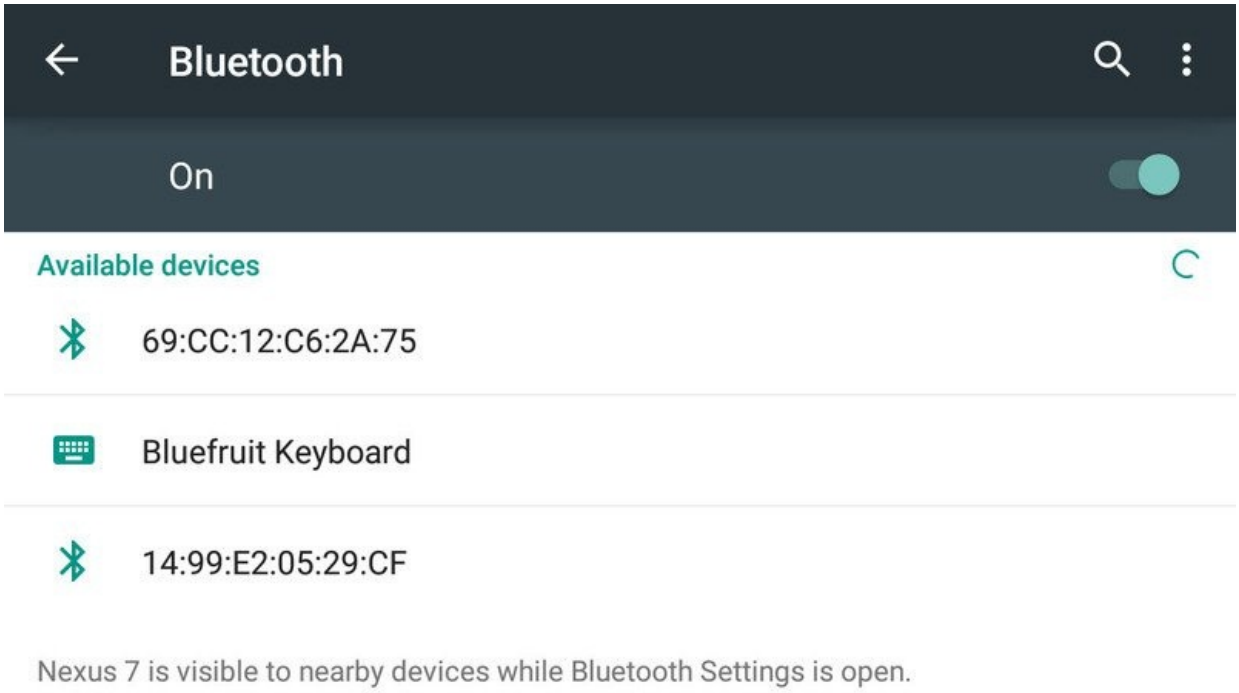
Pour lier le clavier (*keyboard*) sur un périphérique Android compatible BLE, rendez-vous dans l'application **Paramètres** (*Settings* en anglais) et cliquez sur l'icon **Bluetooth**.

Ces captures d'écran son basées sur un Android 5.0 fonctionnant sur un Nexus 7 2013. L'apparence peut varier d'un périphérique à l'autre et d'une version de l'OS à l'autre.



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Dans le volet de configuration Bluetooth vous pouvez voir la liste des périphériques disponibles. Vous devriez y voir le module Bluefruit LE s'annonçant sous le nom **Bluefruit Keyboard**:

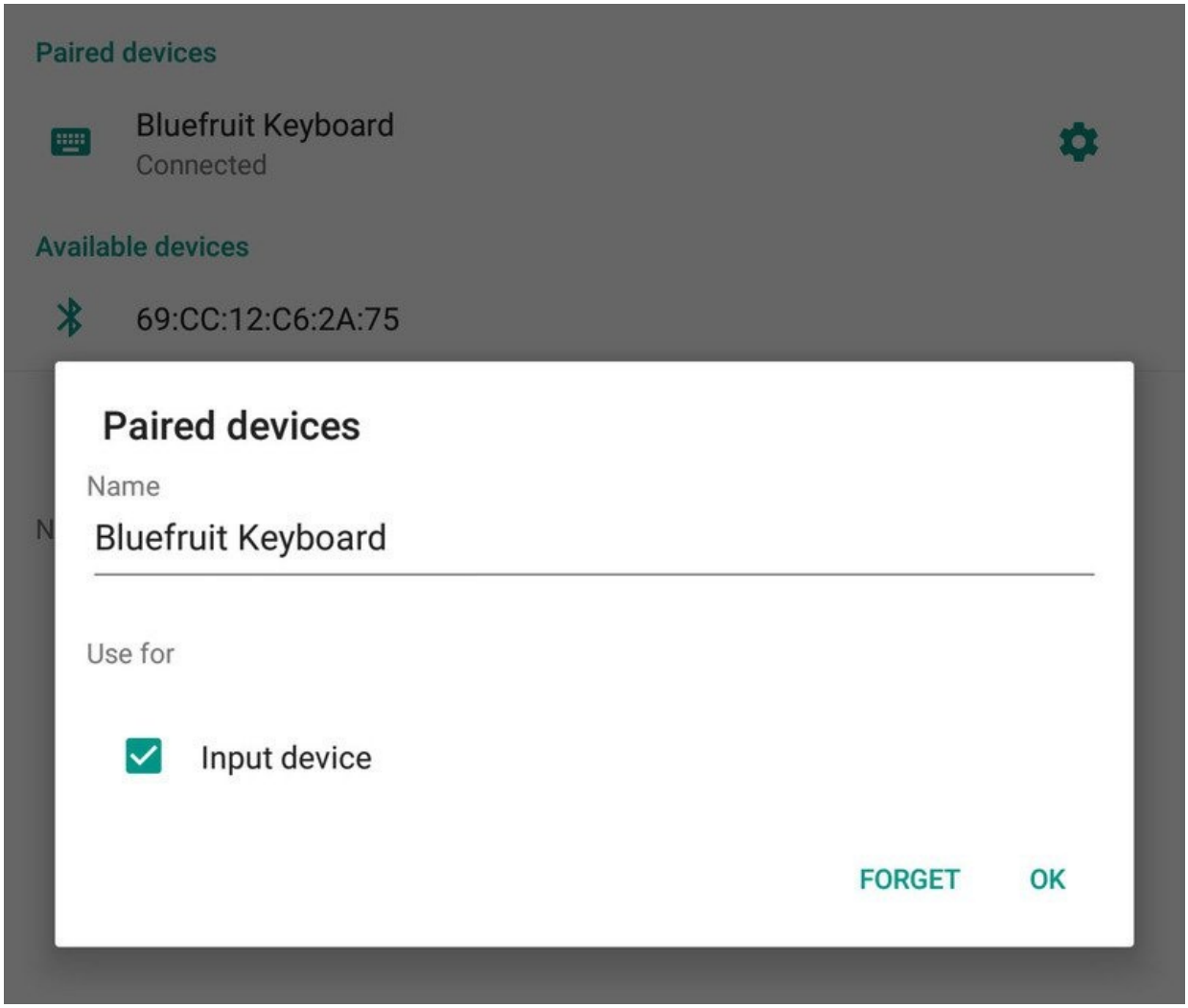


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Sélectionner le périphérique permet de démarrer le processus d'appairage. Processus qui se termine par le placement du périphérique "Bluefruit Keyboard" dans la liste des produits appairés (*Paired devices* en anglais). Le libellé sous le périphérique devrait indiqué 'Connecté' (ou *Connected* en anglais):

Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Pour effacer les information d'appairage, il faut cliquer sur l'icon en forme de roue dentée en regard du périphérique puis cliquer sur le bouton "Abandonner/Relâcher" (*Forget* en anglais):

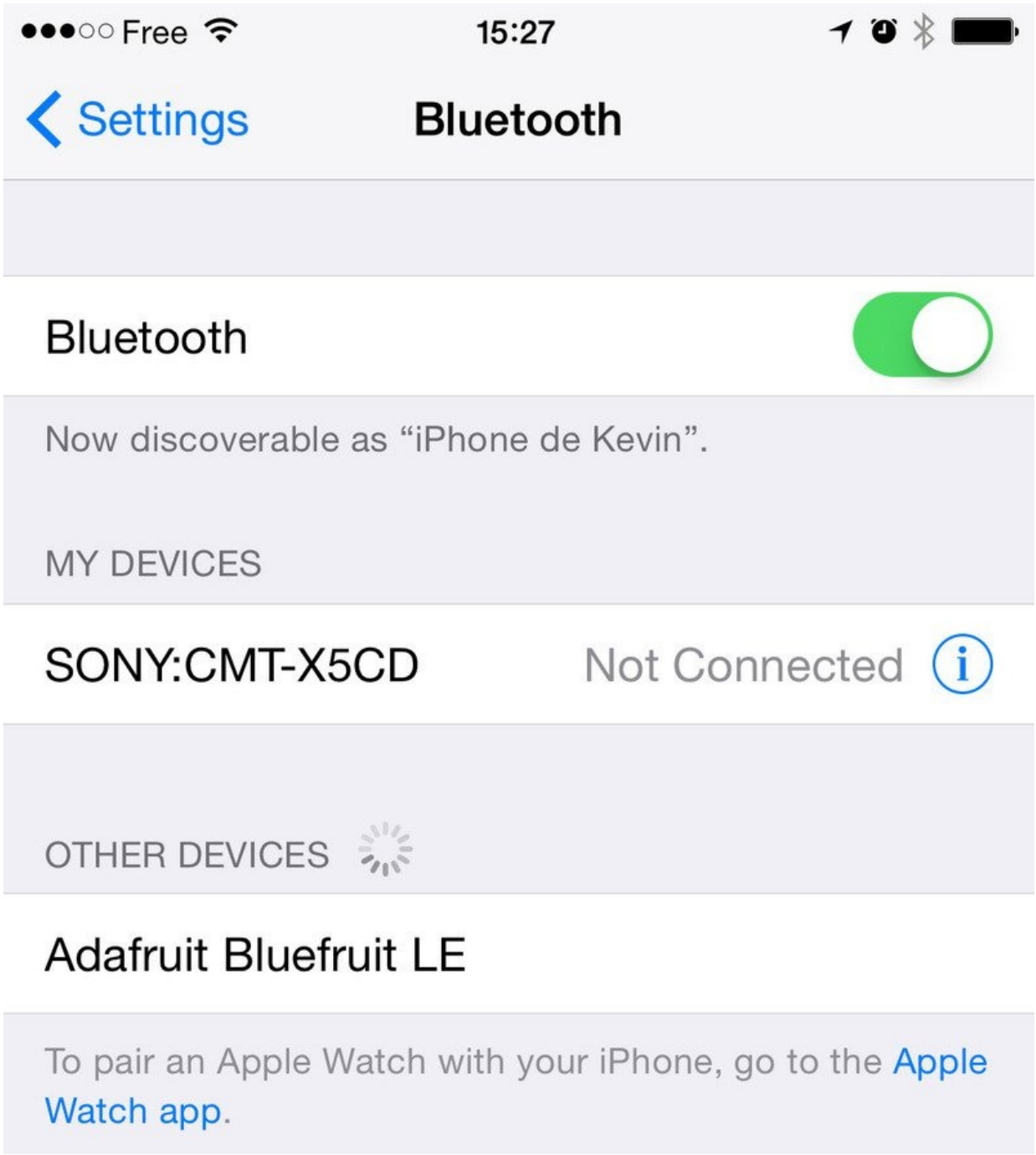


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

iOS

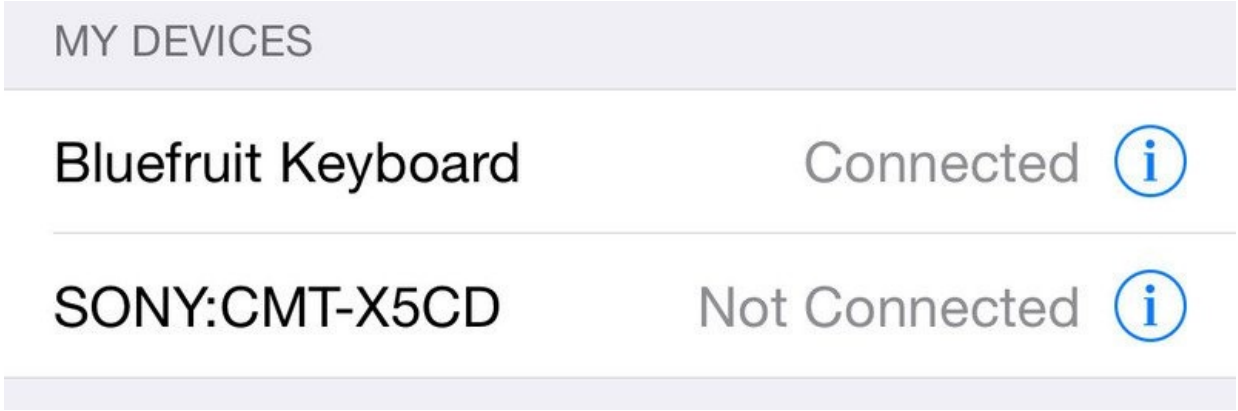
Pour lier le "clavier" sur un périphérique iOS, rendez-vous dans l'application **Configuration'** ***('Settings*** en anglais) de votre téléphone et cliquer sur le point de menu **Bluetooth**.

Le clavier devrait apparaître sous la liste **AUTRES PERIPHERIQUES** (*OTHER DEVICES* en anglais):



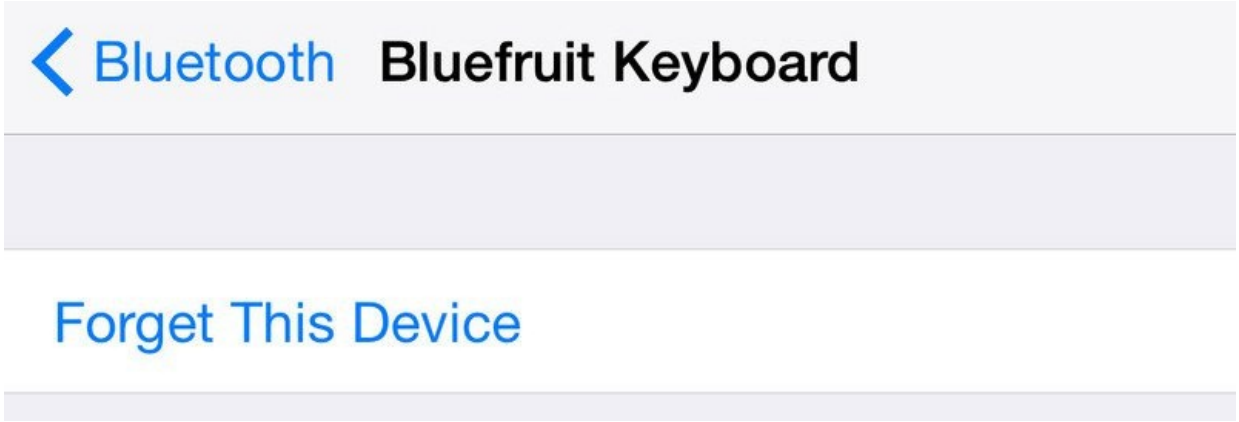
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Une fois le processus d'appairage terminé, le périphérique est déplacé dans la catégorie **MES PERIPHERIQUES** (*MY DEVICES* en anglais). Il sera alors possible de commencer à utiliser le modul e Bluefruit LE comme un clavier:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

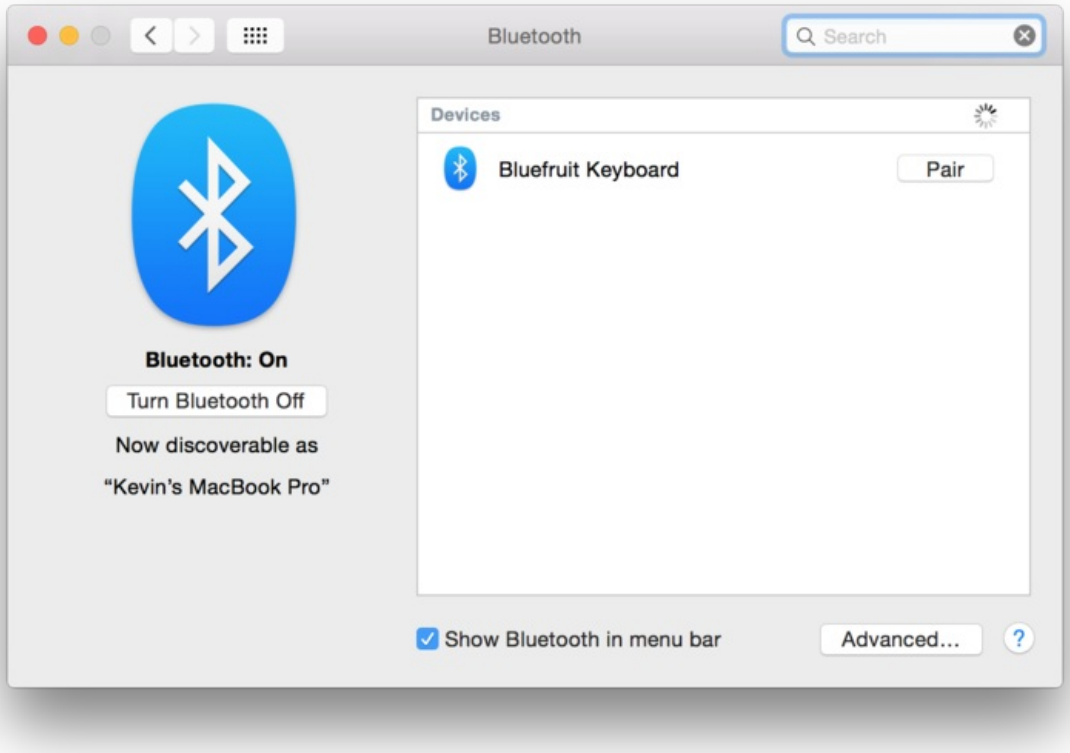
Pour abandonner l'appairage, cliquez sur l'icone 'info' et sélectionnez ensuite l'option **Abandonner ce périphérique** (*Forget this Device* en anglais).



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

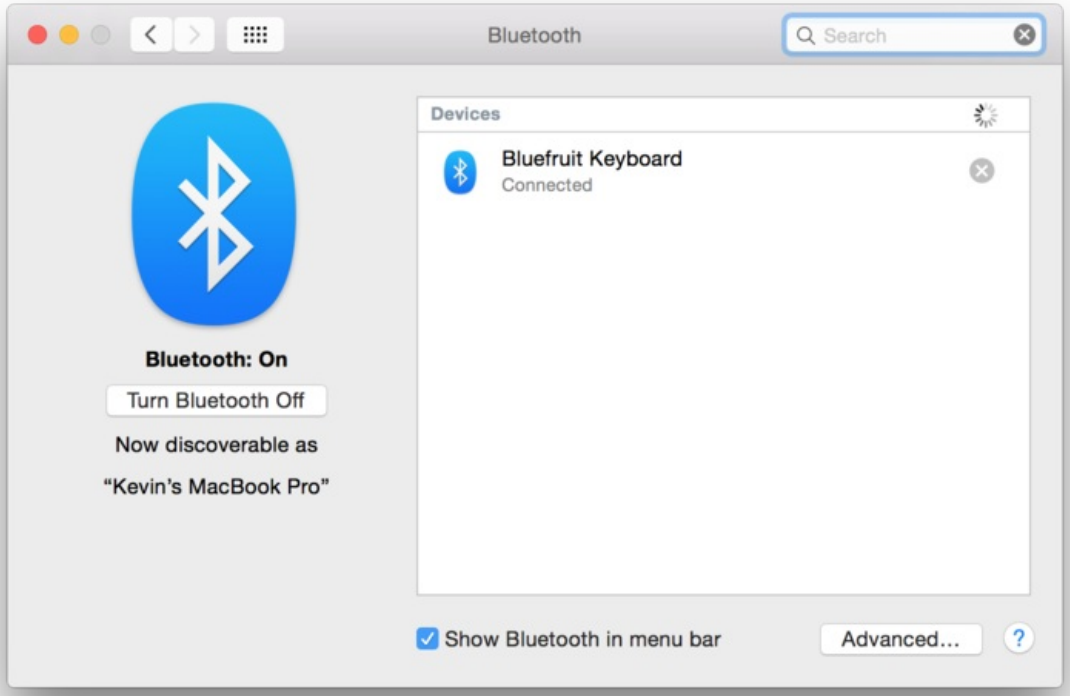
OS X

Pour appairer le clavier sur un périphérique OS X device, rendez vous dans la fenêtre des **Préférence Bluetooth** et cliquez sur le bouton **Appairer** (ou *Pair*) en face du périphérique **Bluefruit Keyboard** (tel qu'il est nommé dans le croquis d'exemple):



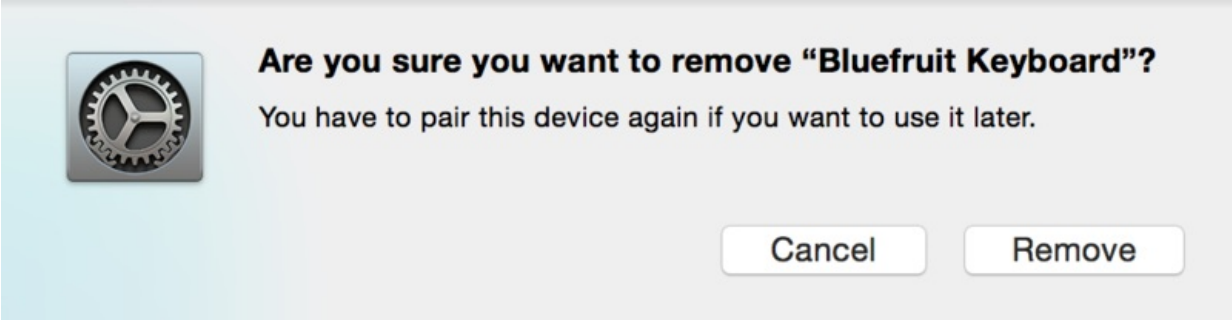
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Une fois appairé, il est possible d'abandonner l'appairage en cliquant sur l'icone 'x' situe à côté du périphérique **Bluefruit Keyboard**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

... et cliquer ensuite sur le bouton **Enlever** (*Remove* en anglais) lorsque la fenêtre de confirmation apparaît:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Contrôleur

Sommaire

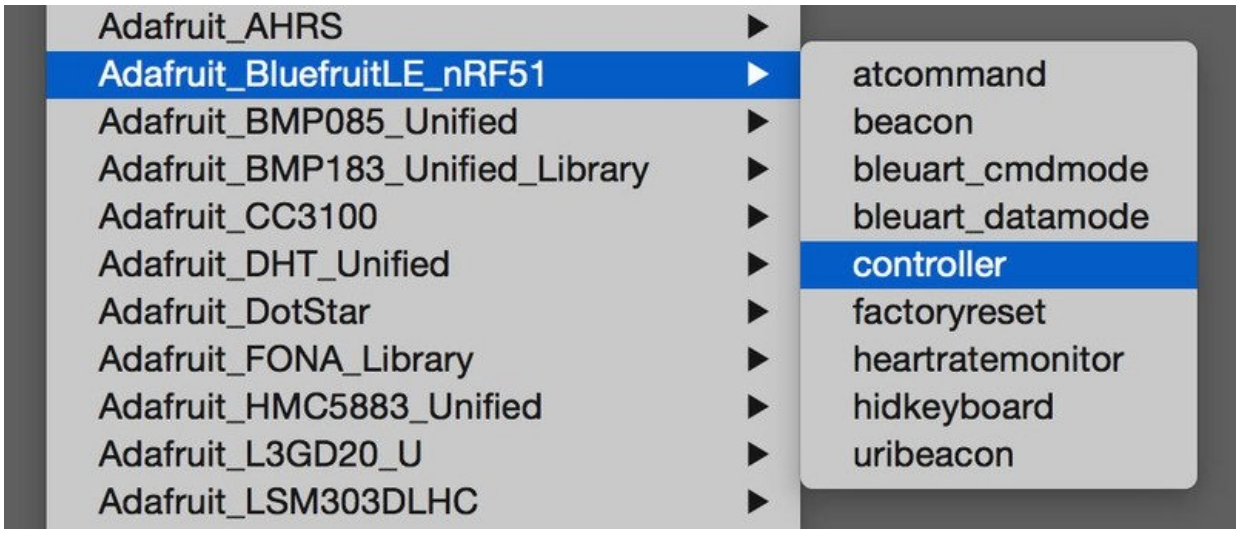
Le croquis Controler

Le croquis d'exemple **Controller** vous permet de transformer votre périphérique iOS ou Android (supportant BLE) en un contrôleur distant ou source de donnée externe. Cela vous permet de profiter des avantages offerts par les senseurs de votre téléphone ou tablette.

Vous pouvez acquérir les données accélérométrique (ou données quaternion https://en.wikipedia.org/wiki/Quaternion Wikipedia) de votre téléphone pour les pousser vers votre Arduino via BLE. Vous pouvez également obtenir les dernière données GPS de votre mobile de la même façon (sans devoir acheter ou alimenter du matériel complémentaire).

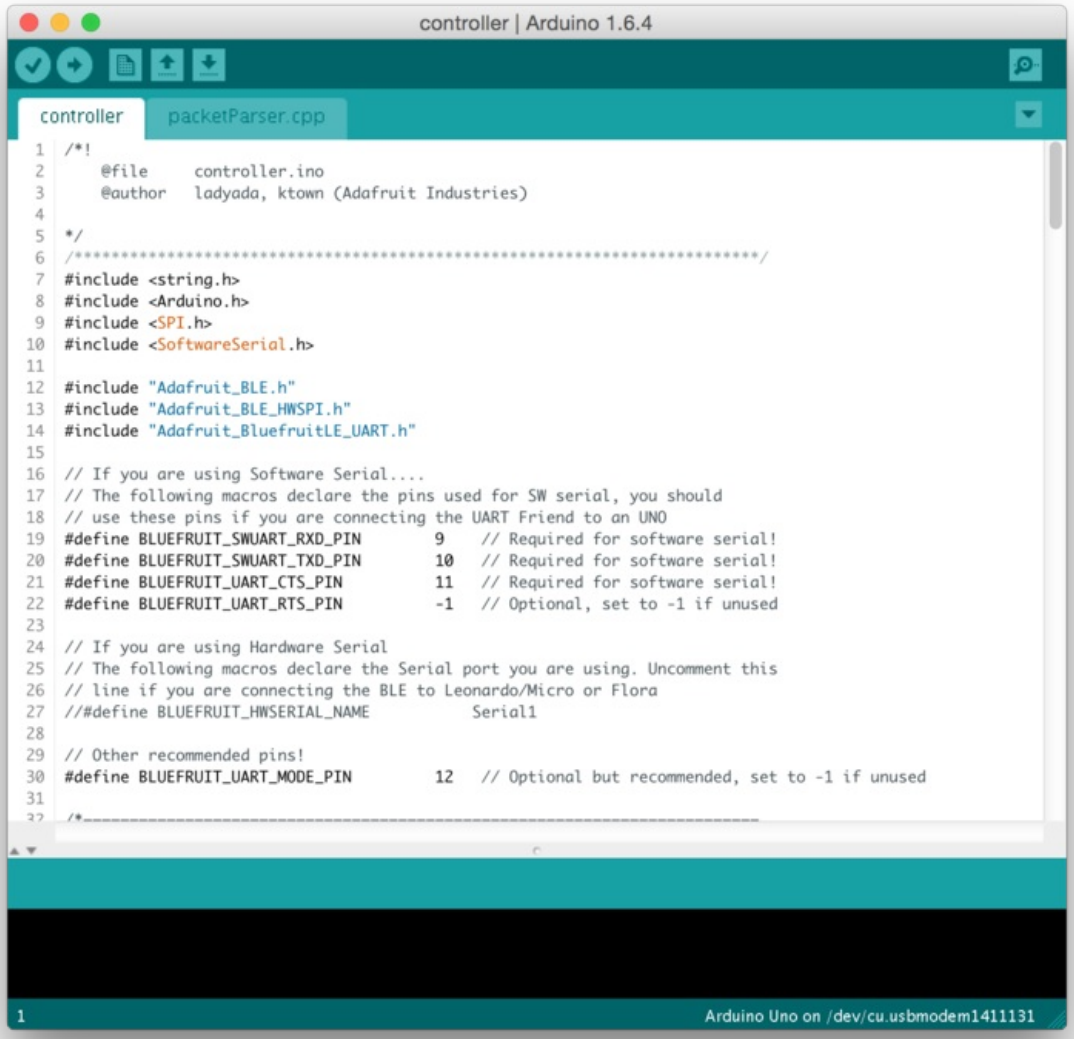
Ouvrir le croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **controller**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

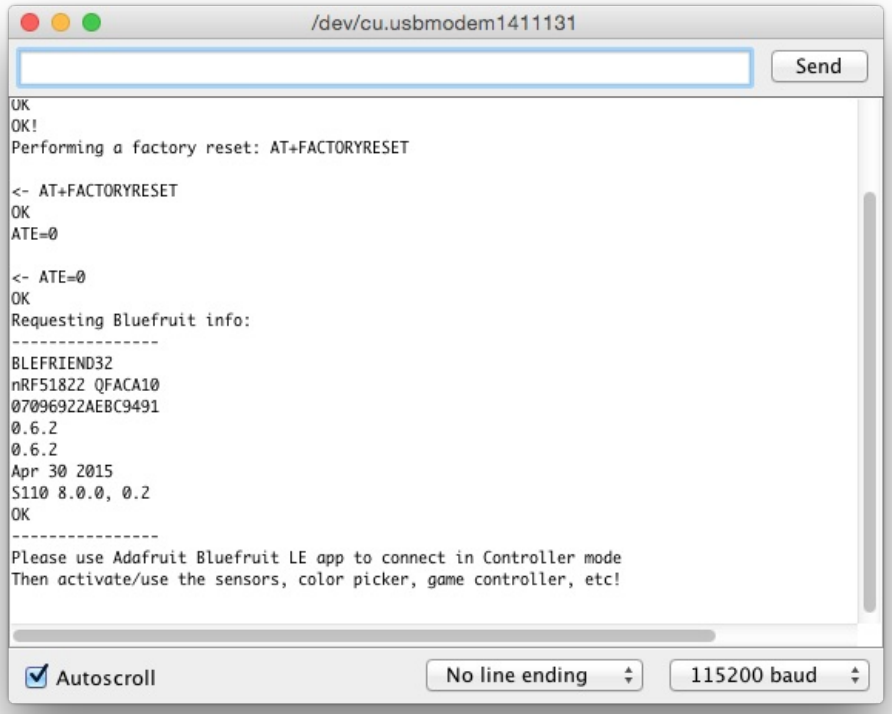
Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel

Si vous utilisez le port série/UART logiciel ou matériel:

- Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.
- N'oubliez pas de **connecter la broche CTS sur la masse/GND du Bluefruit si vous n'utilisez pas le signal CTS!** (Le Flora l'a déjà branché à la masse)

Exécuter le croquis

Une fois le croquis téléversé sur votre carte Arduino, vous pouvez ouvrir le moniteur série via le menu **Outils > Moniteur série**. Assurez-vous que le débit (Baud rate) soit configuré sur **115200** bauds (en bas à droite):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Une fois le croquis en cours d'exécution sur votre Arduino, vous pouvez démarrer l'application "BlueFruit LE Connect".

Si c'est la première fois que vous utilisez "Bluefruit LE Connect", prenez le temps de consulter le guide d'apprentissage "Bluefruit LE Connect" https://learn.adafruit.com/bluefruit-le-connect-for-ios/settings (*Adafruit, anglais*)

Utiliser Bluefruit LE Connect en mode contrôleur

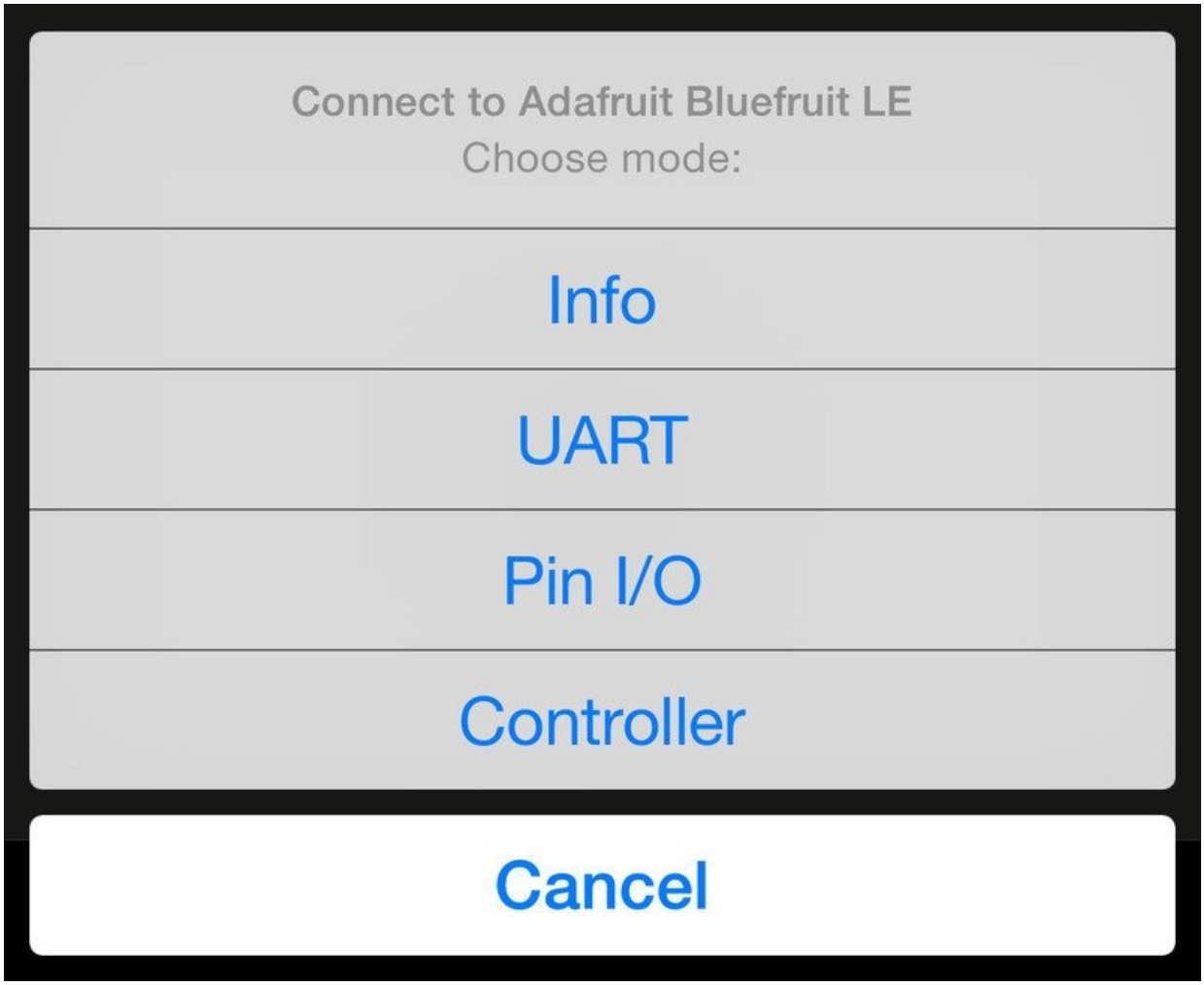
Une fois l'application "Bluefruit LE Connect" démarrée sur votre SmartPhone, sélectionnez le périphérique BlueFruit LE (parmi la liste des périphériques disponible) sur l'écran d'accueil:

Vous pourrez ensuite sélectionner l'activité "**controller**". Cette activité interagit avec le croquis que nous venons de téléverser sur notre Arduino.

Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

L'application affichera une liste des "sources de données" que vous pourrez collecter sur votre smartphone/tablette pour, ensuite, les envoyez vers votre module Bluefruit LE.

Activez/désactivez simplement les senseurs qui vous intéressent.

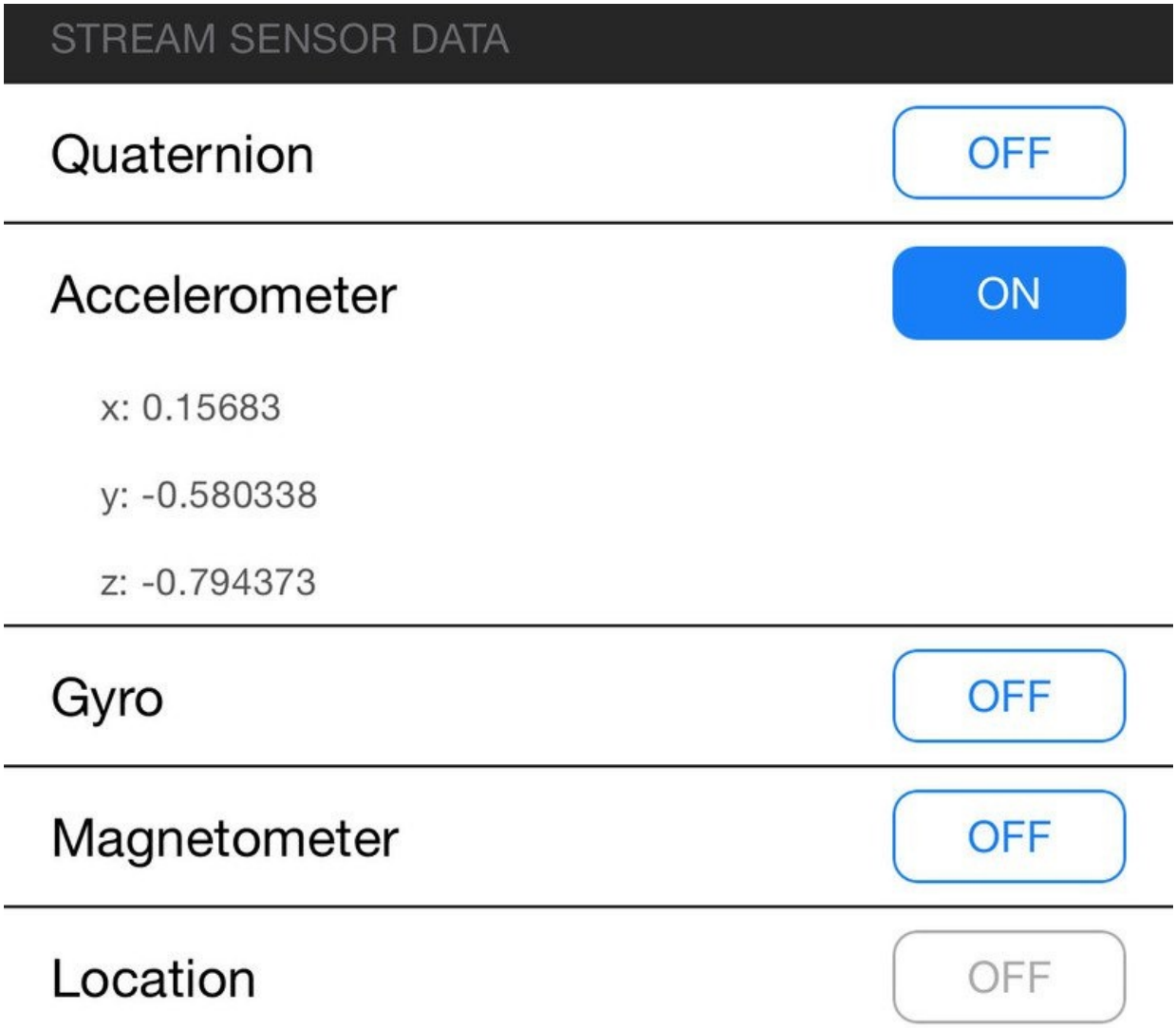


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Streamer les données des senseurs

Vous pouvez selection les données Quaternion (orientation absolute), Accélérometre, Gyroscope, Magnétomètre ou localisation depuis votre SmartPhone et les envoyer directement vers votre Arduino.

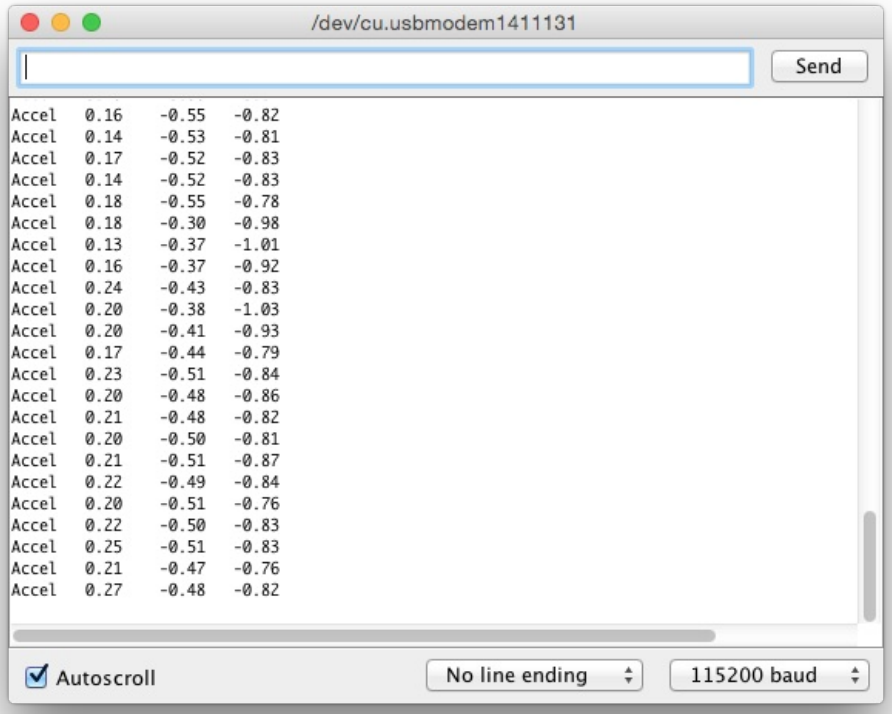
En activant le champ **Accéléromètre**, par exemple, vous pouvez voir les données (mises-à-jour) dans l'app:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Les données sont traitées (parsing) dans le croquis d'exemple et affichés dans le moniteur série:

```
Accel 0.20 -0.51 -0.76
Accel 0.22 -0.50 -0.83
Accel 0.25 -0.51 -0.83
Accel 0.21 -0.47 -0.76
Accel 0.27 -0.48 -0.82
```

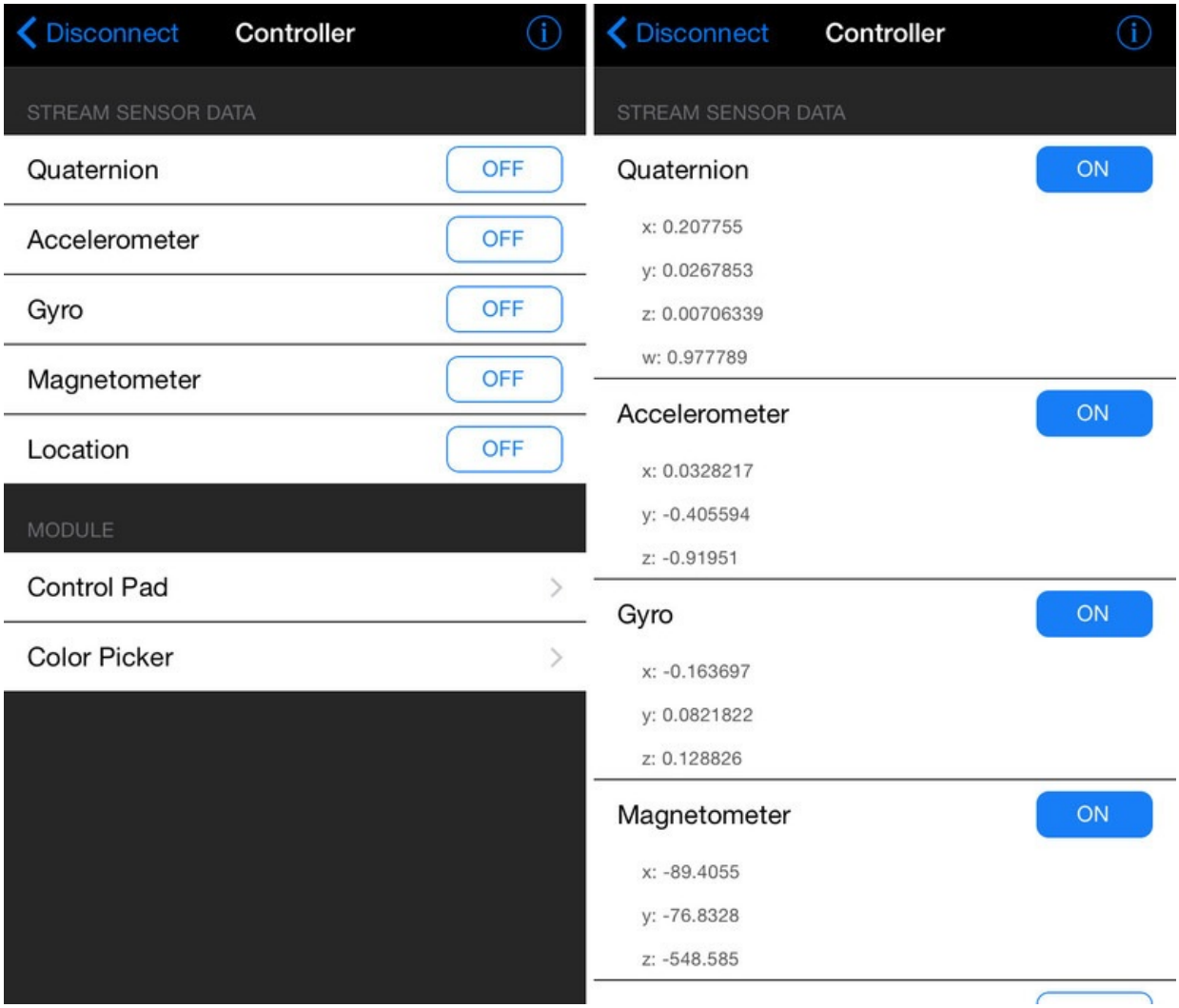


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Note que même si le croquis n'affiche que 2 decimales, les valeurs envoyées par l'App sont des valeurs en virgule flottante codées sur 4 octets (*4-byte floating point value*).

Le pavé de commande

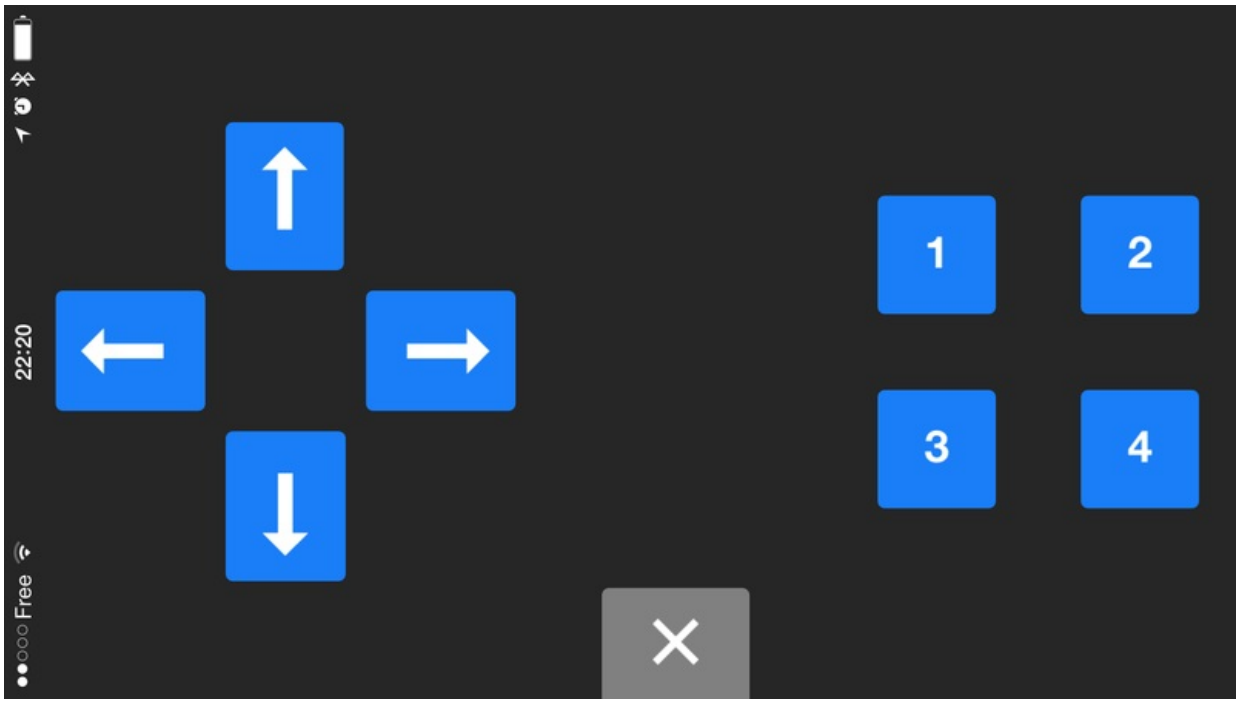
Vous pouvez également utiliser le *module Control Pad* pour capturer les bouton pressé (et relâché) :



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Ce module spécial affiche un "Game Pad" comme celui présenté ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Les boutons pressés et relâchés apparaitrons dans le moniteur série (avec l'ID du bouton utilisé):

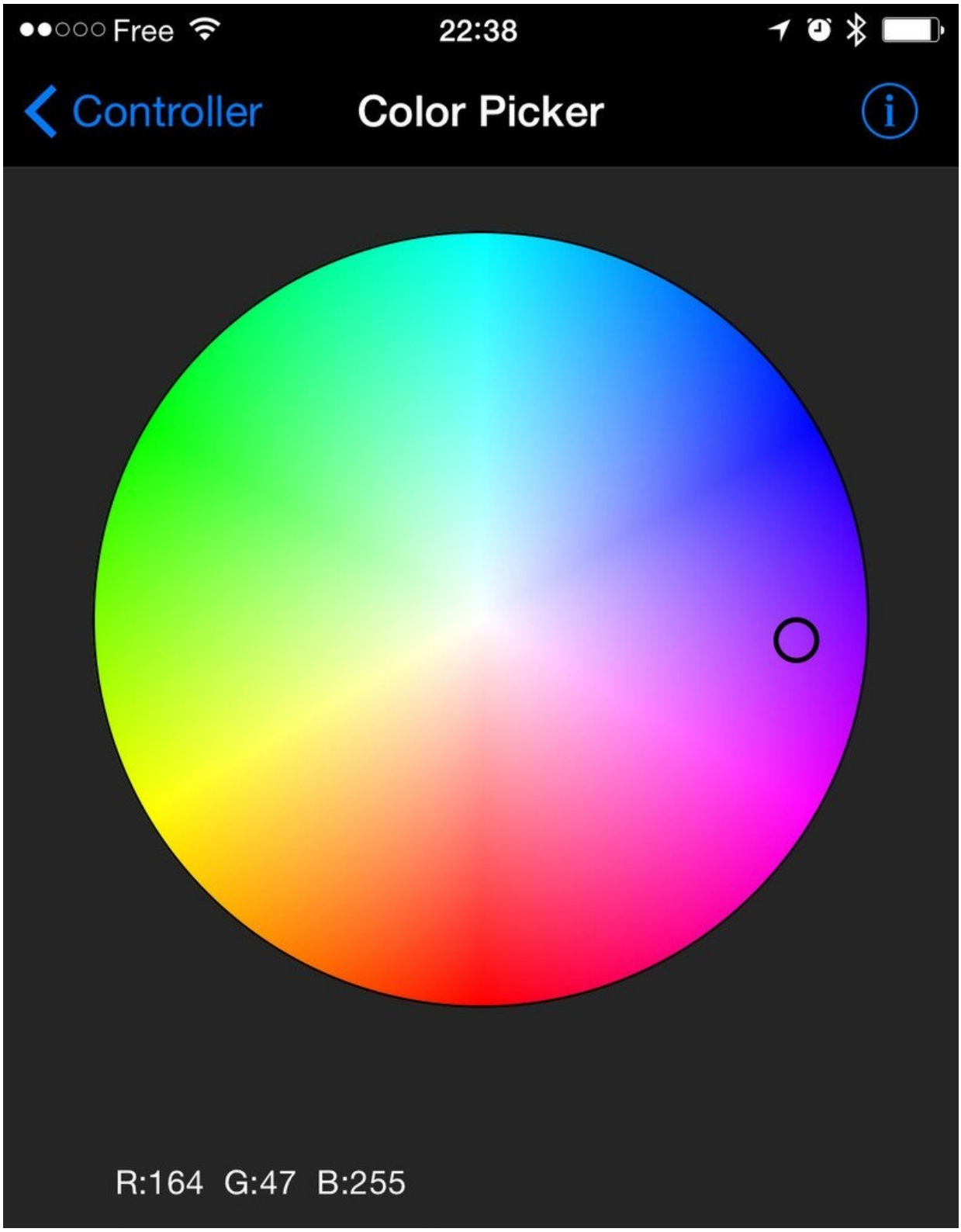
```
Button 8 pressed
Button 8 released
Button 3 pressed
Button 3 released
```

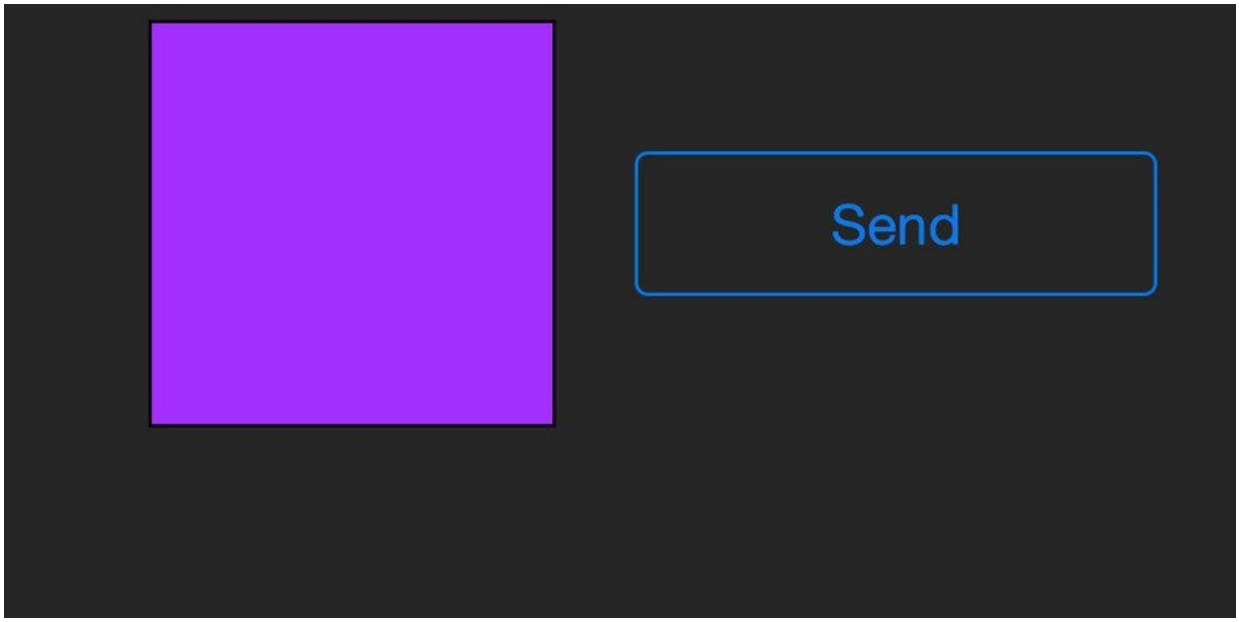
- **Pressed** signifie pressé.
- **Released** signifie relâché.

Sélection de couleur

Vous pouvez également envoyer des données d'une couleur sélectionnée RGB (*Red*=rouge, *Green*=vert, *Blue*=bleu) par l'intermédiaire du **module Color Picker**.

Le **module Color Picker** affiche une boite de sélection de couleur:





Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Sélectionner une coleur retourne le code couleur sous sa codification Hexadécimale avec le format suivant:

```
RGB #A42FFF
```

Voyez cette vidéo d'exemple produit par Adafruit https://youtu.be/Kym6crZF1Pg (*Youtube, anglais*)

Vous pouvez combiner ensemble les croquis d'exemples du "Color Picker" et du "controller" pour réaliser une animation couleur configurable déclenchée par les boutons de l'application mobile-- très pratique pour les projets wearables/fringuables! Vous pouvez télécharger cet exemple combiné (configuré pour Feather mais facile à adapter pour FLORA, BLE Micro, etc.) sur le lien suivant:

feather_bluefruit_neopixe
animation_controller.zip
https://learn.adafruit.com/system/assets/assets/000/029/260/original/feather_bluefruit_neopixel_animation_controller.zip?1450791688

Surveillance cardiaque

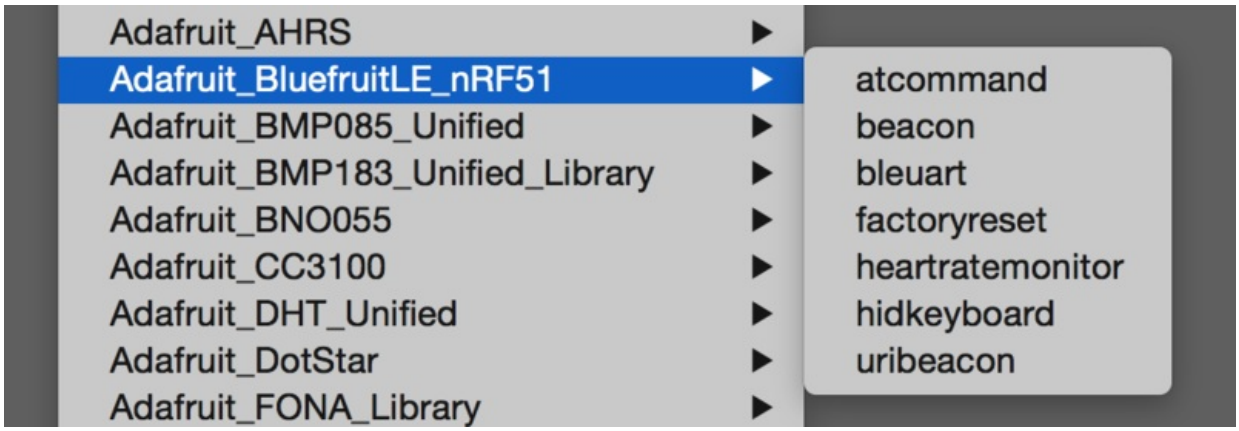
Sommaire

Le croquis HeartRateMonitor

L'exemple **HeartRateMonitor** vous permet de définir un nouveau service GATT, les caractéristiques GATT associées, de faire la mise-à-jour des valeurs de ces caractéristiques en utilisant des commandes AT.

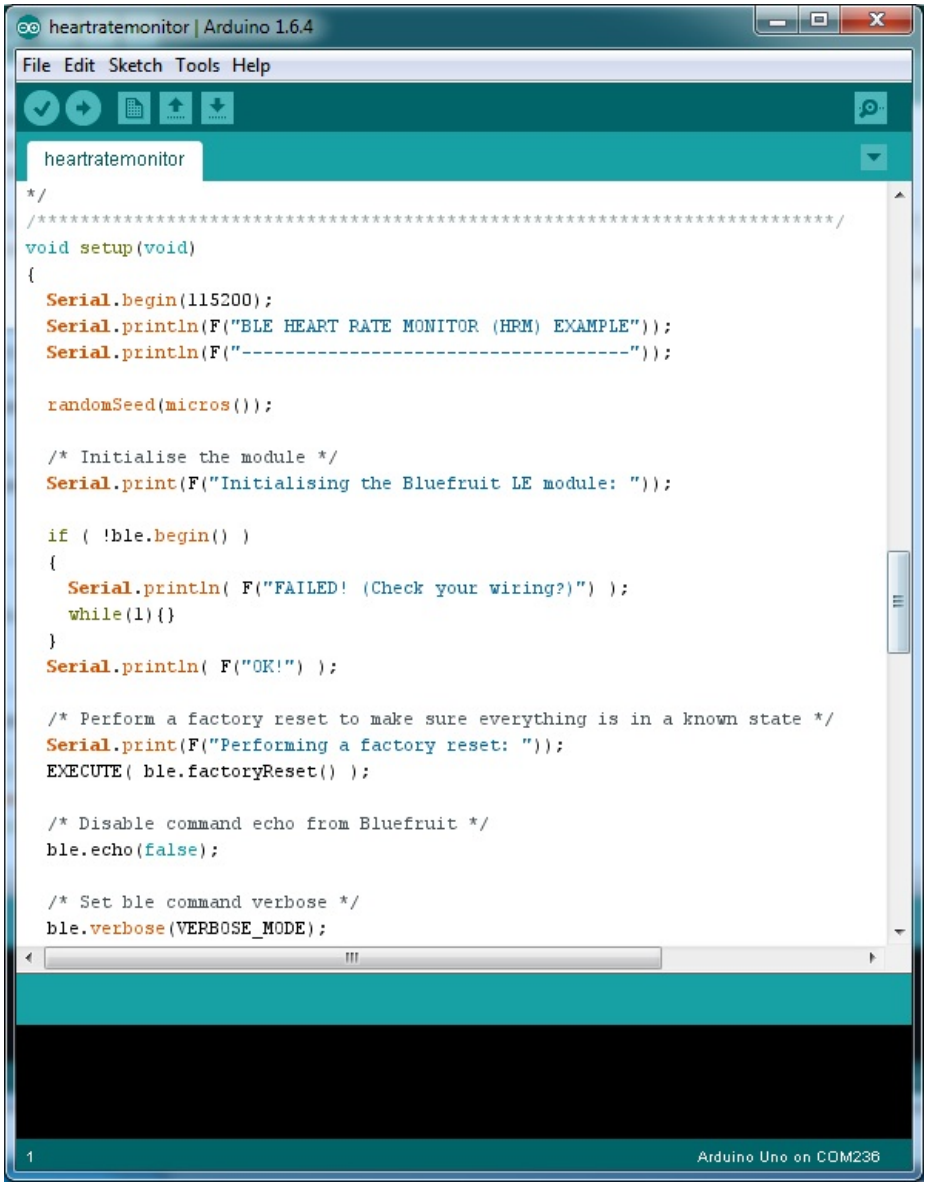
Ouvrir le Croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **heartratemonitor**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Configuration

Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel.

Si vous utilisez l'UART Hardware or Software UART

Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.

Cette démonstration nécessite de longs transfert de données pour envoyer des chaînes de caractères (string). Par conséquent, Adafruit recommande de définir et connecter les broches CTS et RTS, même si vous utilisez un UART matériel.

Si vous utilisez un Flora (ou que vous ne désirez pas brancher CTS ou RTS), fixer la valeur des broches à -1 dans les instructions #define to -1. **N'oubliez pas de connecter la broche CTS sur la masse de votre Bluefruit!** (PS: le Flora à déja cette broche branchée à la masse)

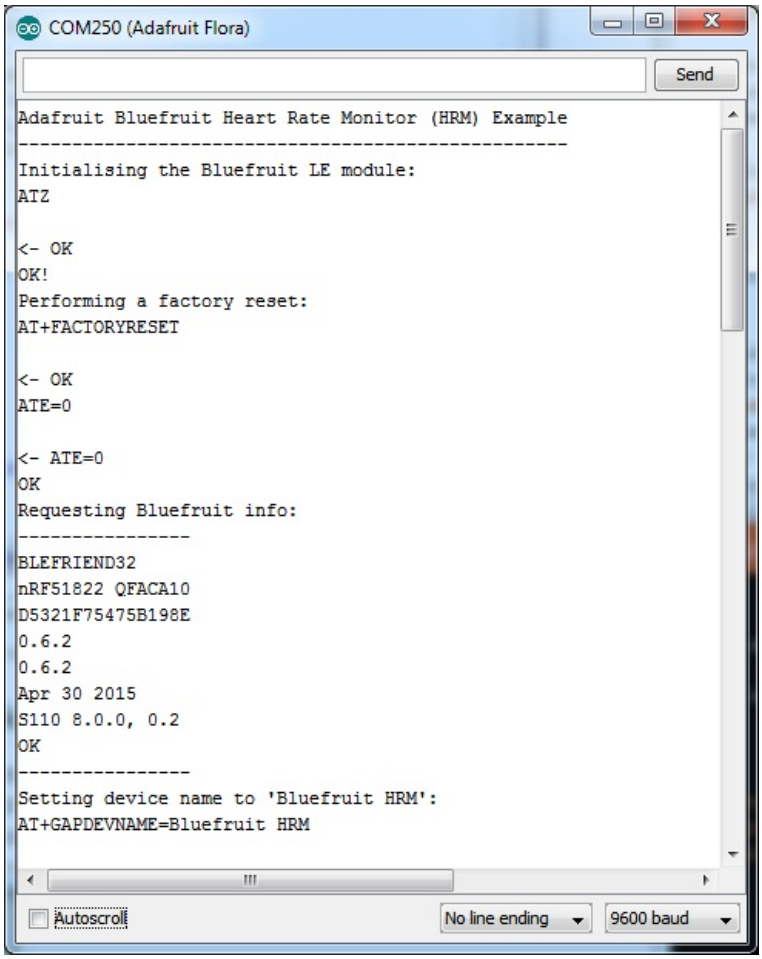
Si vous utilisez RTS et CTS alors vous pouvez retirer la ligne ci-dessous, elle est utilisée pou ralentir le transfert d'information.

```
// Traduction MC Hobby: Cette ligne est nécessaire pour l'implémentation
// Flora, mais c'est aussi une bonne idée si vous envoyez des supers longues
// ligne.
//
// This line is particularly required for Flora, but is a good idea
// anyways for the super long lines ahead!
ble.setInterCharWriteDelay(5); // 5 ms
```

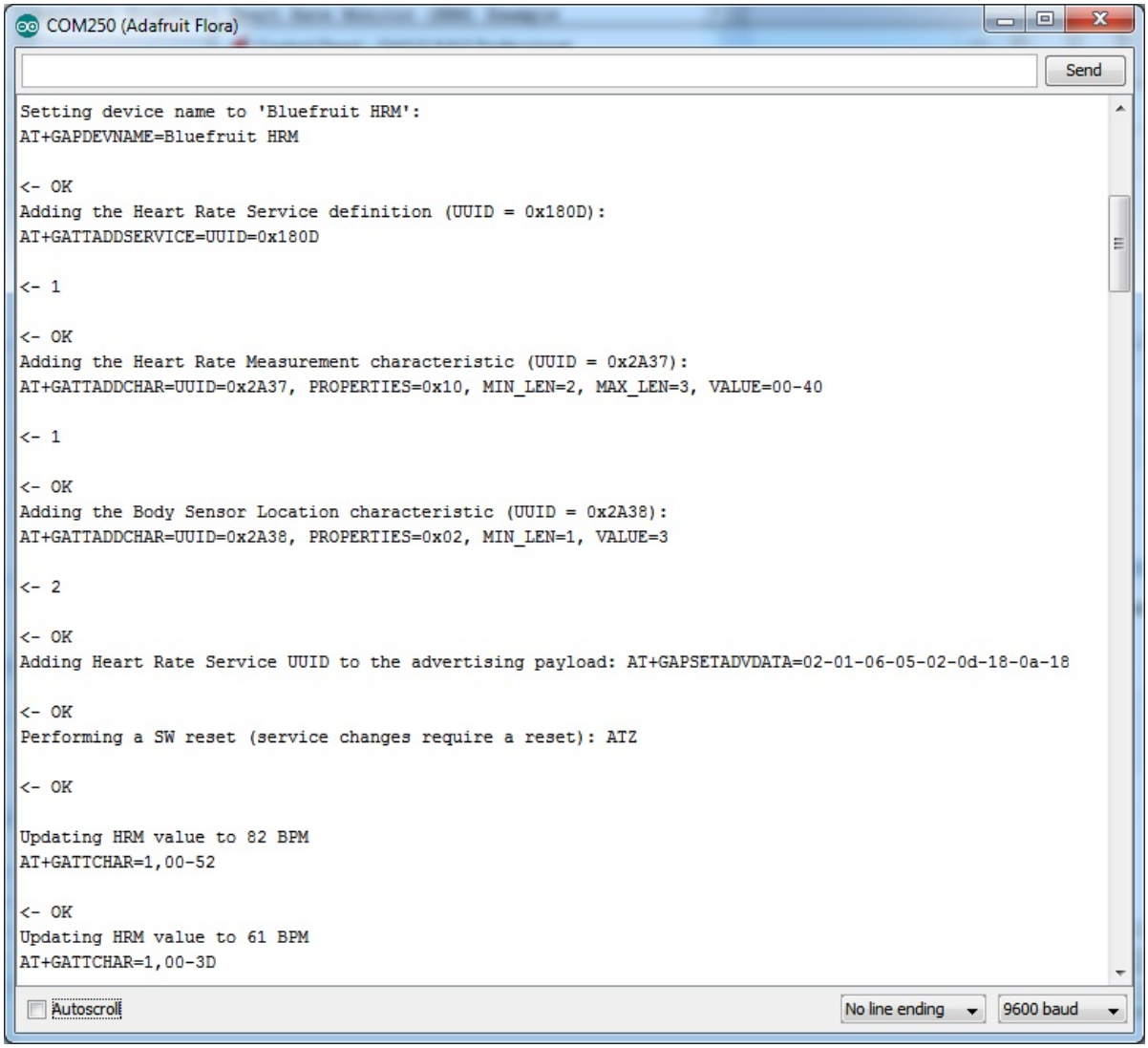
Exécuter le croquis

Une fois le croquis téléversé sur votre carte Arduino, vous pouvez ouvrir le moniteur série via le menu **Outils > Moniteur série**. Assurez-vous que le débit (Baud rate) soit configuré sur **115200** bauds (en bas à droite).

Une fois l'application démarrée, vous pouvez voir les différentes commandes de configurations envoyées vers le module BlueFruit LE pour ajouter le service Heart Beat Monitor.



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

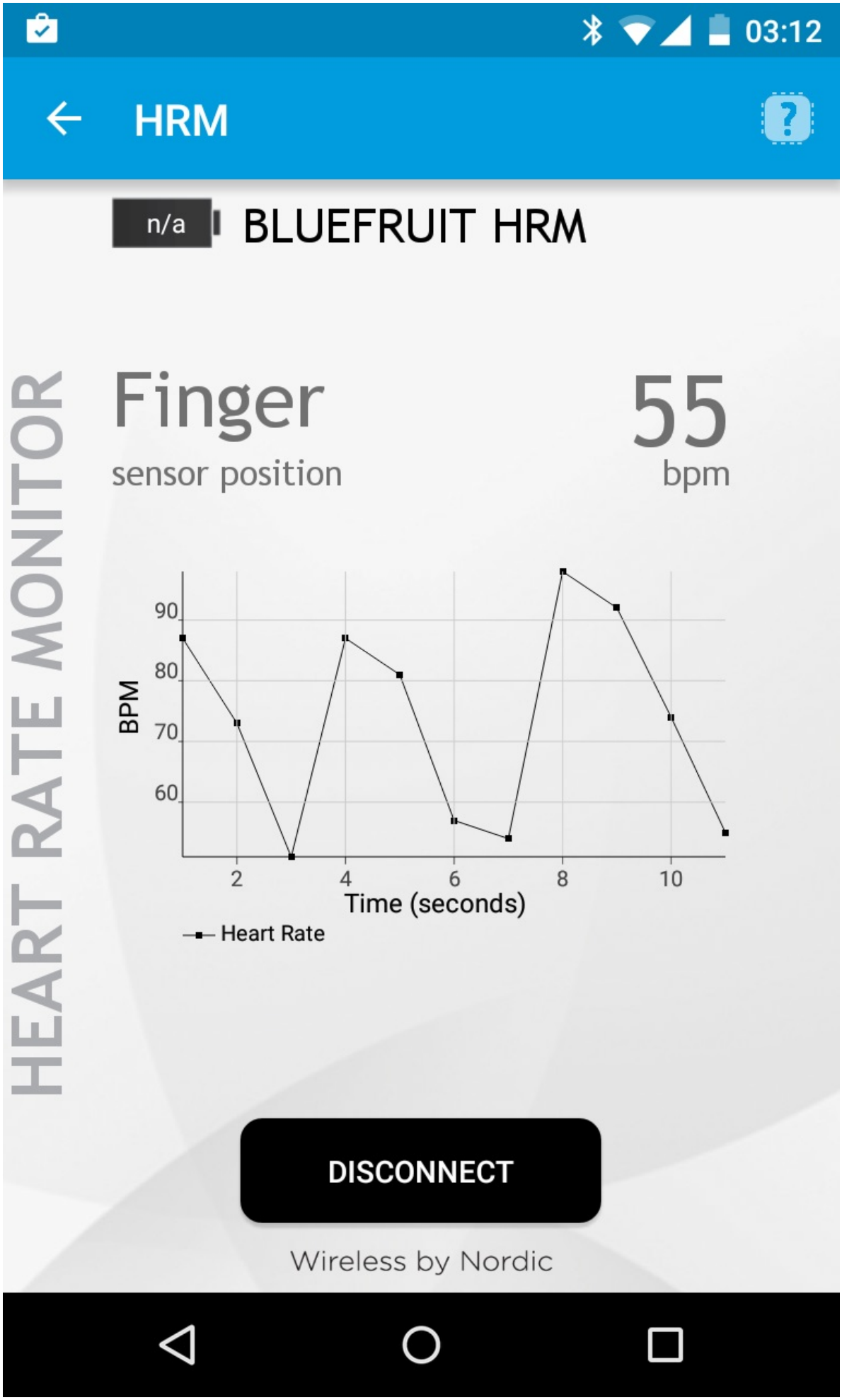


Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Exemple HRM de la boite à outil nRF

Si vous ouvrez une application (sur votre mobile ou Laptop) qui supporte le standard Heart Rate Monitor Service https://developer.bluetooth.org/TechnologyOverview/Pages/HRS.aspx , vous serez alors capable de voir les mises-à-jour de la vitesse de battement du coeur (en synchronisation avec les informations affichées dans le moniteur série):

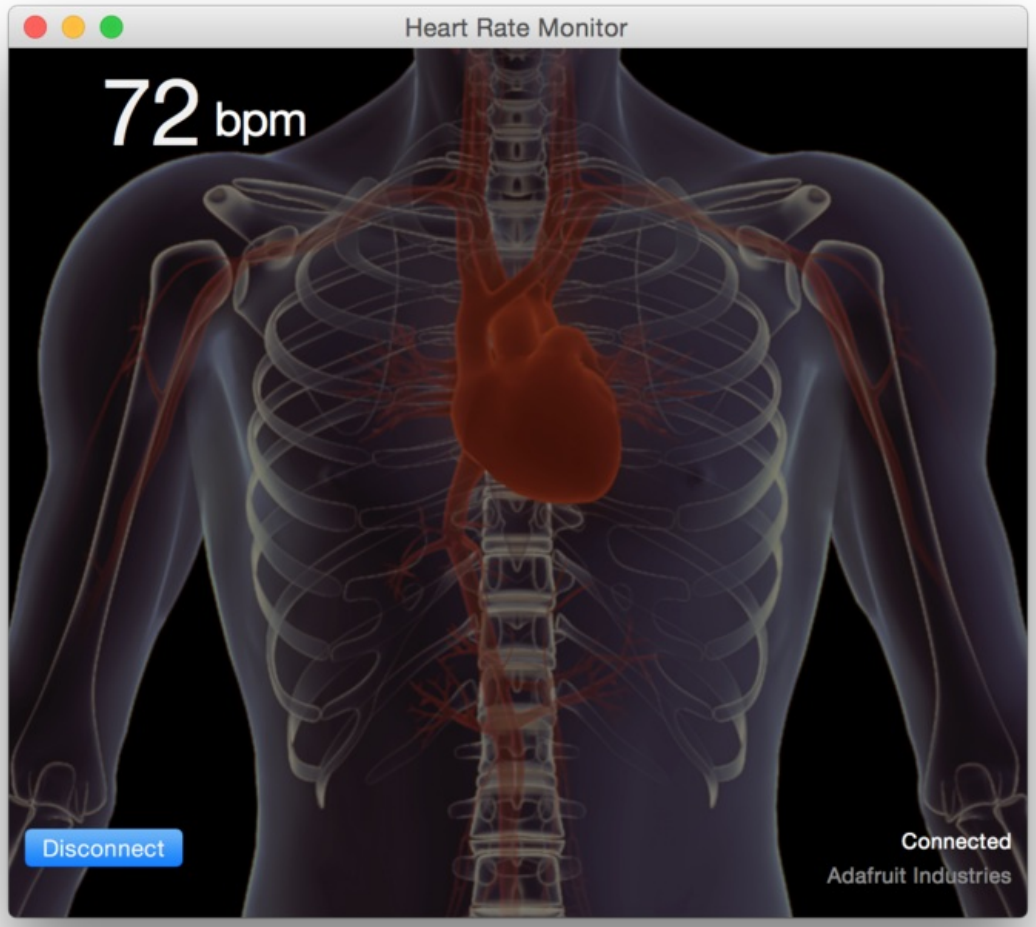
L'image ci-dessous présente une capture d'écran provenant de l'application "nRF Toolbox" de Nordic (sous Android, également disponible pour iOS). La capture indique les données arrivant sur le "Heart Rate Monitor":



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Exemple HRM CoreBluetooth

L'image ci-dessous présente une capture de l'application CoreBluetooth sample https://developer.apple.com/library/mac/samplecode/HeartRateMonitor/Introduction/Intro.html disponible gratuitement pour Apple montrant comment fonctionne les services et caractéristiques Bluetooth Low Energy:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

URI Beacon

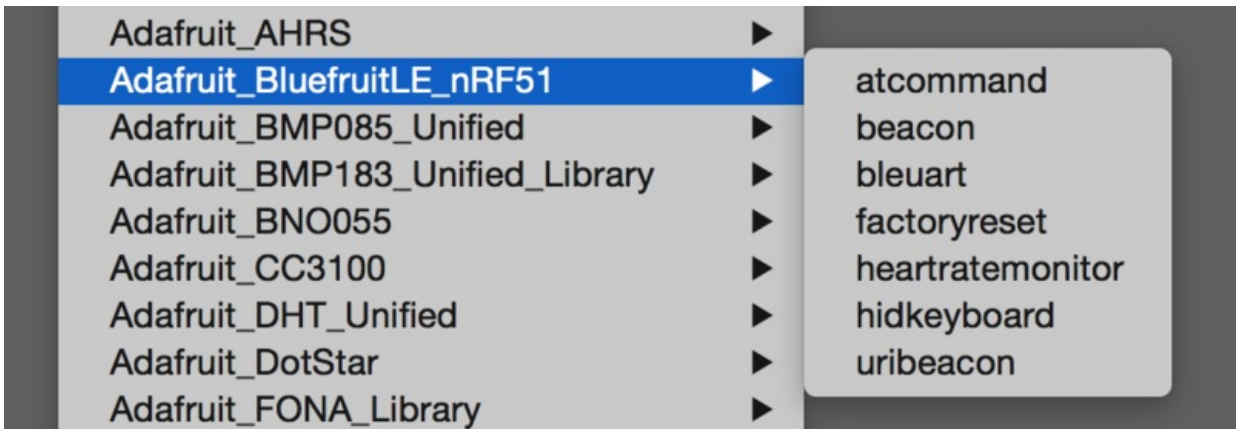
Sommaire

Le croquis UriBeacon

L'exemple **UriBeacon** montre comment utiliser les AT UriBeacon du module pour configurer le module Bluefruit LE comme annonceur UriBeacon en suivant les spécifications UriBeacon https://github.com/google/uribeacon de Google.

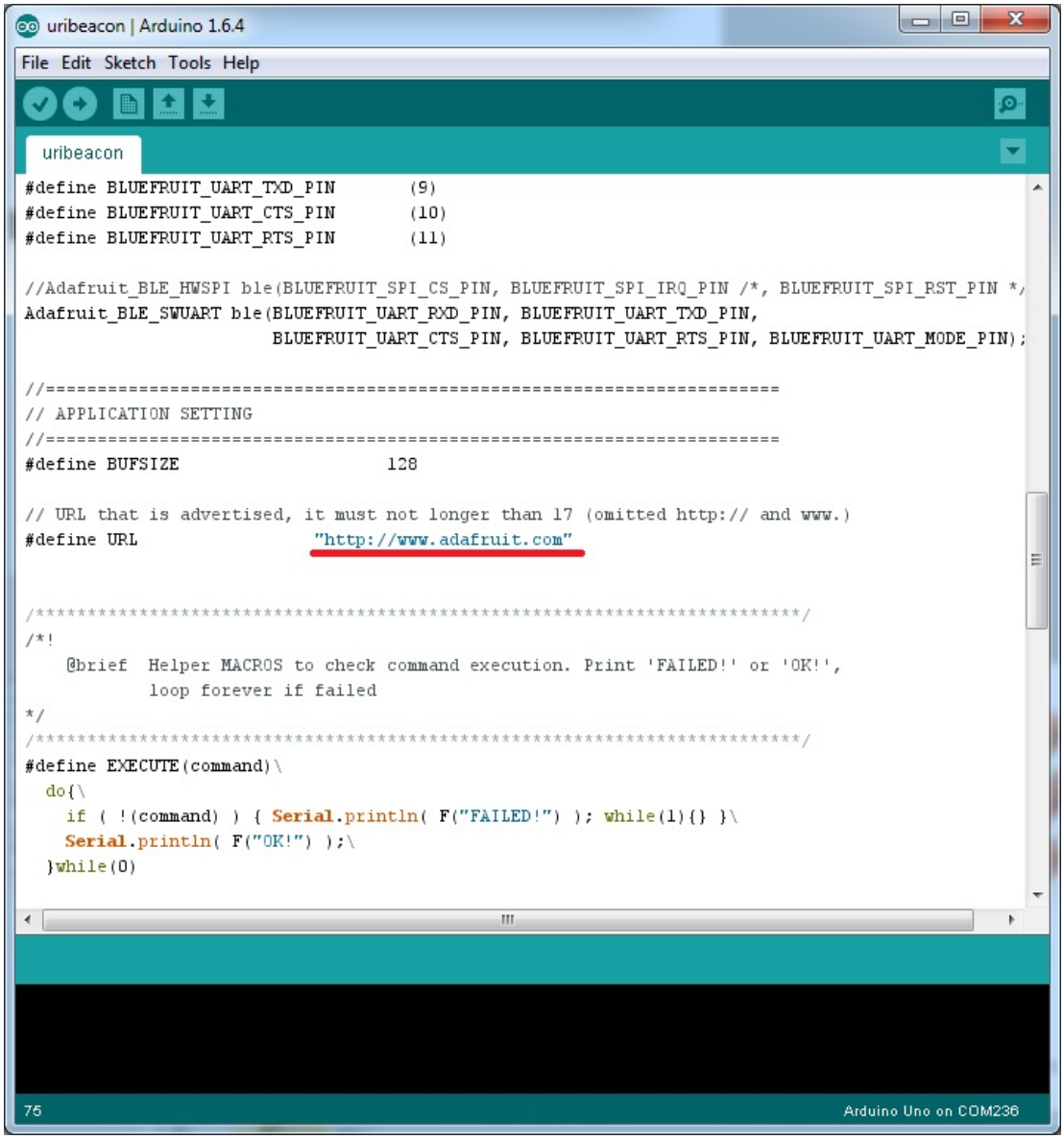
Ouvrir le croquis

Pour ouvrir le croquis ATCommand, cliquez sur le menu **Fichiers > Exemples > Adafruit_BluefruitLE_nRF51** dans Arduino IDE puis sélectionnez **uribeacon**:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Cela ouvrira l'exemple dans l'environnement de développement, comme visible ci-dessous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Vous pouvez éditer le contenu du beacon pour qu'il pointe l'URL vers **http://www.adafruit.com** ou juste téléverser le programme tel quel pour le tester.

Configuration

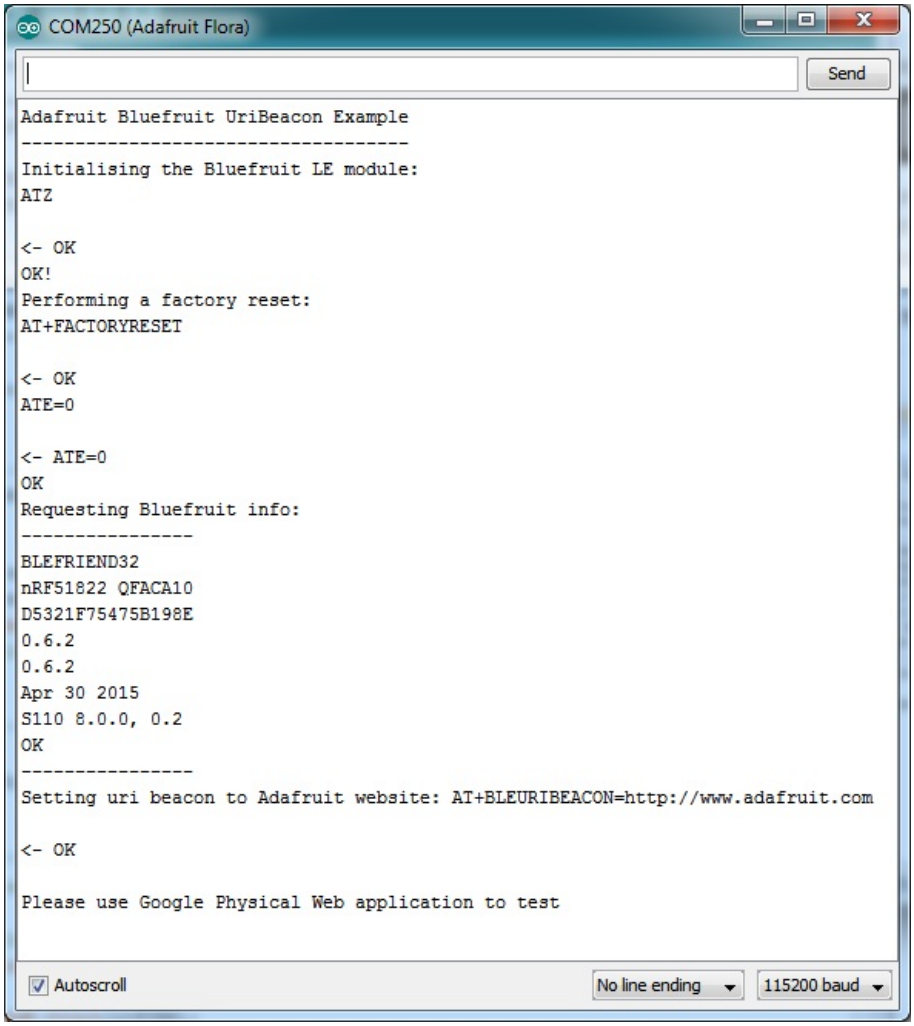
Vérifiez la page **Configuration** présenter plus tôt dans le tutoriel! Il est important de configurer le croquis pour utiliser soit l'UART Logiciel/Matériel, soit le bus SPI Logiciel/Matériel correspondant à votre plateforme. Par défaut, le croquis utilise le bus SPI matériel

Si vous utilisez le port série/UART logiciel ou matériel:

- Ce tutoriel ne requière pas l'utilisation de la broche MODE, **assurez-vous d'avoir l'interrupteur en position CMD** si vous ne configurer pas (et ne connectez pas) la broche MODE.
- N'oubliez pas de **connecter la broche CTS sur la masse/GND du Bluefruit si vous n'utilisez pas le signal CTS!** (Le Flora l'a déjà branché à la masse)

Exécuter le croquis

Une fois le croquis téléversé sur votre carte Arduino, vous pouvez ouvrir le moniteur série via le menu **Outils > Moniteur série**. Assurez-vous que le débit (Baud rate) soit configuré sur **115200** bauds (en bas à droite):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

C'est le moment d'ouvrir un l'application "Physical Web" pour Android https://play.google.com/store/apps/details?id=physical_web.org.physicalweb ou pour iOS https://itunes.apple.com/us/app/physical-web/id927653608?mt=8 et de voir le lien envoyé pour le liste web d'Adafruit:

Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

'''Dépannage'''

Les exemples ne démarrent pas!?!?

Q: Les exemples ne fonctionne pas lorsque lorsque j'utilise Bluefruit Micro ou un Bluefruit LE avec Flora/Due/Leonardo/Micro! Que se passe t'il?

Adafruit à ajouté une ligne spéciale dans la fonction **setup()** de sorte que votre Arduino s'arrête jusqu'à ce qu'il détecte la console Série. C'est très pratique pour le débogage mais empêche également le programme de fonctionner s'il n'est pas connecté sur un ordinateur.

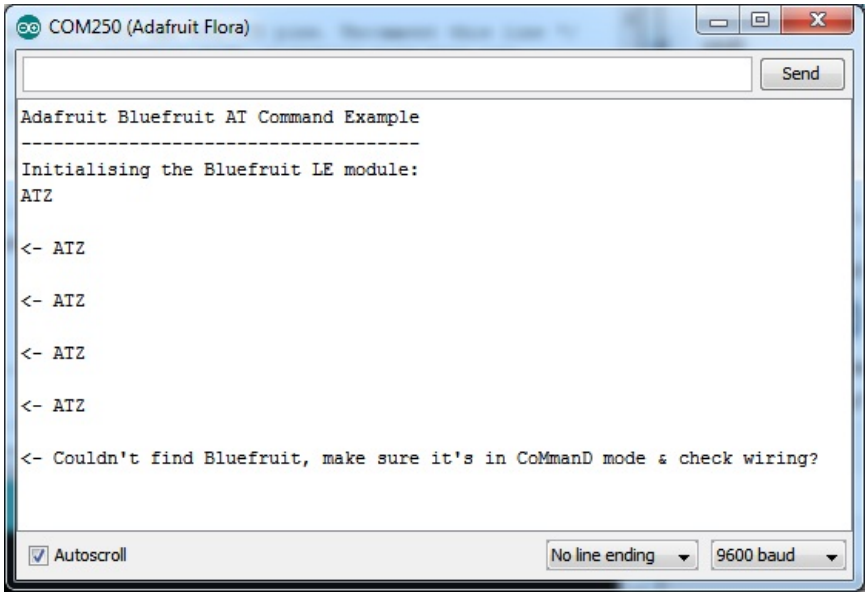
Solution: retirer les lignes suivante de la fonction setup() une fois votre programme débogué.

```
while (!Serial);
delay(500);
```

Ne trouve pas le module Bluefruit LE

Q: Je n'arrive pas à "trouver" le Bluefruit LE! Que faire?

Obtenez vous quelque-chose comme ceci?



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Pour les Bluefruits UART/Serie:

- Vérifiez que le l'interrupteur **MODE** est bien placé sur CMD et que la broche MODE n'est pas branchée sur quoi que ce soit (si elle n'est pas utilisée)!
- Si vous essayez de contrôler la broche **MODE** depuis votre microcontrôleur alors assurez vous d'avoir configuré la correctement la broche de votre microcontrôleur dans le croquis.
- Assurez vous que les broches **RXI** (entrée) et **TXO** (sortie) soient correctement raccordées! Il arrive qu'elles soient mal branchées sur le microcontroleur.
- Assurez-vous d'avoir raccordé la broche **CTS** sur la masse si vous utilisez un port série matériel sans signal CTS
- Contrôler la LED rouge du MODE. Clignote t'elle? Si elle clignote continuellement alors le module est en mode DFU (Device Firmware Update = mise-à-jour du Firmware). Coupez et rebranchez l'alimentation!
- Si vous utilisez un port série matériel ou logiciel: assurez vous d'identifier la type de port (Logiciel ou Materiel), d'avoir correctement identifié les broches et d'utiliser la classe d'objet adéquat.

Pour les Bluefruits sur bus SPI:

- Assurez vous que les branchements des 5 (ou 6) fils soient effectués correctement.
- Assurez vous d'être connecté sur le port SPI matériel si vous utilisez l'interface SPI matérielle. La position des broches du bus SPI matériel diffère d'une plateforme à l'autre.

Pour les Bluefruit Micro:

- Assurez vous d'avoir changé la broche **RESET** sur #4 dans le fichier de configuration. Assurez-vous d'utiliser les bonnes broches correspondant au bus SPI matériel!

Les commandes AT

Sommaire

A propos des commandes AT

Le module Bluefruit LE utilise un ensemble de commandes type "AT" (Hayes) http://en.wikipedia.org/wiki/Hayes_command_set pour configurer le périphérique.

L'avantage des commandes de type "AT" c'est qu'elles sont faciles à utiliser dans une communication machine-à-machine, tout en restant assez facile à lire/interpréter pour les humains.

Commande de test '=?'

Le mode de 'Test' est utiliser pour vérifier si une commande donnée existe (ou pas) dans le système.

Certaines versions de firmware ou certaines configurations pourraient (ou ne pourraient pas) inclure une commande spécifique. Vous pouvez tester la présence de la commande en saisissant le nom de la commande et en y ajoutant "**=?**" (comme dans l'exemple suivant:

```
AT+BLESTARTADV=?
```

Le périphérique réponde "**OK**" si la commande est présente. Si la commande n'existe pas alors le périphérique répondra "**ERROR**".

```
AT+BLESTARTADV=?
OK\r\n
AT+MISSINGCMD=?
ERROR\r\n
```

Les mentions de \r et \n représentent respectivement le retour à la ligne et passage à la ligne suivante. Ces caractère sont rarement affichés tels quels... mais il est important de savoir qu'il sont présents si vous faite dialoguer des machines ensembles.

Commandes en écriture '=xxx'

Le mode en écriture (*Write* en anglais) est utilisé pour pour assigner des valeurs spécifiques à la commande. Par exemple: changer le niveau de puissance du transmetteur radio. Cela permet de réduire la puissance pour économiser facilement de l'énergie. Voyez l'exemple ci-dessous:

Pour écrire une valeur pour la commande, ajoutez simplement un signe '**=**' à la commande suivit du paramètre/valeur que vous voulez fixer/écrire. La valeur doit être différente de '**?**' sinon la commande sera interprétée comme un mode de test:

```
AT+BLEPOWERLEVEL=-8
```

Si l'opération d'écriture est exécuté avec succès alors vous obtiendrez (en général) une réponse "**OK**" sur une nouvelle ligne. Voyez l'exemple ci-dessous:

```
AT+BLEPOWERLEVEL=-8
OK\r\n
```

S'il y a un problème avec la commande (comme un paramètre/valeur invalid), le module renverra la réponse 'ERROR' sur une nouvelle ligne. Voyez l'exemple ci-dessous:

```
AT+BLEPOWERLEVEL=3
ERROR\r\n
```

Note: Une erreur est générée dans ce cas particulier car la valeur '3' n'est pas valide pour la commande AT+BLEPOWERLEVEL . Saisir les valeurs '-4', '0' ou '4' seront acceptées car elles font parties des valeurs acceptées pas cette commande.

Commandes en mode exécution

Le mode d'exécution provoque l'exécution d'une commande spécifique, si cela est possible, et sera utilisé lorsque le nom de la commande est saisit dans paramètres additionnel.

```
AT+FACTORYRESET
```

Vous devriez utiliser le mode d'exécution pour effectuer une réinitialisation d'usine (*factory reset*) en exécutant la commande AT+FACTORYRESET comme ci-dessous:

```
AT+FACTORYRESET
OK\r\n
```

NOTE: De nombreuses commandes destinées à être utilisées en mode lecture produirons un résultat identique (donc une 'lecture') lorsqu'elles sont utilisées en mode exécution.

Par exemple, les deux commandes suivantes produisent le même résultat (mode exécution d'abord, mode lecture ensuite):

```
AT+BLEGETPOWERLEVEL
-4\r\n
OK\r\n
AT+BLEGETPOWERLEVEL?
-4\r\n
OK\r\n
```

Si le commande ne supporte pas le mode d'exécution, la réponse renvoyée est habituellement "**ERROR**" sur une ligne seule.

Mode de lecture '?'

Le mode lecture (**Read** en anglais) est utilisé pour lire la valeur actuelle d'une commande.

Le mode lecture n'est pas supporté par toutes les commandes. Vous pouvez cependant les utiliser pour retrouver des informations tel que le niveau de puissance de transmission actuellement en cours d'utilisation (tranmis power level for the radio). Il suffit d'ajouter un '?' à la commande, comme ci-dessous:

```
AT+BLEPOWERLEVEL?
```

Si la commande ne supporte pas le mode de lecture (ou s'il y a un problème avec la requêtre) alors le module retourne la réponse "**ERROR**".

Si la commande est lue avec succès, vous recevrez un "**OK**" sur une nouvelle ligne (et la valeur souhaitée). Voir l'exemple ci-dessous:

```
AT+BLEPOWERLEVEL?
-4\r\n
OK\r\n
```

Note: pour les commandes simples, le mode 'lecture' et mode d'exécution agissent de façon identique.

AT Standard

Sommaire

Commandes standards

Les commandes Hayes/AT standard suivantes sont disponible sur les modules Bluefruit LE:

AT

La commande AT agit comme un "ping". Elle est habituellement utilisé pour vérifier si le module est en mode "commande".

Si le module est en mode commande, vous devriez recevoir une réponse 'OK'.

- Codebase Revision: 0.3.0
- Paramètres: None
- Sortie: None

```
AT
OK
```

ATI

Affcihe les informations de base a propos du module BlueFruit.

- Codebase Revision: 0.3.0
- Paramètres: aucun
- Sortie: Affiche les valeurs suivantes:
  - Nom de la carte
  - Microcontrolleur/nom du SoC Radio
  - Numéro de série unique
  - Codebase Revision fonctionnant sur le coeur du Bluefruit
  - Firmware Revision du projet
  - Date de la compilation du Firmware
  - Softdevice, Softdevice Version, Bootloader Version (0.5.0+)

```
ATI
BLEFRIEND
nRF51822 QFAAG00
FB462DF92A2C8656
0.5.0
0.5.0
Feb 24 2015
S110 7.1.0, 0.0
OK
```

Updates:

- Version **0.4.7+** du firmware ajoute la révision de la puce après le nom de la puce (si cela peut être détecté, par exemple 'nRF51822 QFAAG00').
- Version **0.5.0+** du firmware ajoute un 7ieme enregistrement contenant le softdevice, version du softdevice et version du bootloader (ex. 'S110 7.1.0, 0.0').

ATZ

Effectue une réinitialisation système 'Reset'.

- Codebase Revision: 0.3.0
- Paramètres: None
- Sortie: None

```
ATZ
OK
```

ATE

Active ou désactive l'écho des caractères entrée avec le parseur de commande AT.

- Codebase Revision: 0.3.0
- Paramètres: '1' = activer l'écho, '0' = désactiver l'écho.
- Sortie: Aucune

```
# Désactive l'écho renvoyé par le module
ATE=0
OK
# Active l'écho
ATE=1
OK
```

+++

Permet de passer dynamiquement entre le mode DATA et COMMAND sans devoir changer l'interrupteur physique CMD/UART.

Lorsque vous êtes en mode COMMAND, saisir un '+++\n' ou '+++\r\n' fera passer le module en mode DATA. En mode DATA, tout ce qui est tapé sur la console est directement envoyé au service BLUE UART.

Pour revenir au mode COMMANDE depuis le mode DATA, saisissez simplement une nouvelle fois '+++\n' ou '+++\r\n' (Assurez vous d'avoir inclus le caractère de retour à la ligne!). Une nouvelle réponse 'OK' sera affiché pour vous informer du retour en mode COMMAND (voyez les deux entrées 'OK' dans le code ci-dessous, le 2ième est celui du retour en mode commande).

- Codebase Revision: 0.4.7
- Paramètres: None
- Sortie: None

Note que +++ peut également être utilisé sur les téléphone mobiles pour envoyer et recevoir des commandes AT sur iOS ou Android. Cette option doit toujours être utilisée avec précaution.

Voyez la commande AT+MODESWITCHEN qui contrôle la disponibilité de la commande +++

```
ATI
BLEFRIEND
nRF51822 QFAAG00
B122AAC33F3D2296
0.4.6
0.4.6
Dec 22 2014
OK
+++
OK
OK
```

Commandes générales

Sommaire

Commandes générales

Cette section du document décrit les commandes d'usage "généralisé" disponible sur tous les modules Bluefruit LE:

AT+FACTORYRESET

Efface les données de configuration de la mémoire non-volatile et effectue une réinitialisation d'usine avant de réinitialiser le module Bluefruit.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Aucun

```
AT+FACTORYRESET
OK
```

Depuis la version 0.5.0+ du firmware, vous pouvez effectuer une réinitialisation d'usine en maintenant le bouton DFU enfoncé pendant 10s jusqu'à ce que la LED bleue (CONNECTED) s'allume, vous pouvez ensuite relâcher le bouton.

AT+DFU

Force le module à passer en mode DFU (Device Firmware Update), permettant de faire une mise-à-jour du firmware via les airs (connexion sans fil) en utilisant l'application mise-à-jour de firmware dédicacée disponible pour iOS et pour Android.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Aucun

Le parser de commande AT ne répondra plus après la saisie d'une commande AT+DFU, puisque le flux d'exécution normal du module est arrêter et et le système interne redémarré pour activer le bootloader du module.

```
AT+DFU
OK
```

AT+HELP

Affiche une liste (séparée par des virgules) des différentes commandes disponibles sur le parser AT.

- Codebase Version: 0.3.0
- Paramètres: Aucun
- Sortie: Une liste des commandes AT disponibles sur le module. A comma-separated list of all AT parser commands available on the system.

L'exemple ci-dessous pourrait ne pas correspondre aux futures versions du Firmware

(ce n'est qu'une illustration)

```
AT+HELP
AT+FACTORYRESET,AT+DFU,ATZ,ATI,ATE,AT+DBGMEMRD,AT+DBGNVMRD,AT+HWLEDPOLARITY,AT+HWLED,AT+HWGETDIETEMP,AT+HWMODEPINPOLARITY,AT+HWMODEPIN,
OK
```

AT+NVMWRITE

Ecrit des données dans la mémoire utilisation non volatile de 256 octets (*bytes*). Aussi dite région NVM pour **N**on-**Volatile m**emory.

- Codebase Version: 0.7.0
- Paramètres:
  - offset: L'offset numérique de la position de départ dans la NVM où sera écrit le premier octet/byte.
  - datatype: Type de donnée, qui peut être une STRING (1), BYTEARRAY (2) ou INTEGER (3). Respectivement *Chaîne de caractère*, *Tableau d'octets*, *Entier*.
  - data: Les données à écrire dans la mémoire NVM (le format exacte du *payload* varie en fonction du type de donnée spécifiée).
- Sortie: Aucune

```
# Ecrit l'entier 32768 à partir de la position 16 de la mémoire NVM
AT+NVMWRITE=16,INTEGER,32768
OK
```

AT+NVMREAD

Lit des données depuis la mémoire NVM de 256 octets/bytes (mémoire utilisateur non volatile).

- Codebase Version: 0.7.0
- Paramètres:
  - offset: l'offset (valeur numérique) du premier octet à lire dans la mémoire NVM.
  - size: Le nombre d'octets/bytes à lire.
  - datatype: Le type de donnée qui sera lue dans la mémoire NVM. Cette information est requise pour pouvoir extraire correctement le type de donnée à extraire et convertir dans la réponse. La valeur doit être une des options suivantes STRING (1), BYTEARRAY (2) or INTEGER (3)
- Sortie: Les données extraite de la NVM et formatée sur base du type de donnée attendu (voir argument *datatype*).

```
# Lire un entier stocké à l'offset 16 de la mémoire NVM
# Un entier fait 4 octets de long.
AT+NVMREAD=16, 4, INTEGER
32768
OK
```

AT+MODESWITCHEN

Active ou désactive la possibilité de passer du mode COMMAND au mode DATA à l'aide de la commande '+++' (sur un périphérique BLE, du côté de la connexion UART du BLE).

- Codebase Version: 0.7.1
- Paramètres:
  - location: doit être une chaîne de caractère, avec soit 'local' ou 'ble' indiquant quel côté de la communication peut recevoir la commande '+++'. 'local' concerne le périphérique Bluefruit et 'ble' concerne le smartphone ou tablette.
  - state: '0' pour désactiver la commande '+++', '1' pour l'activer.
  - Sortie: Aucun

Par défaut, '+++' est activé localement (périphérique BlueFruit) et désactivé en BLE (côté smartphone)

```
# Désactivé la commande '+++' pour le périphérique distant (le smartphone BLE)
AT+MODESWITCHEN=ble,0
OK
```

Matériel

Sommaire

Commandes matériel

Les commandes suivantes vous permettent d'interagir avec la bas niveau matériel du module Bluefruit LE. Vous pourrez lire et modifier l'état des GPIO, effectuer une conversion Analogique-vers-digital (ADC), etc.

AT+BAUDRATE

Change le débit (*baud rate* en anglais) utilisé par l'UART matériel du nRF51822.

- Codebase Revision: 0.7.0
- Paramètres: débit en bauds. Doit être une des valeurs suivantes:
  - 1200
  - 2400
  - 4800
  - 9600
  - 14400
  - 19200
  - 28800
  - 38400
  - 57600
  - 76800
  - 115200
  - 230400
  - 250000
  - 460800
  - 921600
  - 1000000
- Sortie: Le configuration courante du débit

```
# Fixe le débit à 115200 bauds
AT+BAUDRATE=115200
OK

# Vérifier la configuration du débit
AT+BAUDRATE
115200
OK
```

AT+HWADC

Effectue une conversion ADC (analogique vers digital) pour une broche ADC spécifique

- Codebase Revision: 0.3.0
- Paramètres: Le canal ADC (0..7)
- Sortie: Le résultat de la conversion ADC

```
AT+HWADC=0
178
OK
```

AT+HWGETDIETEMP

Obtenir la température (en degrés Celcius) du composant BLE. Peut être utilisé pour faire du débogage (un courant plus élevé signifie généralement une consommation de courant plus important). Cette information ne correspond pas à la température ambiante et ne peut donc pas être utilisé pour remplacer un senseur de température normal.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Le température du composant BLE (en degré Celcuis)

```
AT+HWGETDIETEMP
32.25
OK
```

AT+HWGPIOMODE

Cette commande permet de configurer le mode d'une broche GPIO (en entrée/sortie, etc).

- Codebase Revision: 0.3.0
- Paramètres: cette commande accepte une ou deux valeur (séparés par une virgule lorsqu'il y a deux paramètres):
  - Le numéro de broche/GPIO
  - Le nouveau mode de la broche/GPIO, où:
    - 0 = Entrée/Input
    - 1 = Sortie/Output
    - 2 = Entrée avec activation PullUp (résistance interne qui ramène le potentiel par défaut de la broche à +VCC)
    - 3 = Entrée avec activation PullDown (résistance interne qui ramène le potentiel par défaut de la broche à GND)
- Sortie: Si un seul paramètre est fourni à la commande (le numéro de broche GPIO) alors la commande retourne le mode actuellement configuré pour la broche.

Certaines broches sont réservées pour des fonctions spécifiques du module Bluefruit et ne peuvent pas être utilisées comme GPIO. La commande retournera 'ERROR' si vous essayez d'utiliser ces numéros réservés.

```
# Configurer la broche 14 en sortie/output
AT+HWGPIOMODE=14,0
OK

# Obtenir le mode en cours d'utilisation pour la broche 14
AT+HWPGIOMODE=14
0
OK
```

AT+HWGPIO

Lire ou fixer la valeur de la broche GPIO spécifiée (en fonction du mode de la broche).

- Codebase Revision: 0.3.0
- Paramètres: les paramètres de cette commande changent en fonction du mode de la broche.

GPIO en sortie

MODE DE SORTIE/OUTPUT MODE: Les paramètres suivants (séparés par des virgules) peuvent être utilisés pour modifier l'état d'une broche configurée en sortie:

- Numéro de broche
- Etat de la broche, où:
  - 0 = désactive la broche (Niveau logique bas/LOW/GND)

- 1 = active la broche (Niveau logique haut/HIGH/VCC)

GPIO en entrée

MODE ENTREE/INPUT MODE: Pour lire létat actuel d'une broche en entrée (ou lire l'état d'une broche en sortie). Utiliser le numéro de broche comme le seul paramètre de la commande.

- Sortie: L'état de la broche si vous lisez une entrée (ou vérifiez l'état de sortie d'une broche). La valeur retournée est:
  - 0 signifiant que la broche est au niveau logique bas/LOW/GND
  - 1 signifiant que la broche est au niveai logique haut/HIGH/VCC

Recommendation

	***Essayer de fixer l'état d'une broche qui n'a pas encore été configurée produira la réponse 'ERROR'.***
	***Certaines broches du BlueFruit LE sont réservée pour des fonctions spécifiques. Ces broches ne peuvent pas être utilisées comme GPIO. La commande génèrera une erreur si vous essayez d'utiliser ces numéros réservés.***

Exemples

```
# Place la broche 14 au niveau haut/HIGH
AT+HWGPIO=14,1
OK

# Place la broche 14 au niveau bas/LOW
AT+HWGPIO=14,0
OK

# Lire l'état de la broche 14 14
AT+HWGPIO=14
0
OK

# Essayer de modifier une broche qui n'est pas
# configuré en sortie/output
AT+HWGPIOMODE=14,0
OK
AT+HWGPIO=14,1
ERROR
```

AT+HWI2CSCAN

Fait un scan du bus I2C et essaye de détecter les périphériques I2C qui y sont connectés. Retourne une lise des adresses détectés par le processus de scan.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Une liste des adresse I2C occupées (séparées par des virgules) durant le processus de scan, ou rien si aucun périphérique n'est détecté.

```
# Scan I2C avec deux périphériques détecté
AT+HWI2CSCAN
0x23,0x35
OK

# Scan I2C sans périphériques détectés
AT+HWI2CSCAN
OK
```

AT+HWVBAT

Retourne la mesure de la tension d'alimentation en milli-volts

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Le niveau de VBAT en milli-volts

```
AT+HWVBAT
3248
```

```
OK
```

AT+HWRANDOM

Génère un nombre aléatoire (32-bits) en utilisant le générateur de nombre aléatoire matériel du nRF51822 (basé sur le bruit blanc).

- Codebase Revision: 0.4.7
- Paramètres: Aucun
- Sortie: un nombre aléatoire 32-bits encodé en hexadécimal (ex. '0x12345678')

```
AT+HWRANDOM
0x769ED823
OK
```

AT+HWMODELED

Permet d'écraser la fonctionnalité par défaut de la LED "mode" (qui indique, par défaut, indique le mode de fonctionnement).

- Codebase Revision: 0.6.6
- Paramètres: mode de fonctionnement de la LED, qui peut être une des valeurs suivantes:
  - **disable** ou **DISABLE** ou **0** - Désactive complètement la Led MODE pour économiser de l'énergie.
  - **mode** ou **MODE** ou **1** - Fonctionnement par défaut, indique le MODE actuel.
  - **hwuart** ou **HWUART** ou **2** - reconfigure l'activité de la LED sur l'activité de l'UART matériel (bus série matériel, TX ou RX)
  - **bleuart** ou **BLEUART** ou **3** - reconfigure l'activité de la LED sur l'activité du service BLE UART (caractéristique TX ou RX)
  - **spi** ou **SPI** ou **4** - reconfigure l'activité de la LED sur l'activité SPI
  - **manual** ou **MANUAL** ou **5** - permet de configurer l'état de la Led MODE à l'aide d'un second paramètre (séparé par une virgule) qui peut être "on", "off" ou "toggle" (pour inverser l'état).
- Sortie: Si la commande est exécutée sans paramètre, la commande retourne une chaîne de caractère en majuscule représentant le mode de fonctionnement actuel de la LED (tel que présenté ci-dessus).

```
# Obtenir la configuration de la Led
AT+HWMODELED
MODE
OK

# Changer le fonctionnement de la Led vers BLEUART
AT+HWMODELED=BLEUART
OK

# Inverser manuellement l'état de la Led MODE
AT+HWMODELED=MANUAL,TOGGLE
OK
```

AT+UARTFLOW

Active ou désactive le contrôle de flux matériel (CTS + RTS) pour le port série/UART sur le nRF51822.

- Codebase Revision: 0.7.0
- Paramètres: Etat du contrôle de flux matériel. Peut prendre l'une des valeurs suivantes:
  - **on** ou **1** - activer
  - **off** ou **0** - désactiver
- Sortie: Si la commande est exécutée sans paramètre, elle retour un nombre 0 ou 1 indiquant si le contrôle de flux est activé (1) ou désactivé (0).

```
# Vérifie l'état du contrôle de flux
AT+UARTFLOW
1
OK

# Désactive le contrôle de flux matériel
AT+UARTFLOW=off
OK
```

Technologie Beacon

Sommaire

Beacon

Le module Bluefruit LE d'Adafruit supporte les technologies 'Beacon' suivantes:

- Beacon (Apple) via AT+BLEBEACON
- UriBeacon (Google) via AT+BLEURIBEACON (obsoltète)
- Eddystone (Google) via AT+EDDYSTONE*

Les modules peuvent être configurés pour agir comme 'Beacons' en utilisant les commandes décrites dans cette partie du document.

AT+BLEBEACON

- Codebase Revision: 0.3.0
- Paramètres: les paramètres suivants (séparés par des virgules) sont nécessaires pour activer le mode Beacon:
  - Bluetooth Manufacturer ID (uint16_t)
  - UUID sur 128-bits
  - Valeur majeure (uint16_t)
  - Valeur mineure(uint16_t)
  - RSSI à 1m (int8_t)
    RSSI représente la mesure de la qualité du signal
- Sortie: None

```
# Activer l'émulation Apple iBeacon
# Manufacturer ID = 0x004C
AT+BLEBEACON=0x004C,01-12-23-34-45-56-67-78-89-9A-AB-BC-CD-DE-EF-F0,0x0000,0x0000,-59
OK
# Réinitialiser pour changer les données de publication
ATZ
OK

# Activer l'émulation Beacon de Nordic
# Manufacturer ID = 0x0059
AT+BLEBEACON=0x0059,01-12-23-34-45-56-67-78-89-9A-AB-BC-CD-DE-EF-F0,0x0000,0x0000,-59
OK
# Réinitialiser pour changer les données de publication
ATZ
OK
```

AT+BLEBEACON stocke les données beacon dans ma mémoire de configuration du module Bluefruit LE (non volatile). Ces données sont persistance et donc disponible après une réinitialisation système ou cycle d'alimentation. Pour effacer ou enlever ces données beacon il sera nécessaire d'utiliser la commande de réinitialisation à la configuration d'usine 'AT+FACTORYRESET'.

En utilisant l'émulation Beacon de Nordic avec l'exemple ci-dessus, vous pourrez voir le résultat de la simulation beacon dans l'outil 'Beacon Config' de Nordic (comme ci-dessous):

Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

AT+BLEURIBEACON

Converti l'URI spécifée en paquet d'annonce UriBeacon (*advertising packet*) et configure le module pour s'annoncer comme un UriBeacon (partie du projet "Physical Web" de Google).

Vous pouvez utiliser l'une des aapplication suivantes pour voir l'URI annoncé dans par l'UriBeacon:

- Android 4.3+: "Physical Web" sur Google Play Store
- iOS: "Physical Web" sur l'App Store d'Apple

- Codebase Revision: 0.4.7
- Paramètres: l'URI à annoncer (ex. http://www.adafruit.com/blog)
- Sortie: Aucune si une URI valide à été saisi (longueur acceptable, etc.).

```
AT+BLEURIBEACON=http://www.adafruit.com/blog
OK

# Reinitialise le module pour utiliser les données d'annonce
ATZ
OK
```

Si l'URI fournie est trop longue alors vous obtiendrez le message suivant en sortie:

```
AT+BLEURIBEACON=http://www.adafruit.com/this/uri/is/too/long
URL is too long
ERROR
```

Si vous essayez d'encoder une URI trop longue alors essayez d'utiliser un service en ligne permettant de créer des liens court (comme bit.ly) et encodez ce lien court dans l'UriBeacon.

UriBeacon devrait être considéré comme un standard obsolète. EddyStone devrait être utilisé pour chaque nouveau développement. Adafruit ne fera pas de développement complémentaire pour la technologie UriBeacon.

AT+EDDYSTONEENABLE (commande dépréciée)

Cette commande activera le support Eddystone sur le module Bluefruit LE. Le support Eddystone doit être activé avant d'utiliser les autres commandes.

- Codebase Revision: 0.6.6
- Paramètres: 1 ou 0 (1 = activé, 0 = désactiver)
- Sortie: l'état actuel du support Eddystone si aucun paramètre est fournit à la commande. 1 = activé, 0 = désactivé

Cette commande est remplacée par AT+EDDYSTONESERVICEEN depuis le firmware 0.7.0 . La commande AT+EDDYSTONEENABLE n'est plus disponible depuis la version 0.7.0 afin d'éviter les confusions.

```
# Activer le support Eddystone
AT+EDDYSTONEENABLE=1
OK

# Vérifier le statut du support Eddystone sur le module
AT+EDDYSTONEENABLE
1
OK
```

AT+EDDYSTONEURL

Cette commande fixe l'URL pour le le protocol Eddystone.

- Codebase Revision: 0.6.6
- Paramètres:
  - L'URL a encoder (obligatoire)
  - Le second paramètre (optionel) indique s'il faut continuer à envoyer l'URL Eddystone même si le périphérique est connecté sur un périphérique central
  - Firmware **0.6.7** ajout un 3ieme paramètre optionnel pour la valeur RSSI à 0 mètre. Devrait être mesurer par l'utilisateur final en vérifiant la valeur RSSI sur le périphérique de réception à 1 mètre de distance puis en ajoutant 41 à cette valeur (pour compenser la perte de signal sur un mètre). Un RSSI de -62 à 1m signifie qu'il faudrait encoder la valeur -21 comme valeur RSSI à 0m. La valeur par défaut est -18dBm.
- Sortie:
  - Firmware <= 0.6.6: rien.
  - Firmware >= **0.6.7** exécuter cette commande sans paramètre retournera l'URL actuellement configurée.

```
# Fixe l'URL Eddystone sur Adafruit
AT+EDDYSTONEURL=http://www.adafruit.com
OK

# Fixe l'URL Eddystone sur Adafruit et effectue la publication même lorsqu'il y a une connexion
AT+EDDYSTONEURL=http://www.adafruit.com,1
OK
```

AT+EDDYSTONECONFIGEN

Cette commande active le service de configuration de l'URL Eddystone sur le module Bluefruit LE pendant le nombre de secondes spécifiées.

Cette commande devrait être utilisée en combinaison avec l'application "Physical Web" de Google (disponible sus Android et iOS).

Utilisez cette commande pour sélectionner l'option 'Edit URL' (édition URL) depuis l'Application. Cela permet de changer l'URL de destination via les airs.

- Codebase Revision: 0.6.6
- Paramètres: le nombre de secondes pour l'annonce du service de configuration UUID
- Sortie: Aucun

```
# Active l'annonce du service de configuration Eddystone pendant 5 minutes (300s)
AT+EDDYSTONECONFIGEN=300
OK
```

AT+EDDYSTONESERVICEEN

Ajoute ou retire un service Eddystone dans la table GATT (gouverne l'organisation et l'échange de données entre les périphériques). Nécessite une réinitialisation pour être activé.

- Codebase Revision: 0.7.0
- Paramètres: Utiliser les valeurs suivantes pour activer ou désactiver le service Eddystone:
  - on , 1 - activer le service
  - off , 0 - désactiver le service
- Sortie: Si la commande est utilisée sans paramètre, le commande retournera une valeur indiquant si le service est actif ou non (1 = activé, 0 = désactivé).

Il est nécessaire de réinitialiser le système pour que la commande soit prise en compte.

```
# Active le service Eddystone
AT+EddyStonServiceEn=on
OK

AT+EddyStonServiceEn=1
OK

# Désactive le service Eddystone
AT+EddyStonServiceEn=off
OK

AT+EddyStonServiceEn=0
OK
```

AT+EDDYSTONEBROADCAST

Cette commande peut être utilisé pour démarrer / arrêter l'annonce (le *broadcast*) du *payload* Eddystone. Eddystone utilise l'URL stockée dans la mémoire non-volatile du module.

- Codebase Revision: 0.7.0
- Paramètres: Indique s'il faut (ou non) faire l'annonce / broadcast du *payload* contenant l'URL.
  - on , 1 : Activer la broadcast
  - off , 0 : Désactiver le broadcast
- Sortie: If executed with no Paramètres, the current broadcast state will be displayed as a numeric value.

```
# Active le broadcast de la configuration EddyStone (l'URL, telle que stocké dans la mémoire non volatile)
AT+EddyStoneBroadcast=on
OK

AT+EddyStoneBroadcast=1
OK

# Désactive le broadcast de la configuration EddyStone
AT+EddyStoneBroadcast=off
OK

AT+EddyStoneBroadcast=0
OK
```

Cmd. BLE génériques

Sommaire

Commandes génériques BLE

Les commandes générales suivantes sont disponibles dans tous les modules Bluefruit LE.

AT+BLEPOWERLEVEL

Obtenir ou fixer le niveau de puissance de transmission du module radio (réduire la puissance permet d'augmenter la durée de vie de votre accu/pile car le module consomme moins, cela diminue également la portée du module).

- Codebase Revision: 0.3.0
- Paramètres: Le niveau de puissance en émission (en dBm) qui doit correspondre à l'une des valeurs suivantes:
  - -40 : la puissance de transmission la plus basse.
  - -20
  - -16
  - -12
  - -8
  - -4
  - 0
  - 4 : la puissance de transmission la plus élevée
- Sortie: La valeur actuelle de la puissance de transmission (en dBm)

La niveau de puissance prend effet dès que la commande est exécutée. Si le périphérique n'est pas connecté sur un autre périphérique, le processus d'annonce sera arrêté et redémarrera une fois la puissance de transmission changée.

```
# Obtenir le niveau de puissance en émission (valeur actuelle, en dBm)
AT+BLEPOWERLEVEL
0
OK

# Fixe le niveau de puissance à 4dBm (la valeur max)
AT+BLEPOWERLEVEL=4
OK

# Fixe le niveau de puissance à -12dBm (meilleure durée de vie de l'accu)
AT+BLEPOWERLEVEL=-12
OK

# Fixe le niveau de puissance à une valeur invalid
AT+BLEPOWERLEVEL=-3
ERROR
```

AT+BLEGETADDRTYPE

Obtient le type d'adresse utilisé par le module (pour les périphérique BLE disposant d'une adresse périphérique sur 48-bits).

Normalement, la valeur sera '1' (*aléatoire*), ce qui signifie que le module utiliser une adresse 48-bits qui a été générée aléatoirement durant le processus de fabrication. Cette adresse est écrite dans le module par le fabriquant.

Aléatoire ne signifie pas qu'une adresse aléatoire est générée à chaque fois mais qu'un nombre (généré aléatoirement, généré une seule fois) sera utilisé à chaque fois.

- Codebase Revision: 0.3.0

- Paramètres: Aucun
- Sortie: Le type d'adresse qui peut être l'une des valeurs suivantes:
  - 0 = publique
  - 1 = aléatoire

```
AT+BLEGETADDRTYPE
1
OK
```

AT+BLEGETADDR

Retourne l'adresse 48-bits du périphérique BLE.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: une adresse 48-bits au format suivant: 'AA:BB:CC:DD:EE:FF'

```
AT+BLEGETADDR
E4:C6:C7:31:95:11
OK
```

AT+BLEGETPEERADDR

Retounr l'adresse 48-bits du périphérique appairé (*peer*, *centrale*) sur lequel nous somme connectés.

- Codebase Revision: 0.6.5
- Paramètres: Aucun
- Sortie: L'adresse 48-bits du périphérique central connecté (au format hexadécimal). La commande retourne ERROR si nous ne sommes pas connectés sur un périphérique centrale.

Notez que l'adresse retournée par le périphérique centrale est toujours une valeur aléatoire qui peut changer dans le temps. Cette valeur ne peut pas un élément de sécurisation. Cette commande est fournir pour certains cas d'utilisation particulier, elle n'est généralement pas utilise dans les scénarios couramment.

```
AT+BLEGETPEERADDR
48:B2:26:E6:C1:1D
OK
```

AT+BLEGETRSSI

Obtient la valeur RSSI (*Received Signal Strength Indicator*) qui peut être utilisé pour estimer la fiabilité de la transmission de données entre deux périphériques (Plus ce nombre est bas et meilleure est la fiabilité de transmission).

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Le niveau RSSI (en dBm) si nous sommes connecté sur un périphérique; sinon '0'

```
# Connecté sur un périphérique externe
AT+BLEGETRSSI
-46
OK

# Non connecté sur un périphérique externe
AT+BLEGETRSSI
0
OK
```

Services BLE

Sommaire

Services BLE

Les commandes suivantes vous permettent d'interagir avec les différents services GATT présent sur les modules Bluefruit LE lorsque celui-ci fonctionne en mode commande.

Le service GATT et ses caractéristiques https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-uart-friend/ble-gatt (*Adafruit, anglais*) gouvernent l'organisation et l'échange de données entre les périphériques.

AT+BLEUARTTX

Cette commande transmet le message texte spécifié en paramètre via le Service UART https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/uart-service pendant que le module fonctionne en mode commande.

- Codebase Revision: 0.3.0
- Paramètre: Le payload (données) à transmettre. Le payload peut faire jusqu'à 240 caractères (étant donné que les commandes AT sont limitées à un maximum de 256 octet).
- Sortie: Cette commande produit le message erreur **ERROR** s'il n'est pas possible de se connecter sur le périphérique central, ou si la mémoire tampon d'émission interne du BluFruit (internal TX FIFO) est remplie.

Sur un firmware **0.6.2** et supérieur, AT+BLEUARTTX peut accepter quelques séquences d'échappements:

- \r = retour clavier
- \n = Nouvelle ligne
- \t = Tabulation
- \b = retour en arrière (backspace)
- \\ = Anti-slash

Sur un firmware **0.6.7** et supérieur, AT+BLEUARTTX peut accepter la séquence d'échappement suivante (puisque AT+BLEUARTTX=? à une signification particulière pour le parseur AT):

- \? = transmet un simple point d'interrogation

Sur un firmware **0.7.6** et supérieur, AT+BLEUARTTX peut accepter les séquences d'échappement suivantes:

- \+ = Transmet un simple caractère '+' sans avoir à s'inquiéter de la combinaison de `+++` (changement de mode)

NOTE SUR LES SEQUENCES D'ECHAPPEMENT: Si vous essayez d'envoyer les séquences d'échappement dans le code par l'intermédiaire d'une instruction telle que 'ble.print("...");' prenez note qu'il vous vaudra utiliser des double-back-slash pour le code d'échappement d'une commande AT. Par exemple: ble.println("AT+BLEUARTTX=Some Test\\r\\n");

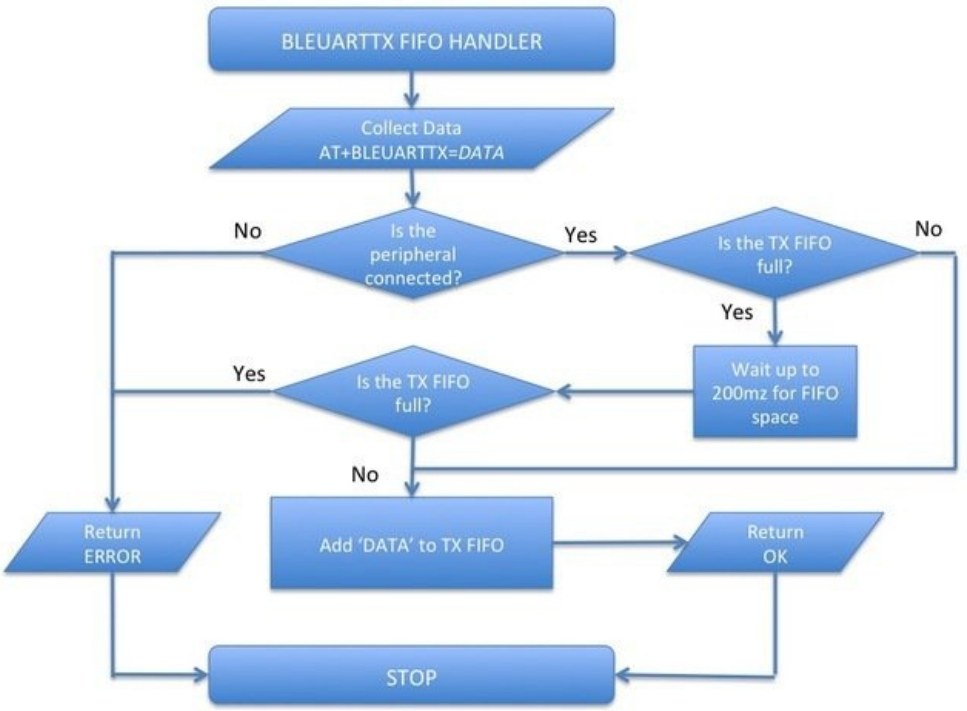
Vous devez être connecté sur un autre périphérique pour que cette commande s'éxécute.

```
# Envoi une chaîne de caractère lorsqu'il
# est connecté sur un autre périphérique
AT+BLEUARTTX=THIS IS A TEST
OK

# Réponse lorsque le Bluefruit n'est pas connecté
# sur un autre périphérique
AT+BLEUARTTX=THIS IS A TEST
ERROR
```

Gestion de la mémoire tampon de réception

Depuis le firmware **0.6.7**, lorsque la mémoire tampon FIFO d'émission est saturée (TX FIFO buffer), un délai bloquant de 200ms est utilisé pour voir si de l'espace se libère dans la pile FIFO avant de retourner une erreur (le message "ERROR"). Le processus exact est détaillé dans le graphique suivant:



Note: La vérification complète de la mémoire tampon FIFO d'émission sera réalisé pour chaque transaction GATT (jusqu'à 20 octets de donnée par transaction), par conséquent, les importants transferts de donnée peuvent rencontrer plusieurs fois des délais d'attente de 200ms.

Vous pouvez utiliser la commande **AT+BLEUARTFIFO=TX** pour vérifier la taille du buffer FIFO d'émission avant d'envoyer des donnée. Cela permet de s'assurer qu'il y a assez de place libre dans le buffer.

La mémoire tampon (buffer) d'émission à la taille suivante (la taille dépend de la version du Firmware utilisé):

- Firmware <=0.6.6: **160 caractères**
- Firmware >=0.6.7: **1024 caractères**

Lors de grand transfert de donnée, il est possible qu'une partie du payload soit transmit, et que la commande produise une erreur si la mémoire tampon FIFO d'émission ne se vide pas dans les temps au milieu du processus de transfert des données (les données sont transmissent par paquet de 20 octets).

Vous devriez toujours vérifier la taille de la mémoire tampon FIFO d'émission si vous désirez assurer un transfert fiable. La commande AT+BLEUARTFIFO permet de connaître cette taille. Si la taille n'est pas suffisante pour inclure votre payload alors introduisez un délai d'attente dans votre programme pour laisser l'opportunité à la mémoire tampon de se vider.

De simples commande AT+BLEUARTTX peuvent tenir dans le FIFO, mais de multiple instance de cette commande (avec donnée) peuvent remplir le FIFO en cours de transfert.

AT+BLEUARTTXF

C'est une fonction commode qui réalise la même fonction que AT+BLEUARTTX **mais** un envoi immédiat dans un simple packet BLE ('F' pour Forcer le paquet). Cette commande acceptera un maximum de 20 caractères, ce qui est la limite de ce qui peut être envoyé dans un seul paquet.

- Codebase Revision: 0.7.6
- Paramètres: voyez AT+BLEUARTTX
- Sortie: voyez AT+BLEUARTTX

AT+BLEUARTRX

Cette commande fait un dump de la mémoire tampon de réception du service UART https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/uart-service sur l'écran (si le service UART à reçu des données pendant l'exécution en mode de commande). Les données seront retirée de la mémoire tampon une fois affichées en utilisant cette commande.

Tout caractère laissé dans la mémoire tampon en repassant en mode Data provoquera l'affichage des caractères de la mémoire tampon dès la fin du changement de mode (dans les limites de l'espace disponible dans la mémoire tampon, qui est de 1024 octets --ou 160 octets pour pour la première génération des cartes BLEFriend).

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Le contenu de la mémoire tampon en réception (RX buffer) s'il y en a des données disponibles. Sinon rien.

Vous pouvez également utiliser la commande AT+BLEUARTFIFO=RX pour voir s'il n'y a des données disponibles.

```
# Résultat de la commande lorsqu'il y en a de disponible
AT+BLEUARTRX
Sent from Android
OK

# Résultat de la commande lorsqu'il n'y a pas de donnée disponible
AT+BLEUARTRX
OK
```

AT+BLEUARTFIFO

Cette commande retournera l'espace disponible dans les FIFOs de l'UART BLE (les mémoires tampons TX et RX). Si vous transmettez de grandes quantités de données, vous pourriez avoir besoin de vérifier si vous avez assez d'espace libre dans la mémoire tampon FIFO TX avant l'émission. Gardez à l'esprit que les paquet GATT individuels peuvent chacun contenir 20 octets (données utilisateur).

- Codebase Revision: 0.6.7
- Paramètre: exécuter cette commande sans paramètre retournera les valeurs (séparée par des virgules) indiquant l'espace disponible dans la mémoire tampon de d'émission (buffer TX), suivit de la mémoire tampon en réception (RX buffer). Pour faire une requête sur une mémoire tampon spécifique, vous pouvez exécuter la commande avec soit la valeur "TX" ou "RX" (Par exemple: "AT+BLEUARTFIFO=TX").
- Sortie: L'espace libre dans les mémoires tampons en émission et réception (TX and RX FIFO buffer). S'il y a un paramètre complémentaire, seul le buffer concerné retourna l'information souhaitée.

```
AT+BLEUARTFIFO
1024,1024
OK

AT+BLEUARTFIFO=TX
1024
OK

AT+BLEUARTFIFO=RX
1024
OK
```

AT+BLEKEYBOARDEN

Cette commande va activer le support "GATT over HID" (GoH) - le support clavier, ce qui vous permet d'émuler un clavier sur les périphériques iOS et Android.

Par défaut, ce support clavier HID est désactivé. Par conséquent fixez BLEKEYBOARDEN à 1 puis effectuez une réinitialisation système avant que le "clavier" soit énuméré et apparaisse dans les préférences Bluetooth de votre téléphone (où il pourra être lié comme clavier BLE).

- Codebase Revision: 0.5.0
- Paramètres: 1 ou 0 (1 = activer, 0 = désactiver)
- Sortie: Aucun

	Depuis le firmware 0.6.6 cette commande est un alias pour AT+BLEHIDEN
	***Vous devez effectuer une réinitialisation système (ATZ) avant que les changements soient effectifs!***
	***Avant de pouvoir utiliser votre clavier "HID over GATT", vous aurez besoin de lier le module Bluefruit LE avec votre mobile (dans le panneau des préférences Bluetooth).***

```
# Activer le support clavier BLE puis réinitialisation
AT+BLEKEYBOARDEN=1
OK
ATZ
OK

# Désactive le support clavier puis réinitialisation
AT+BLEKEYBOARDEN=0
OK
ATZ
OK
```

AT+BLEKEYBOARD

Envoi des données textes via l'interface clavier BLE (si elle a précédemment été activée avec AT+BLEKEYBOARDEN).

- Codebase Revision: 0.5.0
- Paramètre: la chaîne de caractère (incluant éventuellement des caractères d'échappement) à transmettre
- Sortie: Aucune

Il est possible d'envoyer tout caractère Alpha-Numérique valide. Les séquences d'échappement suivantes sont supportées:

- \r - Retour à la ligne (*Carriage Return*)
- \n - Passer à la ligne suivante (*Line Feed*)
- \b - Un caractère en arrière (*Backspace*)
- \t - Une tabulation (*Tab*)
- \\ - Un anti-slash (*Backslash*)

Depuis la version 0.6.7, vous pouvez également utiliser la séquence suivante pour envoyer un simple caractère ? seul). Car la commande 'AT+BLEKEYBOARD=?' à une autre interprétation pour l'interpréteur de commande:

- \? - point d'interrogation

```
# Envoi une URI avec retour à la ligne (pour être exécuté dans Chrome, etc).
AT+BLEKEYBOARD=http://www.adafruit.com\r\n
OK

# Envoi un simple point d'interrogation (cas particulier du firmware 0.6.7+)
AT+BLEKEYBOARD=\?
```

```
OK
```

AT+BLEKEYBOARDCODE

Envoi une séquence brute (en hexadécimale) de codes de touche HID USB (des *keycodes* en anglais) vers l'interface clavier BLE incluant les codes de modification (Shift, Alt, Control,...). Accepte jusqu'à 6 caractères alpha-numériques.

- Codebase Revision: 0.5.0
- Paramètre: un ensemble de valeurs hexadécimales séparés par un tiret ('-'). Notez qu'il s'agit des valeurs des scancode HID, non des valeurs standards ASCII!
- Sortie: Aucune

> Le valeur des codes touches HID (scancode HID) ne correspondent pas aux codes ASCII! Par exemple, 'a' dispose d'un code touche HID de '04' et il n'y a pas de code touche pour le 'A' puisqu'elle s'obtient avec le code de modification majuscule. Une recherche Google sur les termes 'usb hid keyboard scan codes' (et voyez l'exemple ci-dessous).

Cette commande accepte les valeurs ASCII encodées comme suit dans le *payload* hexadécimal, cela correspond à la façon dont "HID via GATT" envoi les données claviers:

- **octet 0**: Code de modification (appelé *modifier* en anglais).
- **octet 1**: Reservé (devrait toujours être 00)
- **octets 2..7**: les valeurs hexadécimales pour les caractères encodés en ASCII (Vous pouvez saisir '00' s'il n'y a pas de caractères utilisés ou laisser vide les caractères en fin de commande)

Après avoir envoyé une séquence de code touche avec la commande AT+BLEKEYBOARDCODE, vous devez envoyer une seconde commande avec AT+BLEKEYBOARDCODE avec au moins deux caractères 00 pour indiquer que les touches sont relâchées!

```
# Envoyer 'abc' avec la touche majuscule/shift enfoncée
# permet d'obtenir 'ABC'
AT+BLEKEYBOARDCODE=02-00-04-05-06-00-00
OK
# Indiquer que les touches sont relâchées (obligatoire!)
AT+BLEKEYBOARDCODE=00-00
OK
```

Une liste des code touche HID peut être trouvé ici http://www.freebsddiary.org/APC/usb_hid_usages.php (voir la section 7).

Valeurs de modification

Ces valeurs de modification (dite *modifier* en anglais) permettent d'indiquer si plusieurs autres touches particulières sont également enfoncées (Shift, Alt, Ctrl, ...).

L'octet de modification peut avoir un ou plusieurs des bits suivants activés:

- Bit 0 (0x01): Contrôle Gauche (*Left Control*)
- Bit 1 (0x02): Majuscule Gauche (*Left Shift*)
- Bit 2 (0x04): Alternatif Gauche (*Left Alt*)
- Bit 3 (0x08): Window Gauche (*Left Window*)
- Bit 4 (0x10): Contrôle Droit (*Right Control*)
- Bit 5 (0x20): Majuscule Droite (*Right Shift*)
- Bit 6 (0x40): Alternatif Droit (*Right Alt*)
- Bit 7 (0x80): Window Droit (*Right Window*)

AT+BLEHIDEN

Cette commande active le support GoH (*GATT over HID*) permettant d'emuler un clavier, souris ou contrôle de média tels qu'ils sont supportés sur les périphériques iOS, Android, OSX et Windows 10. Le support HID est désactivé par défaut. Il est donc nécessaire de l'activer en fixant la valeur à 1 dans BLEHIDEN puis de faire une réinitialisation système avant que le périphérique HID soit énuméré et apparaisse sur votre périphérique centrale (votre smartphone).

- Codebase Revision: 0.6.6
- Paramètres: 1 = activer ou 0 = désactivé
- Sortie: Aucune

> Il sera normalement nécessaire de *lier/appairer* le périphérique Bluefruit LE pour pouvoir

utiliser les commandes HID. Le processus d'appairage varie d'un système d'exploitation à l'autre.

Si vous avez déjà appairé le module avec un périhérique et que vous ayez besoin d'effacer cet appairage alors vous pouvez exécuter la commande AT+FACTORYRESET qui effacera toutes les données du module Bluefruit LE (y compris celle d'appairage).

```
# Activer le support "GATT over HID" du module Bluefruit LE
AT+BLEHIDEN=1
OK

# Réinitialiser le module pour activer la fonctionnalité
ATZ
OK
```

AT+BLEHIDMOUSEMOVE

Déplace la souris HID (ou fait un défilement si un nombre d'impulsion est précisé).

Toutes les valeurs sont des entiers 8 bits signées (-128 à +127). Les valeurs positives provoque un déplacement vers la droite / vers le bas. L'origine des coordonnées se trouvent dans le coin supérieur gauche.

- Codebase Revision: 0.6.6
- Paramètres: impulsions X (+/-), impulsions Y (+/-), rouge de défilement (+/-), défilement horizontal (+/-)
- Sortie: Aucun

```
# Déplace la souris de 100 impulsions à droite et 100 impulsions vers le bas
AT+BLEHIDMOUSEMOVE=100,100
OK

# Faire un défilement de 20 pixels ou lignes (en fonction du contexte) vers le bas
AT+BLEHIDMOUSEMOVE=,,20,
OK

# Défilement horizontal vers la droite (la fonctionnalité dépend de l'OS)
AT+BLEHIDMOUSEMOVE=0,0,0,100
```

AT+BLEHIDMOUSEBUTTON

Permet de manipuler les boutons de la souris HID à l'aide de textes spécifiques.

- Codebase Revision: 0.6.6
- Paramètres: **Bouton** à l'aide [L][R][M][B][F], **Action** à l'aide de [PRESS][CLICK][DOUBLECLICK][HOLD]
  - Bouton:
    - L = Bouton gauche, *Left Button*
    - R = Bouton droit, *Right Button*
    - M = Bouton du milieu, *Middle Button*
    - B = Bouton "en Arrière", *Back Button*
    - F = Bouton "en avant", *Forward Button*
  - Action:
    - PRESS = presser le bouton
    - CLICK = cliquer le bouton
    - DOUBLECLICK = double cliquer le bouton
    - HOLD = maintenir le bouton
  - 3ieme paramètre:
    - Si le second paramètre (Action) est "HOLD", un 3ième paramètre optionnel permet de spécifier combien de temps le bouton doit être maintenu pressé (en millisecondes).
- Aucune: Aucune

```
# Double cliquer le bouton gauche
AT+BLEHIDMOUSEBUTTON=L,doubleclick
OK

# Presser le bouton gauche, bouger la souris, puis relâcher le bouton
# Cela permet de réaliser une opération 'glisser-déposer'
AT+BLEHIDMOUSEBUTTON=L
OK
AT+BLEHIDMOUSEMOVE=-100,50
OK
AT+BLEHIDMOUSEBUTTON=0
OK
```

```
# Maintient le bouton "en arrière" de la souris pendant 200 milli-secondes (Dépendant de l'OS)
AT+BLEHIDMOUSEBUTTON=B,hold,200
OK
```

AT+BLEHIDCONTROLKEY

Envoi une commande HID de contrôle Media vers le périphérique lié (modification du volume, luminosité de l'écran, sélection de la piste musicale, etc.).

- Codebase Revision: 0.6.6
- Paramètre: Le code de la touche HID (de contrôle multimédia) à envoyer, suivi du délai pendant lequel il faut maintenir le bouton enfoncé (ms). Voir les valeurs ci-dessous.
- Sortie: Aucune.

Le code de la touche multimédia à envoyer:

- Contrôle système (fonctionne avec la plupart des systèmes)
  - BRIGHTNESS+ : augmentation de la luminosité
  - BRIGHTNESS- : diminution de la luminosité
- Contôle des médias (fonctionne avec la plupart des systèmes)
  - PLAYPAUSE : jouer - pause
  - MEDIANEXT : média/piste suivante
  - MEDIAPREVIOUS : média/piste précédente
  - MEDIASTOP : média arrêt
- Contrôle du volume (fonctionne avec la plupart des systèmes)
  - VOLUME
  - MUTE : muet
  - BASS : sons graves
  - TREBLE : sons aigus
  - BASS_BOOST : Booster les bass
  - VOLUME+ : augmenter le volume
  - VOLUME- : diminuer le volume
  - BASS+ : augmenter les basses
  - BASS- : diminuer les basses
  - TREBLE+ : augmenter les aigus
  - TREBLE- : diminuer les aigus
- Démarrage d'application (Windows 10 uniquement)
  - EMAILREADER : lecture des e-mails
  - CALCULATOR : calculatrice
  - FILEBROWSER : navigateur de fichier
- Contrôle de navigateur/explorateur de fichier (Firefox sous Windows/Android uniquement)
  - SEARCH : Rechercher
  - HOME : Page d'accueil / répertoire racine
  - BACK : en arrière
  - FORWARD : en avant
  - STOP : arrêt
  - REFRESH : rafraîchir
  - BOOKMARKS : marque-pages

Vous pouvez également envoyer une valeur hexadécimale brute (16 bits) au format '0xABCD'. Une liste complète des code HID 16 bit (*HID Consumer Control Key Codes*) peut être trouvé ici http://www.freebsddiary.org/APC/usb_hid_usages.php (section 12).

Si vous n'êtes pas appairé et connecté sur un périphérique centrale alors la commande retournera ERROR. Assurez vous d'être appairé et d'avoir activé le support HID avant d'avoir exécuté ces commandes.

```
# Mettre le volume du périphérique appairé en muet
AT+BLEHIDCONTROLKEY=MUTE
OK

# Maintenir la touche VOLUME+ enfoncée pendant 500ms
AT+BLEHIDCONTROLKEY=VOLUME+,500
OK

# Envoyer un code touche HID (valeur 16-bit brute). 0x006F = Brightness+
AT+BLEHIDCONTROLKEY=0x006F
OK
```

AT+BLEHIDGAMEPADEN

Active le support du service HID pour le joystick/gamepad. Le gamepad HID est désactivé par défaut (depuis la version 0.7.6 du firmware) parce qu'il cause des problème sur iOS et OS X et devrait uniquement être utilisé avec les périphériques Android et Windows.

- Codebase Revision: 0.7.6
- Paramètres: indique si le service gamepad devrait être activé ou désactivé.
  - **1** ou **on**
  - **0** ou **off**
- Sortie: Lorsque la commande est exécutée sans paramètre, l'interpréteur de commande retourne une valeur numérique indiquant l'état d'activation du service. 1 = activé ou 0 = désactivé.

> Cette commande nécessite une réinitialisation système (ATZ) pour que la modification soit effective.

AT+BLEHIDGAMEPAD

Envoi une commande HID gamepad via BLE

- Codebase Revision: 0.7.0
- Paramètres: Les paramètres suivants (séparés par une virgule) sont disponibles:
  - **Axe x**: GAUCHE ou DROITE. Si X=-1 alors c'est le bouton 'GAUCHE' qui est pressé. Si X=1 alors c'est le bouton de 'DROITE' qui est pressé. Si X=0 alors aucun des deux boutons (gauche ou droite) est pressé
  - **Axe y**: HAUT ou BAS. Si Y=-1 alors c'est le bouton 'HAUT' qui est pressé. Si Y=1 ALORS KES LE BOUTON 'BAs' qui est pressé. Si Y=0 alors aucun des deux boutons (haut ou bas) est pressé
  - **Boutons**: de 0x00 à 0xFF, qui est un octet dont les différents bits indiquent lequels des 8 boutons sont pressés (boutons de 0-7)
- Output: aucune

> Le gamepad HID est désactivé par défaut depuis la version version 0.7.6 du firmware. Le service HID Gamepad doit être préalablement activé à laide de la commande AT+BLEHIDGAMEPADEN=1 avant de pouvoir l'utiliser.

> Note: Il est nécessaire d'envoyer les deux événements 'pressé' et 'relâché' pour chaque bouton utilisé. A défaut, le bouton restera "pressé" jusqu'à la réception de la commande de "relâchement" du bouton.

```
# Presser 'DROITE' et le 'Bouton0' en même temps
# Boutons = 0b00000001 en binaire -> 1 en décimal -> 0x01 en hexadécimal
AT+BLEHIDGAMEPAD=1,0,0x01

# Presser 'HAUT' et 'Bouton1' + 'Bouton0' en même temps
# Boutons = 0b00000011 en binaire -> 3 en décimal -> 0x03 en hexadécimal
AT+BLEHIDGAMEPAD=0,-1,0x03
```

AT+BLEMIDIEN

Active ou désactive le service MIDI ble.

- Codebase Revision: 0.7.0
- Paramètre: Activer / Désactiver le service MIDI BLE avec les valeurs suivante:
  - **1** ou **on** - pour activer
  - **0** ou **off** - pour désactiver
- Sortie: Lorsqu'elle est exécutée sans paramètre, l'interpréteur de commande retourne une valeur numérique correspondant à l'état d'activation du service MIDI BLE. 1 = activé ou 0 = désactivé.

> Note: cette commande nécessitera une réinitialisation (ATZ) pour que la modification soit prise en compte.

```
# Vérifier l'état actuel du service MIDI
AT+BLEMIDIEN
1
OK

# Activer le service MIDI
AT+BLEMIDIEN=1
```

```
OK
```

AT+BLEMIDIRX

Lit la mémoire tampon et retourne le tableau de caractère MIDI entrant.

- Codebase Revision: 0.7.0
- Paramètre: Aucun
- Sortie: l'événement midi retourné sous forme de tableau d'octet

```
AT+BLEMIDIRX
90-3C-7F
OK
```

AT+BLEMIDITX

Envoyer un événement MIDI vers l'hôte.

- Codebase Revision: 0.7.0
- Paramètre: l'événement MIDI sous forme d'un tableau hexadécimal, qui peut être soit:
  - Une série d'événement MIDI complet (jusqu'à 4 événements)
  - 1 événement MIDI complet (exactement) + plusieurs événements **running events** sans statut (*several running events without status*) (jusqu'à 7)
- Sortie: Aucune

```
# Envoi de un événement (middle C/DO grave avec la vélocité maximale)
AT+BLEMIDITX=90-3C-7F
OK

# Envoi de 2 événements
AT+BLEMIDITX=90-3C-7F-A0-3C-7F
OK

# Envoi d'un événement complet + "running event"
AT+BLEMIDITX=90-3C-7F-3C-7F
OK
```

AT+BLEBATTEN

Active le service Battery en suivant la définition de "Bluetooth SIG".

Le Service Battery https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.battery_service.xml permet d'exposer l'état de la batterie au périphérique appairé.

- Codebase Revision: 0.7.0
- Paramètre: Indique si le service Battery doit être activé ou désactivé:
  - **1** ou **on** - activer le service
  - **0** ou **off** - désactiver le service
- Sortie: Si la commande est exécutée sans paramètre alors l'interpréteur de commande retourne une valeur numérique indiquant l'état d'activation du service. Activer = 1, Désactiver = 0.

> Nécessite une réinitialisation système (ATZ) pour que la modification soit prise en compte.

AT+BLEBATTVAL

Permet au module d'envoyer le niveau de sa batterie au périphérique central (votre smartphone).

- Codebase Revision: 0.7.0
- Paramètre: Le pourcentage (0..100) de la batterie à communiquer.
- Sortie: Si la commande est exécutée sans paramètre, l'interpréteur de commande retourne le niveau de batterie actuel tel qu'il à été stocké dans les caractéristiques.

```
# Fixer le niveau de la batterie à 72%
AT+BLEBATTVAL=72
OK
```

GAP BLE

Sommaire

BLE GAP

GAP https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap (*Adafruit, Anglais*) qui signifie *Generic Access Profile* gouverne l'annonce et les connexions avec les périphériques Bluetooth Low Energy.

Les commandes suivantes peuvent être utilisées pour configurer les paramètres GAP sur le module BLE.

Vous pouvez utiliser ces commandes pour modifier les données d'annonce (*advertising data*). Par exemple: le nom du périphérique tel qu'il doit apparaître sur votre smartphone durant le processus d'appairage. Les commandes permettent également de récupérer des informations concernant la connexion qui a été établie entre deux périphériques, ou la déconnexion si vous ne désirez pas maintenir une connexion.

AT+GAPCONNECTABLE

Cette commande est utilisé pour rendre le périphérique 'non connectable'.

- Codebase Revision: 0.7.0
- Paramètre: Indique si le périphérique doit (ou ne doit pas) s'annoncer comme "connectable". Utiliser une des valeurs suivantes:
  - **1** ou **yes** - s'annoncer comme connectable
  - **0** ou **no** - ne pas s'annoncer comme connectable
- Sortie: Afficher l'état 'connectable' du périphérique si la commande est appelée sans paramètre

```
# Rendre le périphérique non-connectable (avertissement uniquement)
AT+GAPCONNECTABLE=0
OK

# Vérifier le statut "connectable" actuel du module.
AT+GAPCONNECTABLE
1
OK
```

AT+GAPGETCONN

Affiche le statut actuel de la connexion (si nous sommes connecté sur un autre périphérique BLE ou non).

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: 1 si connecté, sinon 0

```
# Connecté
AT+GAPGETCONN
1
OK

# Non connecté
AT+GAPGETCONN
0
OK
```

AT+GAPDISCONNECT

Déconnecter le module du périphérique centrale (smartphone).

- Codebase Revision: 0.3.0
- Paramètre: Aucun
- Sortie: Aucun

```
AT+GAPDISCONNECT
OK
```

AT+GAPDEVNAME

Obtenir ou modifier le nom du module, celui qui est inclus dans le *payload* d'annonce du module Bluefruit LE

- Codebase Revision: 0.3.0
- Paramètre:
  - Aucun pour lire le nom actuel du périphérique
  - Le nouveau nom du périphérique si vous voulez changer son nom
- Sortie: Le nom actuel du périphérique si la commande est exécutée en mode lecture.

Le nom du module est stocké dans la mémoire non volatile. Ce nouveau nom est donc persistant. La modification est effective dès la réinitialisation du système (ATZ). Pour réinitialiser le périphérique aux valeurs d'usine et effacer les données de configuration, il faut utiliser la commande AT+FACTORYRESET

```
# Lecture du nom actuel du périphérique
AT+GAPDEVNAME
UART
OK

# Changer le nom du module BLE à 'BLEFriend'
AT+GAPDEVNAME=BLEFriend
OK
# Réinitialisation système pour appliquer les modifications
ATZ
OK
```

AT+GAPDELBONDS

Efface les informations d'appairage stockés dans le module Bluefruit LE.

- Codebase Revision: 0.3.0
- Paramètres: Aucun
- Sortie: Aucun

```
AT+GAPDELBONDS
OK
```

AT+GAPINTERVALS

Retrouver ou modifier les différentes intervalles d'annonce et de connexion pour le module Bluefruit LE.

Soyez extrêmement attentif en utilisant cette commande puisque vous pouvez facilement provoquer des problèmes en modifiant les intervalles. En effet, certains téléphones mobile pourrait ne plus reconnaître le module ou refuser de s'y connecter.

- Codebase Revision: 0.3.0
- Paramètres: Si vous désirez modifier les intervalles GAP, les données doivent être saisi dans l'ordre suivant (séparés avec des virgules):
  - Minimum connection interval (Intervalle de connexion minimal, en millisecondes)
  - Maximum connection interval (Intervalle de connexion maximal, en millisecondes)
  - Fast Advertising interval (Intervalle d'annonce rapide, en millisecondes)
  - Fast Advertising timeout (Timeout d'annonce rapide, en millisecondes)
  - >= 0.7.0: Low power advertising interval (Intervalle d'annonce en mode d'économie d'énergie, en millisecondes), 417.5 ms par défaut

Afin d'économisé l'énergie, les modules Bluefruit passent en mode "*lower advertising rate*" (faible débit d'annonce) après avoir atteind le temps "*fast advertising timeout*" (en secondes). La valeur par défaut est de 30 secondes (*Fast Advertising Timeout*). L'intervalle "*low power advertising interval*" est hard-codé à environ 0.6s dans les firmwares avant < 0.7.0. La support du paramétrage "*low power interval*" à été ajouté dans le firmware 0.7.0 via un 5ieme paramètre optionnel.

Voici les différentes limitations min et max pour les paramètres GAP:

- **Absolute minimum connection interval** (minimum absolu pour l'intervalle de connexion): 10ms
- **Absolute maximum connection interval** (maximum absolu pour l'intervalle de connexion) : 4000ms
- **Absolute minimum fast advertising interval** (minimum absolu pour l'intervalle d'annonce rapide): 20ms
- **Absolute maximum fast advertisting interval** (maximum absolu pour l'intervalle d'annonce rapide): 10240ms
- **Absolute minimum low power advertising interval** (minimum absolu pour l'intervalle annonce en économie d'énergie): 20ms
- **Absolute maximum low power advertising interval** (maximum absolu pour l'intervalle annonce en économie d'énergie): 10240ms

Si vous désirez uniquement modifier un intervalle alors laissez les autres valeurs - entre les virgules- vides (ex: ",,110," modifiera uniquement la 3ième valeur "advertising interval").

- Sortie: en lisant les paramètres d'intervalle GAP, les informations suivantes seront affichées (séparés par des virgules):
  - Minimum connection interval (interval de connexion minimum, en milisecondes)
  - Maximum connection interval (interval de connexion maximum, en milisecondes)
  - Advertising interval (interval de connexion maximum, en milisecondes)
  - Advertising timeout (timeout d'annonce, en milisecondes)

La modification des intervalles GAP est enregistré dans la mémoire non-volatile et les nouvelles valeurs seront utilisées après la réinitialisation du système. Pour réinitialiser le périphérique aux paramètres d'usine (et effacer vos paramètres de la mémoire), vous pouvez utiliser la commande AT+FACTORYRESET

```
# Lecture des différents intervales GAP
AT+GAPINTERVALS
20,100,100,30

# Modification de tous les intervalles
AT+GAPINTERVALS=20,200,200,30
OK

# Modifier uniquement l'intervalle d'annonce (advertising interval)
AT+GAPINTERVALS=,,150,
OK
```

AT+GAPSTARTADV

Force le module Bluefruit LE commencer l'émission des paquets d'annonce si ce n'est pas déjà le cas (et en partant du principe qu'il n'est pas déjà connecté sur un périhiphérique externe).

- Codebase Revision: 0.3.0
- Paramètre: Aucun
- Sortie: Aucun

```
# Résultat de la commande lorsque le module n'émet pas de paquet d'annonce
AT+GAPSTARTADV
OK

# Résultat de la commande lorsque le Bluefruit émet déjà des paquets d'annonce
AT+GAPSTARTADV
ERROR

# Résulatat de la commande lorque le module est déjà connecté sur un autre périphérique
AT+GAPSTARTADV
ERROR
```

AT+GAPSTOPADV

Arrête l'émission des paquets d'annonce depuis le module Bluefruit LE.

- Codebase Revision: 0.3.0
- Paramètre: Aucun
- Sortie: Aucun

```
AT+GAPSTOPADV
OK
```

AT+GAPSETADVDATA

Surcharge les données du payload d'annonce avec le tableau d'octet spécifié en paramètre (remplace les données normalement utilisées dans le payload d'annonce normal). Le payload d'annonce du module suit les guidelines dans les Spécifications Core Bluetooth 4.0 or 4.1 https://www.bluetooth.org/en-us/specification/adopted-specifications .

En particulier, les **Core Specification Supplement (CSS) v4** contient des détails concernant les champs de données d'annonce habituels comme les 'Flags/drapeaux' (Partie A, Section 1.3) et une liste des UUID des différents services (Partie A, Section 1.1). Une liste de tous les types de donnée GAP possible est disponible sur la page Bluetooth SIG's Generic Access Profile https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile .

Le payload d'annonce est constitué de données GAP (*Generic Access Profile*) https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile insérées dans un paquet d'annonce au format: [U8:LEN] [U8:Data Type Value] [n:Value]

	***ATTENTION : cette commande nécessite un certain degré de connaissances concernent les détails bas niveau des spécifications "Bluetooth 4.0 or 4.1 Core". Cette commande ne devrait être utilisée que par des utilisateurs expérimentés. Une mauvaise utilisation de cette commande rendra votre module indétectable par le périphérique central (votre SmartPhone).***
	***ATTENTION : Cette commande remplace le payload d'annonce et peut empêcher certains services de fonctionner comme attendu.***
	Vous pouvez utiliser la commande AT+FACTORYRESET pour restaurer le payload normal (celui par défaut).

Par exemple : pour insérer le type de donnée drapeau (*flags*, valeur "Data Type" 0x01) et utiliser la valeur 0x06/0b00000110 ("BR/EDR Not Supported" et "LE General Discoverable Mode"), il faut utiliser le tableau de données suivant:

```
02-01-06
```

- 0x02 indique le nombre d'octets dans l'entrée
- 0x01 indique la valeur 'type de donnée' (*Data Type Value*) qui indique que c'est un drapeau (*Flag*).
- 0x06 (0b00000110) est la valeur du drapeau (*Flag*). Il active les éléments suivants (voir les spécifications Bluetooth Core 4.0, Volume 3, Part C, 18.1):
  - **LE General Discoverable Mode** (tout le monde peut découvrir ce périphérique)
  - **BR/EDR Not Supported** (C'est un périphérique Bluetooth Low Energy uniquement)

Si nous désirons également inclure deux UUIDs de services dans les informations d'annonces (pour que les périphériques distants savent que le module supporte ces services) alors nous pouvons ajouter les données suivantes au tableau d'octets:

```
05-02-0D-18-0A-18
```

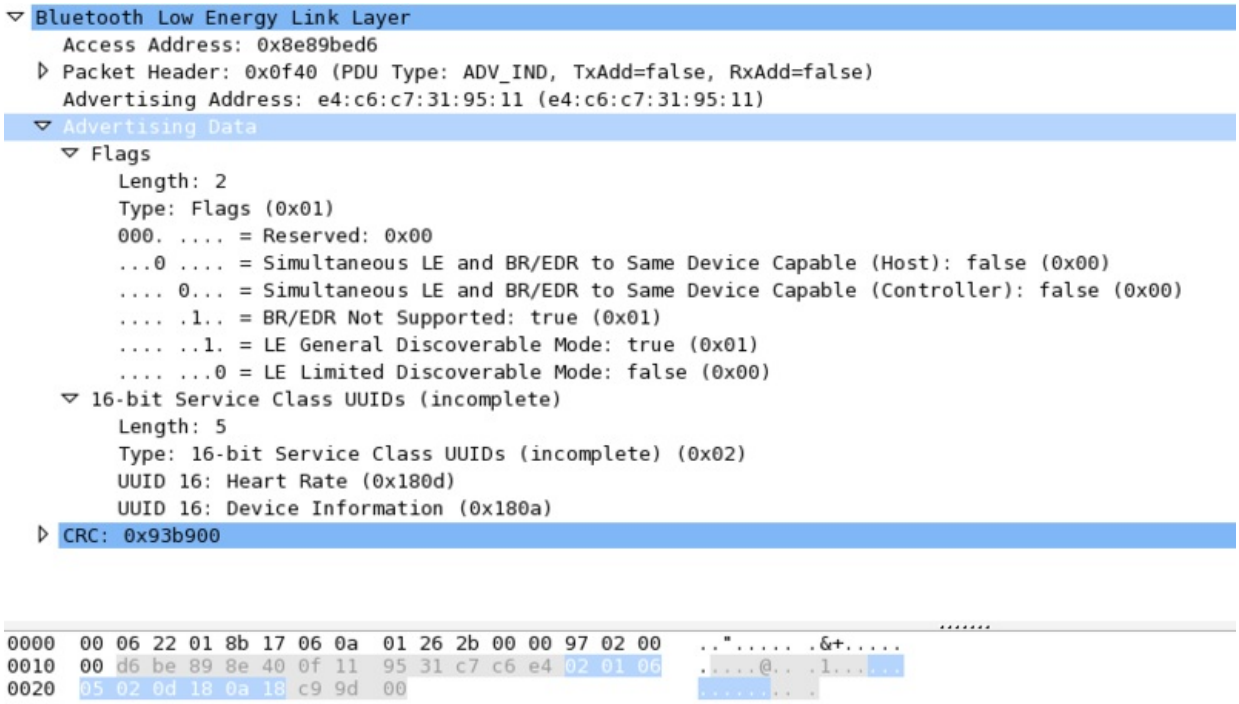
- 0x05 indique le nombre d'octets en entrée (5)
- 0x02 est la valeur indiquant le type de donnée *Data Type Value* https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile . Indique qu'il s'agit d'une liste incomplète de service UUIDs (**Incomplete List of 16-bit Service Class UUIDs**)
- 0x0D 0x18 est le premier UUID 16 bits (que l'on converti en **0x180D**) et correspond au service Heart Rate https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.heart_rate.xml .
- 0x0A 0x18 est un autre UUID 16 bits (que l'on converti en **0x180A**) et correspondant au service Device Information https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.device_information.xml .

	***Inclure les UUIDs des services est important puisque certaines applications mobiles ne fonctionnerons qu'avec des périphériques annonçant un service UUID spécifique dans les paquets d'annonce. C'est par exemple le cas pour la plupart des applications de Nordic Semiconductors.***

- Codebase Revision: 0.3.0
- Paramètres: le tableau d'octets brute qui devrait être inséré dans la section de donnée du bloc d'annonce. Veuillez à rester à l'intérieur de l'espace définit dans les spécifications Bluetooth Core.
- Réponse: Aucune

```
# S'annoncer comme Découvrable et BLE uniquement avec
# les UUIDs 16 bits 0x180D et 0x180A
AT+GAPSETADVDATA=02-01-06-05-02-0d-18-0a-18
OK
```

Le résultat de cette commande est visible sur la capture d'écran ci-dessous. Capture obtenue à l'aide d'un sniffer capturant et analysant les paquets d'annonce dans Wireshark. Les données d'annonce (dans le payload) sont visibles en surbrillance (en bleu, dans le tableau d'octets en bas de l'image et l'analyse du paquet dans la section supérieure):



GATT BLE

Sommaire

GATT BLE

GATT https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt est l'acronyme de "Generic ATTribute Profile" (profile d'attribut générique). GATT gouverne l'organisation et l'échange de données entre les périphériques connectés. Un périphérique (nommé "*the peripheral*" dans la nomenclature) agit comme un serveur GATT qui enregistre les données dans des enregistrements Attributs. Un second périphérique (nommé la "central") agit comme un client GATT, demandant les données au serveur lorsque cela est nécessaire.

Les commandes suivantes peuvent être utilisées pour créer des services et caractéristiques GATT personnalisés sur le BLEFriend. Ces services et caractéristiques sont utilisés pour enregistrer et échanger les données.

Attention: toutes les caractéristiques que vous aller définir seront sauvegardées dans la mémoire FLASH de configuration (non volatile) du périphérique. Ces caractéristiques seront réinitialisées à chaque démarrage du périphérique.

Il est nécessaire d'effectuer une réinitialisation système avec un 'ATZ' avec la plupart des commandes ci-dessous pour que la modification soit effective!

Limitations de GATT

Etant donné la mémoire SRAM et les ressources limitées du module, les commandes ont certaines limitations détaillées ci-dessous. Gardez ces limitations en mémoire lorsque vous créez des services et caractéristiques GATT personnalisés.

Ces valeurs sont applicable à partir du firmware firmware 0.7.0:

- Nombre maximal de services: 10
- Nombre maximal de caractéristiques: 30
- Taille de la mémoire tampon Max pour chaque caractéristique: 32 octets
- Nombre maximum de CCCDs: 16

Vous pouvez utiliser la commande de réinitialisation d'usine (**AT+FACTORYRESET**) si vous avez besoin d'effacer les valeurs d'une précédente configuration. Saisissez la commande avant de débuter une nouvelle configuration.

AT+GATTCLEAR

Efface tous les services et caractéristiques GATT personnalisés qui ont été définis sur le périphérique.

- Codebase Revision: 0.3.0
- Paramètres: Aucune
- Réponse: Aucune

```
AT+GATTCLEAR
OK
```

AT+GATTADDSERVICE

Ajoute une nouvelle définition de service personnalisé sur le périhiphérique.

- Codebase Revision : 0.3.0
- Paramètre : La commande accepte une série de paires "clé-valeur", paires séparés par des virgules. Ces paires sont utilisées pour définir les propriétés du service.
  Les paires clé-valeur suivantes peuvent être utilisés:
  - **UUID** : Le UUID 16-bits pour ce service. La valeur 16-bits doit être mentionnée au format hexadécimal (0x1234).
  - **UUID128** : Le UUID 128-bits pour ce service. Les valeurs 128-bits doivent avoir le format suivant: 00-11-22-33-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF
- Réponse : La valeur d'index du service GATT personnalisé dans la table lookup. (Il est important de mémoriser cette valeur indexe pour pouvoir travailler avec le service.)

Note: Les valeurs des clés ne sont pas sensible à la case

Un seul type de UUID peut être saisi pour le service (soit UUID ou soit UUID128)

```
# Efface toutes les définitions antérieurs
# de services/caractéristiques personnalisées
AT+GATTCLEAR
OK

# Ajoute le service "battery" (UUID = 0x180F, état de la
# batterie) sur le périphérique
AT+GATTADDSERVICE=UUID=0x180F
1
OK

# Ajoute la caractéristique "battery measurement"
# (UUID = 0x2A19, mesure de la batterie), notification activée
AT+GATTADDCHAR=UUID=0x2A19,PROPERTIES=0x10,MIN_LEN=1,VALUE=100
1
OK
```

AT+GATTADDCHAR

Ajouter une caractéristique personnalisée au dernier service ajouté sur le périphérique (via AT+GATTADDSERVICE).

AT+GATTADDCHAR doit être exécuté APRES AT+GATTADDSERVICE et ajoutera une nouvelle caractéristique à la dernière définition de service ajouté sur le périphérique.

Depuis la version 0.6.6 du firmware, il est également possible d'utiliser des UUIDs 128-bit personnalisés avec cette commande. Voyez les exemple ci-dessous pour plus d'information.

- Codebase Revision : 0.3.0
- Paramètres : cette commande accepte un ensemble de paires clé-valeur (séparées par des virgules) pour définir les propriétés de la caractéristique. Les paires clé-valeur suivantes peuvent être utilisés:
  - **UUID** : L'UUID 16-bit de la caractéristique (qui sera inséré dans les 3ième et 4ièmes octets des UUID 128-bit des services parent). Cette valeur devrait être encodée au format hexadécimal au format (ex. 'UUID=0x1234'). Cette valeur doit être unique, et ne doit pas entrer en conflit avec les octets 3+4 de l'UUID 128-bit du service parent.
  - **PROPERTIES** : Le champs propriétés 8-bit de la caractéristique, tel que définit dans Bluetooth SIG. Les valeurs suivantes peuvent être utilisées:
    - 0x02 - Read (lecture)
    - 0x04 - Write Without Response (écriture sans réponse)
    - 0x08 - Write (écriture)
    - 0x10 - Notify (notification)
    - 0x20 - Indicate (désigner)
  - **MIN_LEN** : La taille minimale des valeurs pour cette caractéristique (en octets, min = 1, max = 20, défaut = 1)
  - **MAX_LEN** : La taille minimale des valeurs pour cette caractéristique (in octets, min = 1, max = 20, défaut = 1)
  - **VALUE** : Valeur initiale assignée à la caractéristique (dans les limites de 'MIN_LEN' et 'MAX_LEN'). La valeur peut être un entier ("-100", "27"), un hexadécimal ("0xABCD"), ou un tableau d'octet ("aa-bb-cc-dd") ou une chaâine de caractère ("GATT!").
  - **>=0.7.0 - DATATYPE** : cet argument indique le type de données stockée dans la caractéristique et est utilisé pour aider le parsing des données. DATATYPE peut avoir l'une des valeurs suivantes:
    - AUTO (0, défaut)
    - STRING (1, chaîne de caractère)
    - BYTEARRAY (2, tableau d'octets)
    - INTEGER (3, entier)
  - **>=0.7.0 - DESCRIPTION**: Ajoute la chaîne de caractère en argument comme description (*description entry*) de la

caractéristique
    - **>=0.7.0 - PRESENTATION**: Ajoute la chaîne de caractère en argument comme format de présentation (*presentation format entry*) de la caractéristique.
- **Réponse**: La valeur d'index de la caractéristique dans la table lookup des caractéristiques GATT personnalisées. (Il est important de mémoriser cette index pour travailler ensuite avec les caractéristiques.)

Note: Les valeurs des clés ne sont pas sensible à la case

Assurez-vous de l'unicité du UUID 16-bit et qu'il n'entre pas en conflit avec les octets 3+4 de l'UUID service 128-bit

```
# Efface toutes les définitions antérieurs
# de services/caractéristiques personnalisées
AT+GATTCLEAR
OK

# Ajoute le service "battery" (UUID = 0x180F, état de la
# batterie) sur le périphérique
AT+GATTADDSERVICE=UUID=0x180F
1
OK

# Ajoute la caractéristique "battery measurement"
# (UUID = 0x2A19, mesure de la batterie), notification activée
AT+GATTADDCHAR=UUID=0x2A19,PROPERTIES=0x10,MIN_LEN=1,VALUE=100
1
OK
```

```
# Efface toutes les définitions antérieurs
# de services/caractéristiques personnalisées
AT+GATTCLEAR
OK

# Ajoute un service personnalisé sur le périphérique
AT+GATTADDSERVICE=UUID128=00-11-00-11-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF
1
OK

# Ajouter une caractéristique personnalisée au
# service ci-dessus (s'assurer qu'il n'y a pas de conflit entre
# le UUID 16-bit et les octets 3+4 d'un UUID service 128-bit)
AT+GATTADDCHAR=UUID=0x0002,PROPERTIES=0x02,MIN_LEN=1,VALUE=100
1
OK
```

La version **0.6.6** du firmware de Bluefruit LE est capable d'utiliser le drapeau "nouveau **UUID128**" pour ajouter un des UUIDs 128-bit personnalisés qui ne sont pas relatifs à un service UUID parent (qui est utilisé lorsque l'on passe un drapeau "**UUID** 16-bit ").

Pour spécifier un UUID 128-bit pour votre caractéristique personnalisée, saisissez une valeur ressemblant à la syntaxe suivante:

```
# Ajouter une caractéristique personnalisée
# au service définit ci-dessous (en utilisant un
# UUID 128-bit personalisé)
AT+GATTADDCHAR=UUID128=00-11-22-33-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF,PROPERTIES=0x02,MIN_LEN=1,VALUE=100
1
OK
```

La version **0.7.0** du firmware de Bluefruit LE ajoute les nouveau mots clés **DESCRIPTION** et **PRESENTATION**, correspondant aux spécifications GATT Characteristic User Description https://developer.bluetooth.org/gatt/descriptors/Pages/DescriptorViewer.aspx?u=org.bluetooth.descriptor.gatt.characteristic_user_description.xml et Characteristic Presentation Format https://developer.bluetooth.org/gatt/descriptors/Pages/DescriptorViewer.aspx?u=org.bluetooth.descriptor.gatt.characteristic_presentation_format.xml .

Le champs **DESCRIPTION** est une chaîne de caractère qui contient une courte description textuelle de la caractéristique. Certaines Apps pourraient ignorer cette information, elle devrait néanmoins être visible en utilisant l'application "Master Control Panel" de Nordic sous iOS et Android.

Le champs **PRESENTATION** contient un payload 7-octets qui encapsule les données *presentation format* de la caractéristique. Il nécessite en ensemble d'octets spécifique pour fonctionner correctement. Voyez le lien suivant pour plus de détails sur le format du payload: https://developer.bluetooth.org/gatt/descriptors/Pages/DescriptorViewer.aspx?u=org.bluetooth.descriptor.gatt.characteristic_presentation_format.xml

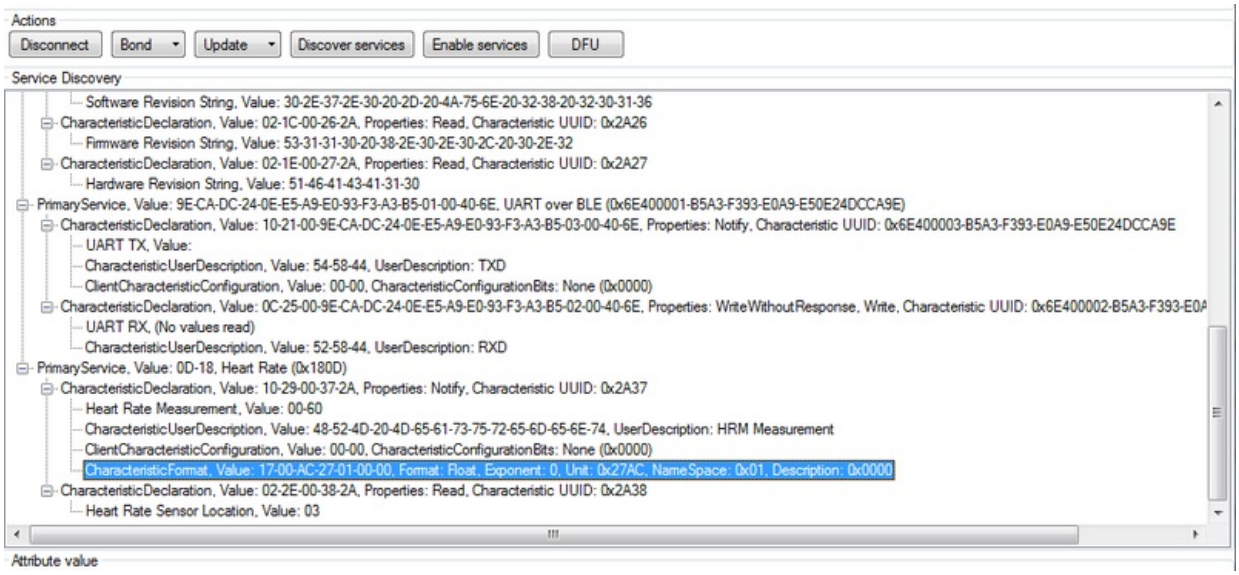
L'exemple suivant montre comment vous pouvez utiliser ces deux nouveaux champs:

```
AT+GATTADDCHAR=UUID=0x2A37, PROPERTIES=0x10, MIN_LEN=2, MAX_LEN=3, VALUE=00-40,
        DESCRIPTION=HRM Measurement, PRESENTATION=17-00-AC-27-01-00-00
```

Pour le format de présentation de la caractéristique nous avons:

- Format = IEEE-11073 32-bit FLOAT (Decimal 23, Hex 0x17)
- Exposant = 0/aucun
- Unité = température thermodynamique (Thermodynamic temperature): Degrés Fahrenheit (0x27AC) - Bluetooth LE Unit List
- Espace de nom/Namespace = Bluetooth SIG Assigned Number (0x01)
- Description = Aucune (0x0000)

L'application "Master Control Panel" de Nordic affiche l'information suivante:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

AT+GATTCHAR

Fixer ou retrouver la valeur d'une caractéristique GATT personnalisée personnalisée (basé sur le numéro d'index *index ID* retourné par AT+GATTADDCHAR lorsque la caractéristique à été ajouté sur le périphérique).

- Codebase Revision: 0.3.0
- Paramètre: cette fonction prend un ou deux paramètres (séparé par une virgule, un paramètre = lecture, deux paramètres = écriture).
  - Le premier paramètre est l'index de la caractéristique, tel que retourné par la fonction AT+GATTADDCHAR. Ce paramètre est toujours nécessaire et s'il n'y a pas de second paramètres alors la valeur courante de la caractéristique est retournée.
  - Le second paramètre (optionnel) est la nouvelle valeur à assigner à la caractéristique (dans les limites définies par MIN_SIZE et MAX_SIZE lorsque la caractéristique à été créée).
- Réponse:
  - Si la commande est utilisée en mode lecture (avec l'index de la caractéristique comme seul paramètre) alors la réponse affichera la valeur actuelle de la caractéristique.
  - Si la commande est utilisée en mode écriture (avec index de la caractéristique + une virgule + la nouvelle valeur) alors la caractéristique sera mise-à-jour pour utiliser la nouvelle valeur.

```
# Efface toutes les définitions antérieurs
# de services/caractéristiques personnalisées
AT+GATTCLEAR
OK

# Ajoute le service "battery" (UUID = 0x180F, état de la
# batterie) sur le périphérique
AT+GATTADDSERVICE=UUID=0x180F
1
OK

# Ajoute la caractéristique "battery measurement"
# (UUID = 0x2A19, mesure de la batterie), notification activée
AT+GATTADDCHAR=UUID=0x2A19,PROPERTIES=0x10,MIN_LEN=1,VALUE=100
1
OK

# Lire la caractéristique "battery measurement" (index ID = 1)
AT+GATTCHAR=1
0x64
OK

# Modifier la caractéristique "battery measurement" vers 32 (hex 0x20)
```

```
AT+GATTCHAR=1,32
OK

# Vérifier la valeur écrite
AT+GATTCHAR=1
0x20
OK
```

AT+GATTLIST

Liste tous les services et caractéristiques GATT personnalisés qui ont été définis sur le périphérique.

- Codebase Revision: 0.3.0
- Paramètre: Aucun
- Réponse: Une liste de tous les services personnalisé et toutes les caractéristiques définie sur le périphérique.

```
# Efface tous les services/caractéristiques personnalisés
AT+GATTCLEAR
OK

# Ajouter le service "battery" (UUID = 0x180F) sur le périphérique
AT+GATTADDSERVICE=UUID=0x180F
1
OK

# Ajouter la caractéristique "battery measurement" (UUID = 0x2A19), notification activée
AT+GATTADDCHAR=UUID=0x2A19,PROPERTIES=0x10,MIN_LEN=1,VALUE=100
1
OK

# Ajouter un service personalisé sur le périphérique
AT+GATTADDSERVICE=UUID128=00-11-00-11-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF
2
OK

# Ajouter une caractéristique personnalisé sur le service ci-dessus
# (S'assurer qu'il n'y a pas de conflit entre l'UUID 16 bits et
# les octets 3+4 sur UUID service 128-bit)
AT+GATTADDCHAR=UUID=0x0002,PROPERTIES=0x02,MIN_LEN=1,VALUE=100
2
OK

# Obtenir une liste de tous les services GATT personnalisés et
# de toutes les caractéristiques personnalisées sur le périphérique
AT+GATTLIST
ID=01,UUID=0x180F
  ID=01,UUID=0x2A19,PROPERTIES=0x10,MIN_LEN=1,MAX_LEN=1,VALUE=0x64
ID=02,UUID=0x11, UUID128=00-11-00-11-44-55-66-77-88-99-AA-BB-CC-DD-EE-FF
  ID=02,UUID=0x02,PROPERTIES=0x02,MIN_LEN=1,MAX_LEN=1,VALUE=0x64
OK
```

AT+GATTCHARRAW

Cette commande en lecture seule lit les données binaires d'une caractéristique (au lieu de ASCII. L'information n'est pas affichable mais représente moins de surcharge et facilite l'écriture de bibliothèques pour Arduino (ou autre microcontroleur).

- Codebase Revision: 0.7.0
- Paramètre: Le numéro d'index de la caractéristique (*index ID*) dont on désire les données
- Sortie: les données binaire correspondant à la caractéristique.

Note : Il s'agit d'une commande spécialisée et il n'y a pas de caractère NEWLINE en fin de commande!

Débogage

Sommaire

A propos du débogage

Cette section expose les commandes disponibles pour faire du débogage avec les modules Bluefruit LE:

Utilisez ces commandes avec précaution car elles peuvent facilement conduire à des plantages matériels (HardFault error) sur le coeur ARM, ce qui bloquera le périphérique (il ne répondra plus).

AT+DBGMEMRD

Affiche le contenu brut de la mémoire à l'adresse spécifié.

- Codebase Revision: 0.3.0
- Paramètre: Les paramètres suivants (séparés avec des virgules) peuvent être utilisés:
  - L'adresse de départ pour la lecture de la mémoire (sous la forme hexadécimale, avec ou sans être préfixé de '0x')
  - La taille du mot (*word*). Peut être 1, 2, 4 ou 8
  - Le nombre de mots à lire
- Sortie: le contenu brut de la mémoire au format hexadécimal en utilisant la longueur spécifiée et la taille du mot (voir les exemples ci-dessous pour plus de détails)

```
# Lit 12 valeurs de 1-octets à partir de 0x10000009
AT+DBGMEMRD=0x10000009,1,12
FF FF FF FF FF FF FF 00 04 00 00 00
OK

# Essaye de lire 2 valeurs de 4-octets à partir de 0x10000000
AT+DBGMEMRD=0x10000000,4,2
55AA55AA 55AA55AA
OK

# Essaye de lire 2 valeurs de 4-octets à partir de 0x10000009
# Provoquera une erreur parce que Cortex M0 ne peut pas faire
# de lecture non alignée, et toutes les valeurs autres que 8-bit
# doit démarrer une une adresse paire
AT+DBGMEMRD=0x10000009,4,2
MISALIGNED ACCESS
ERROR
```

AT+DBGNVMRD

Affiche le contenu brut de la section des données de configuration (de la mémoire non volatile).

- Codebase Revision: 0.3.0
- Propriété: Aucune
- Sortie: Les données brute de la mémoire de configuration (de la mémoire non volatile).

```
AT+DBGNVMRD
FE CA 38 05 00 03 00 00 01 12 01 00 55 41 52 54 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 14 00 64 00 64 00 1E 00 00 00 00 00 00 00 00 00 00 00
OK
```

AT+DBGSTACKSIZE

Retourne la taille actuelle de la pile. Aide à détecter le dépassement de pile ou détecter l'usage de la pile mémoire lorsque l'on optimise l'utilisation de la mémoire sur le système.

- Codebase Revision: 0.4.7
- Paramètre: None
- Sortie: la taille actuelle de la pile mémoire (en octets)

```
AT+DBGSTACKSIZE
1032
OK
```

AT+DBGSTACKDUMP

Fait un dumps du contenu de la pile. Les sections non utilisées de la pile sont remplies avec '0xCAFEFOOD' pour identifier plus facilement où s'arrête l'utilisation de la pile.

C'est une commande uniquement destinée au débogage et développement.

- Codebase Revision: 0.4.7
- Paramètre: None
- Sortie: Le contenu entier de la région mémoire contenant la pile

```
AT+DBGSTACKDUMP
0x20003800: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003810: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003820: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003830: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003840: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003850: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003860: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003870: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003880: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003890: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038A0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038B0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038C0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038D0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038E0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200038F0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003900: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003910: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003920: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003930: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003940: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003950: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003960: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003970: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003980: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003990: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039A0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039B0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039C0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039D0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039E0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x200039F0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A00: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A10: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A20: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A30: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A40: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A50: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A60: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A70: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A80: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003A90: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AA0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AB0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AC0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AD0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AE0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003AF0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B00: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B10: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B20: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B30: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B40: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B50: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B60: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B70: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B80: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003B90: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003BA0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
```

0x20003BB0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003BC0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003BD0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003BE0: CAFEF00D CAFEF00D CAFEF00D CAFEF00D
0x20003BF0: CAFEF00D CAFEF00D 00000000 CAFEF00D
0x20003C00: 00000004 20001D04 CAFEF00D FFFFEF68
0x20003C10: CAFEF00D 00001098 CAFEF00D CAFEF00D
0x20003C20: CAFEF00D CAFEF00D 00001006 200018D8
0x20003C30: 00000001 200018D8 20001C50 00000004
0x20003C40: 20001BB0 000134A5 0000100D 20001D28
0x20003C50: 00000006 00000006 20001C38 20001D44
0x20003C60: 20001C6C 20001D44 00000006 00000005
0x20003C70: 20001D38 00000005 20001D38 00000000
0x20003C80: 00000001 00012083 200018C8 000013D3
0x20003C90: 00550000 00000001 80E80000 4FC40000
0x20003CA0: 000080E8 00000009 60900000 000080E8
0x20003CB0: 60140000 20002764 0009608F 000080E8
0x20003CC0: 80000000 000080E8 00000000 00129F5F
0x20003CD0: 00000000 0001E4D9 80E80000 200018C8
0x20003CE0: 200018D4 00000000 80E80000 000000FF
0x20003CF0: 0000011C 0001BCE1 0000203A 0001BC1D
0x20003D00: 00000000 0001BC1D 80E80000 0001BCE1
0x20003D10: 0000011C 0001BDA9 80E80000 0001BDA9
0x20003D20: 0000011C FFFFFFF9 008B8000 0001BC1D
0x20003D30: 00000048 00000010 0000A000 00000009
0x20003D40: 0001E326 00000001 80E80000 51538000
0x20003D50: 000080E8 0001E9CF 00000000 00000009
0x20003D60: 61C78000 000080E8 00000048 00000504
0x20003D70: 0000A1FC 0002125C 00000000 000080E8
0x20003D80: 00000000 0012A236 00000000 0001E4D9
0x20003D90: 000080E8 00000009 00004998 000080E8
0x20003DA0: 622C8000 0012A29B 00000042 0001E479
0x20003DB0: 40011000 000185EF 00006E10 00000000
0x20003DC0: 00000000 00000004 0000000C 00000000
0x20003DD0: 62780000 00018579 2000311B 0001ACDF
0x20003DE0: 00000000 20003054 20002050 00000001
0x20003DF0: 20003DF8 0002085D 00000001 200030D4
0x20003E00: 00000200 0001F663 00000001 200030D4
0x20003E10: 00000001 2000311B 0001F631 00020A6D
0x20003E20: 00000001 00000000 0000000C 200030D4
0x20003E30: 2000311B 00000042 200030D4 00020AD7
0x20003E40: 20002050 200030D4 20002050 00020833
0x20003E50: 20002050 20003F1B 20002050 0001FF89
0x20003E60: 20002050 0001FFA3 00000005 20003ED8
0x20003E70: 20002050 0001FF8B 00000010 00020491
0x20003E80: 00000001 0012A54E 00000020 00022409
0x20003E90: 00000000 20002050 200030D4 0001FF8B
0x20003EA0: 00021263 00000005 0000000C 20003F74
0x20003EB0: 20003ED8 20002050 200030D4 00020187
0x20003EC0: 20003ED4 20003054 00000000 20003F75
0x20003ED0: 00000008 20003F64 00000084 FFFFFFFF
0x20003EE0: FFFFFFFF 00000008 00000001 00000008
0x20003EF0: 20302058 2000311B 0001F631 00020A6D
0x20003F00: 20002050 00000000 0000000C 200030D4
0x20003F10: 32002050 32303032 00323330 000258D7
0x20003F20: 20002050 200030D4 20002050 00020833
0x20003F30: 00000000 20002050 00000020 000001CE
0x20003F40: 20003F40 200030D4 00000004 0001ED83
0x20003F50: 200030D4 20003F60 000001D6 000001D7
0x20003F60: 000001D8 00016559 0000000C 00000000
0x20003F70: 6C383025 00000058 200030D4 FFFFFFFF
0x20003F80: 1FFF4000 00000028 00000028 000217F8
0x20003F90: 200020C7 000166C5 000166AD 00017ED9
0x20003FA0: FFFFFFFF 200020B8 2000306C 200030D4
0x20003FB0: 200020B4 000180AD 1FFF4000 200020B0
0x20003FC0: 200020B0 200020B0 1FFF4000 0001A63D
0x20003FD0: CAFEF00D CAFEF00D 200020B4 00000002
0x20003FE0: FFFFFFFF FFFFFFFF 1FFF4000 00000000
0x20003FF0: 00000000 00000000 00000000 00016113
OK

Historique des versions

Adafruit maintient un historique complet des modifications de son firmware.

Vous trouverez cette historique directement depuis le tutoriel d'Adafruit https://learn.adafruit.com/adafruit-bluefruit-le-shield/history (*Adafruit, anglais*).

Service GATT en détail

GATT Service Details

En Bluetooth Low Energy, les données sont organisées autour d'unités appelées 'Services GATT https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gatt#services-and-characteristics ' et 'Caractéristiques'. Pour exposer des données vers un autre périphérique, vous devez instancier au moins un service sur votre périphérique.

Les modules Adafruit Bluefruit LE Pro supporte quelques services 'standard' décris ci-après (d'autres pourraient être ajoutés dans le futur).

UART Service

The UART Service is the standard means of sending and receiving data between connected devices, and simulates a familiar two-line UART interface (one line to transmit data, another to receive it).

Vous trouverez plus d'information sur les commandes AT du service UART ici. Le service est décrit en détail sur la page dédicacée au Service UART Service https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/uart-service .

Service UART

Sommaire

Le service UART

Base UUID: 6E40**0001**-B5A3-F393-E0A9-E50E24DCCA9E

Ce service simule une connexion UART (série) basique avec les deux lignes, TXD et RXD.

Il est basé sur les spécifications propriétaire UART service de Nordic Semiconductors. Les données envoyées ou reçues par ce service peuvent être visualisées à l'aide des APPs nRFUART de Nordic Semiconductors pour Android et iOS.

Ce service est disponible sur tous les modules Bluefruit LE et est automatiquement démarré durant la séquence de mise sous tension.

Caractéristiques

Le Service UART de Nordic inclus les caractéristiques suivantes:

Nom	Requis	UUID	Type	R	W	N	I
TX	Oui	0x0002	U8[20]		X		
RX	Oui	0x0003	U8[20]	X		X	

R = Read (*lecture*); W = Write (*écriture*); N = Notify (*notification*); I = Indicate (*indicateur*)

Les noms des caractéristiques sont assignés en fonction du point de vue du périphérique central (*Central device*, autrement dit, le SmartPhone ou la Tablette)

TX (0x0002)

Cette caractéristiques est utilisé pour envoyer des données vers le noeud senseur et peut être connecté sur le périphérique Centrale (le téléphone mobile, tablette, etc.).

RX (0x0003)

Cette caractéristique est utilisée pour envoyer des données vers le périphérique cental. La notification peut être activée par le périphérique connecté de sorte d'une alerte est communiquée chaque fois que le canal TX est mis-à-jour.

Réinitialisation d'usine

Sommaire

Réinitialisation d'usine

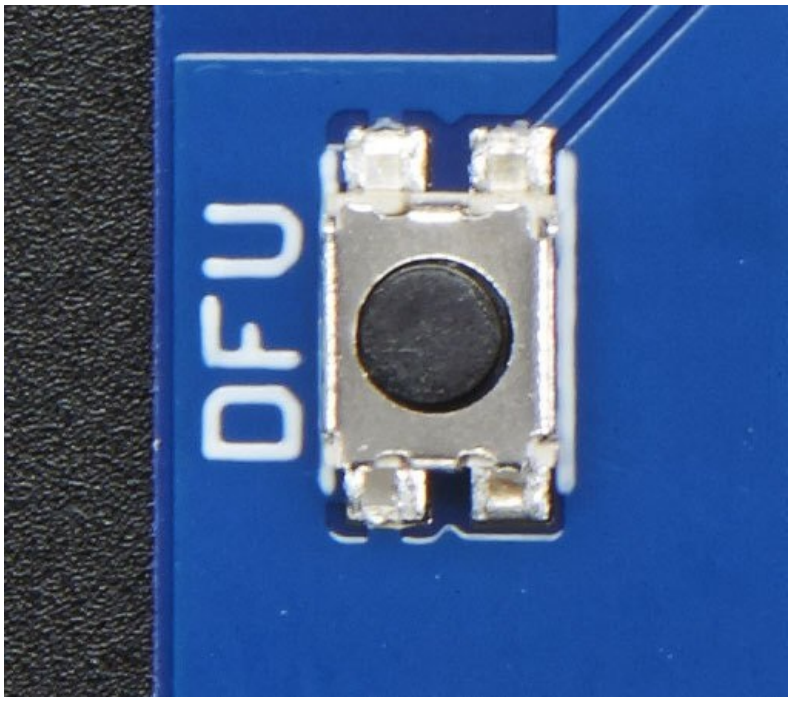
Il y a plusieurs méthodes utilisable pour effectuer une réinitialisation d'usine sur votre module Bluefruit LE. Cela permet de réinitialiser le module s'il est mal configuré ou s'il est nécessaire d'effacer des changements permanent (comme UriBeacon ou modification du payload d'avertissement, etc).

Ces commandes sont identique pour les version UART et SPI du Bluefruit LE

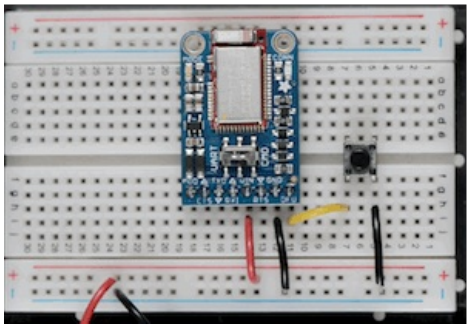
Réinitialisation d'usine via la broche DFU

Si vous maintenez la broche DFU au niveau vas (raccorder la broche sur la masse) prendant plus de 5 secondes alors les LEDs rouge et bleue à côté du module commencerons a clignoter et le périphérique effectuera une réinitialisation d'usine dès le relâchement de la broche DFU (déconnectée de GND).

Si vous disposez du bouton DFU (au lieu de la broche), maintenez le simplement enfoncé.



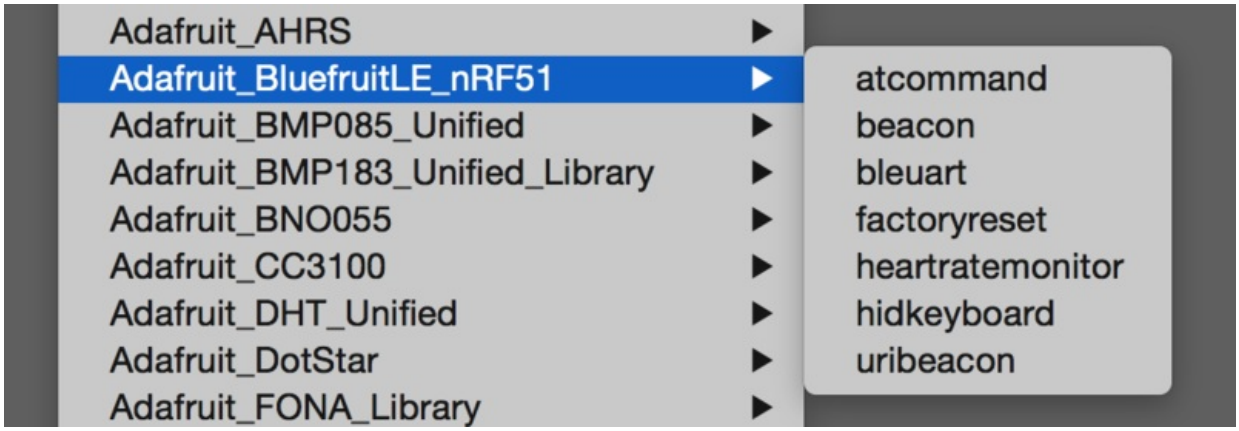
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

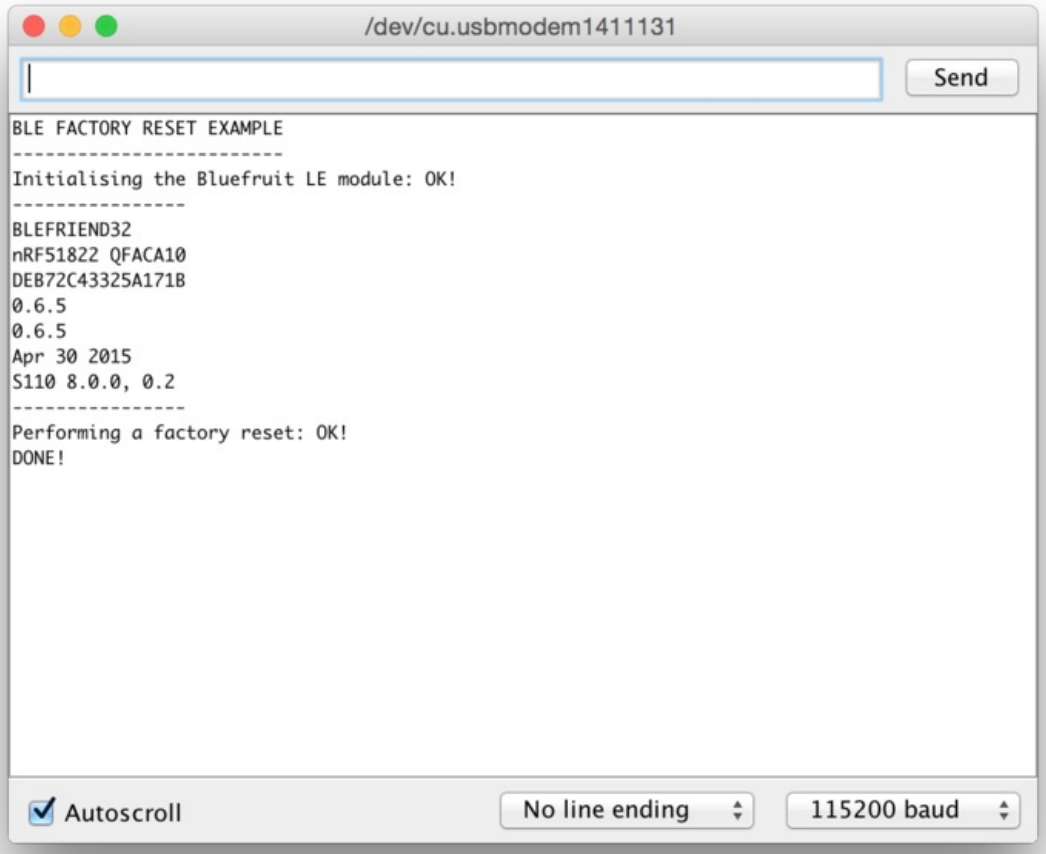
Croquis d'exemple FactoryReset

Il existe également un croquis/sketch d'exemple nommé FactoryReset (voir les exemples de la bibliothèque Adafruit Bluefruit LE disponible sous le point de menu '*Fichier > Exemples > Adafruit_BluefruitLE_nRF51* (Voyez la section logiciel dans ce tutoriel https://learn.adafruit.com/introducing-the-adafruit-bluefruit-spi-breakout/software pour les instructions concernant l'installation de la bibliothèque):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Téléversez le croquis/sketch, ouvrez un moniteur série et le croquis devrait faire une réinitialisation d'usine pour vous:



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

AT+FACTORYRESET

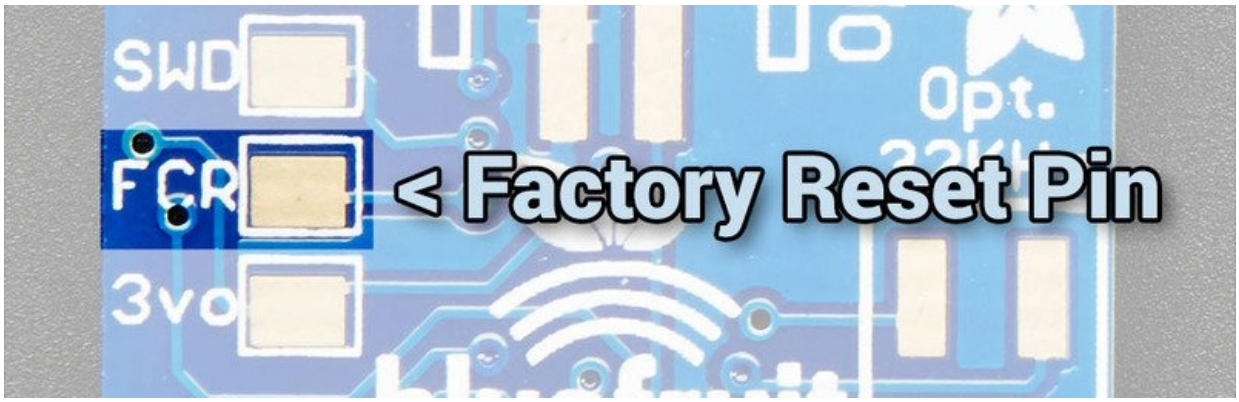
Vous pouvez également effectuer une réinitialisation d'usine en utilisant la commande AT+FACTORYRESET sur le module Bluefruit LE depuis votre terminal de commande favoris ou en utilisant le croquis/sketch d'exemple ATCommand.

```
AT+FACTORYRESET
OK
```

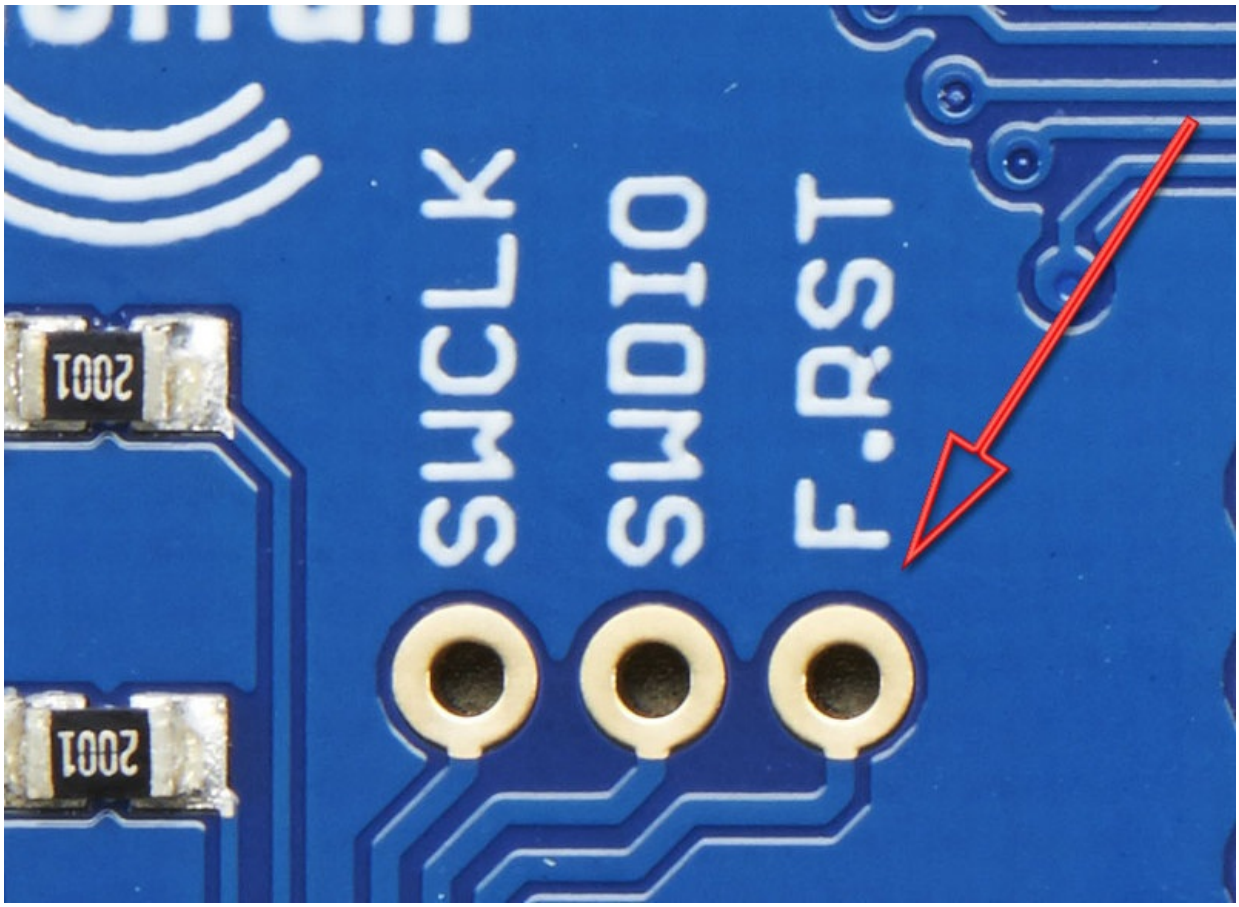
Cette provoquera également une réinitialisation d'usine.

Réinitialisation d'usine via la pastille de test FCTR

Sous la carte Bluefruit LE Friend ou le shield il y a une pastille de test ou broche qui expose la broche "Factory Reset" du module. Cette pastille/broche porte la mention **FCR** ou **F.RST** sur la sérigraphie. Placer cette broche au niveau bas avant la mise-sous-tension provoquera la réinitialisation d'usine au démarrage.



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Mise-à-jour Firmware (DFU)

Mise à jour DFU

DFU est l'acronyme de Device Firmware Update (mise-à-jour du Firmware du périphérique).

Adafruit travaille constamment sur le firmware du Bluefruit LE pour y ajouter de nouvelles fonctionnalités et maintenir le firmware à jour.

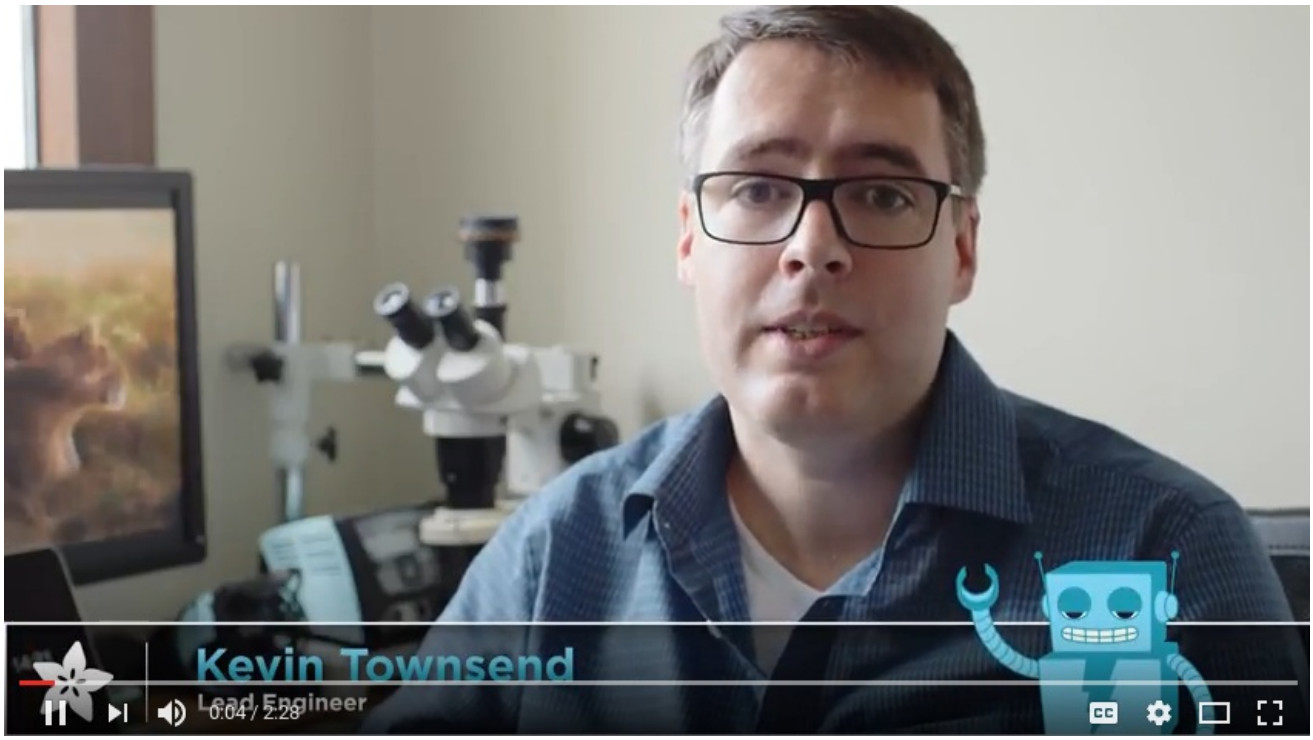
Pour faciliter la mise-à-jour de ces modification, Adafruit à inclus un procédé de mise-à-jour "over the air" (via la connexion Bluetooth) pour tous les modules BlueFruit LE à base de nRF51.

Cela signifie que vous pourrez faire la mise-à-jour depuis un smartphone sous Android ou iOS.

Adafruit Bluefruit LE Connect

La mise-à-jour de votre périphérique Bluefruit LE avec le dernier Firmware est facilité en installant de l'App Bluefruit LE Connect d'Adafruit https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect (sur Android) depuis le Google Play Store ou l'App Bluefruit LE Connect pour iOS https://itunes.apple.com/app/adafruit-bluefruit-le-connect/id830125974?mt=8 frpuis l'Apple App Store.

A chaque nouvelle mise-à-jour du Firmware, l'application proposera de télécharger le dernier binaire s'occupe de tous les détails concernant le transfert de celui-ci vers le périphérique Bluefruit device (comme présenté sur la vidéo suivante):



Vous pouvez également consulter la vidéo directement sur YouTube https://youtu.be/tLmqMWC7pWY

SDEP (SPI Data Transport)

En cours de traduction/élaboration.

Sommaire

SDEP (SPI Data Transport)

Le Bluefruit LE SPI Shield (et Firend) utilise le même ensemble de commande que les modules BlueFruit UART (ATI, AT+HELP, etc.).

Il a donc fallut mettre une solution au point pour que les commandes ATs puissent aussi être envoyées via le bus SPI.

Les commandes AT (du texte) sont donc encodées en messages binaires en utilisant un protocol nommé **SDEP** (Simple Data Exhange Protocol) par Adafruit.

Présentation de SDEP

SDEP à été conçu comme un protocole bus neutre pour gérer la transmission de commandes et réponses binaires -- incluant les messages d'erreur --. Le protocole est conçu pour être facile à étendre. Un protocole neutre pour bus (*Bus neutral*) signifie que nous pouvons utiliser SDEP quelque soit le mécanisme de transport (USB HID, SPI, I2C, Donnée via une connexion sans fil, etc.).

Tous les messages SDEP ont **une entête de 4 octets** et dans le cas des modules Bluefruit LE **jusqu'à 16 octets de payload**. Les grands messages sont coupé en plusieurs messages de 4+16 octets (appelés *message chunks*), ce qui permet de reconstruire le message de l'autre côté du bus.

La limite de 20 octets (4 octets pour l'entête + 16 octets de payload) a été choisi pour tenir compte de la taille maximal d'un paquet dans Bluetooth Low Energy 4.0 (20 octets par paquet).

Configurer SPI

Bien que SDEP soit prévu pour être un protocole neutre vis-à-vis du bus, dans le cas du Bluefruit LE SPI Friend ou du shield BlueFruit LE, le transport SPI est utilisé avec les contraintes et suppositions suivantes, principalement guidés par les limitations matérielles du nRF51822:

Conditions requises du SPI matérielles

- Le signal d'horloge SPI doit être <= 4MHz
- Un délais de 100µs doit être ajouté entre le moment où la ligne CS est activée et la transmission des premières données sur le bus SPI
- La ligne CS doit rester activée (*asserted*) pour tout le paquet de donnée (et non activé pour chaque octet transmit)
- La ligne CS peut cependant être activée ou désactivée entre l'envoi de différents paquets SDEP (chaque paquet ayant jusqu'à 20 octets max).
- Les commandes SPI doivent être constituées pour une transmission MSB (most significant bit https://en.wikipedia.org/wiki/Most_significant_bit) en premier (le bit le plus significatif en premier).

Broche IRQ

La ligne IRQ est activée par le par le Bluefruit LE Shield (ou Friend) SPI dès que (et aussi longtemps que) un paquet SDEP complet est disponible dans la mémoire tampon (*buffer*) du nRF51822. Vous devriez lire le paquet lorsque l'IRQ (interruption) est activée, garder la ligne CS activée durant la transaction entière (comme détaillé ci-dessous).

La ligne IRQ reste activée aussi longtemps qu'un ou plusieurs paquets sont disponibles. Par conséquent, la ligne pourrait rester active après la lecture d'un paquet, cela signifie qu'il reste encore des paquets à lire dans la mémoire tampon FIFO du module SPI esclave.

FIFO est l'acronyme de *First In First Out* qui signifie "Premier Rentré Premier Sorti".

Paquets SDEP et identification des erreurs SPI

Une fois la ligne CS activée et le délai de 100µS écoulé, un simple octet devrait être lu sur le bus et ce dernier indiquera le type de payload (paquet) disponible sur le nRF51822 (Voir "*Message Type Indicator*" ci-dessous pour plus d'information sur les types de message SDEP). Gardez la ligne CS active après la lecture de ce premier octet au cas ou vous auriez besoin de poursuivre la lecture de données complémentaire (le restant du payload s'il fait plus d'un octet).

Si nous nous trouvons face à un "*Message Type Indicator*" SDEP standard (0x10, 0x20, 0x40 or 0x80) alors nous pouvons poursuivrez normalement la lecture

Il existe deux autre "*Message Type Indicator*" dont il faut tenir compte. Ces derniers indique un problème du côté esclave de la communication SPI (le nRF51822):

- **0xFE** : Le périphérique esclave n'est pas prêt (attendre un peu et réessayer)
- **0xFF** : Indicateur de dépassement de lecture (*read overflow indicator*) sur le périphérique esclave (il y a eu une tentative de lire plus de donnée que de donnée vraiment disponible sur l'esclave)

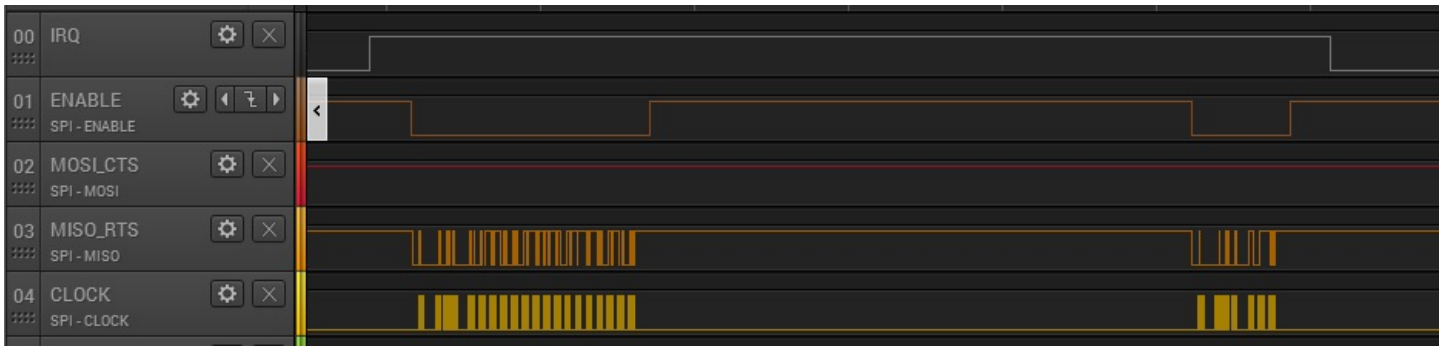
Cela signifie qu'il y a 6 réponses possible dansle premier octet lu (donc la valeur du "message type indicator" renvoyée par le périphérique esclave).

Ce premier octet après l'envoi d'une commande SDEP peut avoir les valeurs suivantes:

- 0x10, 0x20, 0x40, 0x80 qui indiquent un type de message valide
- 0xFE, 0xFF qui indiquent une condition d'erreur

Exemple de Transaction

L'image suivante présente un exemple de réponse SDEP qui est réparti sur deux paquets (étant donné que la taille de la réponse est > 20 octets). Notez comme la ligne IRQ reste active entre les deux paquets puisque plus d'un paquet était disponible dans la mémoire tampon FIFO sur le Bluefruit LE (côté esclave du bus SPI):



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

SDEP - protocole d'échange de donnée simplifié

SDEP est l'acronyme de "Simple Data Exchange Protocol" qui signifie "Protocole simple d'échange de donnée".

SDEP peut être utilisé pour envoyer et recevoir des messages binaires entre deux périphériques connectés ensembles par l'intermédiaire d'un bus série (USB HID, USB Bulk, SPI, I2C, Wireless, etc.), échangeant des données en utilisant l'un des 4 types de messages disponibles (message de Commande, Réponse, Alerte et Erreur).

Le protocole est conçu pour pour être flexible et extensible, avec le seule exigence d'avoir **des messages individuel de maximum 20 octets**, et que le premier octet de chaque message est un octet/byte (U8) qui indique le type de message. Le type de message définit également le format du restant du paquet (*payload*).

Endianness

Endianness https://fr.wikipedia.org/wiki/Endianness#Big_endian (*wikipedia*) est l'ordre dans lequel sont organiser les octets en mémoire et dans les communications lorsqu'ils transportent une informations d'une taille supérieure à 1 octet. C'est le cas des entiers qui se codent sur

16 ou 32 bits.

Dans le protocole SDEP, toutes les valeurs supérieures à 8-bits sont encodées avec le format *little endian*. Toute déviance à cette règle doit être clairement documenté.

Indicateur du type de message

Le tout premier octet de chaque message est un identifiant 8 bits appelé **Message Type Indicator** en anglais. Cette valeur indique le type message envoyé et permet de déterminer le format du restant du message.

Type de message	ID (U8)
Commande	0x10
Réponse	0x20
Alerte	0x40
Erreur	0x80

Transactions de données SDEP

L'un ou l'autre des périphériques connectés peu initier un transaction SDEP. Néanmoins, certains protocoles de transport impose des restrictions concernant lequel "des périphériques" peut initier un transfert.

Par exemple, le périphérique maître initie TOUJOURS les transactions avec le Bluetooth Low Energy ou USB, ce qui signifie que le périphérique esclave peut uniquement répondre au commandes entrantes.

Tous les périphérique recevenant un **Message de Commande** doit répondre avec un **Message Réponse**, un Message d'Erreur ou un Message d'Alerte.

> Cette partie du tutoriel est très technique et ne concernera, a priori, que quelques utilisateurs avancés. Vous pouvez poursuivre votre lecture directement sur tutoriel d'Adafruit

Si cette partie du tutoriel vous intéresse, vous pouvez **poursuivre votre lecture** sur https://learn.adafruit.com/adafruit-bluefruit-le-shield/sdep-spi-data-transport#sdep-data-transactions

Ressources logicielles

Sommaire

Introduction

Cette page contient un certain nombre d'outils open source pour la plupart des plateformes supportant Bluetooth Low Energy.

Ces outils permettront de faciliter la communication entre le module Bluefruit LE et autres périphériques Centrales (SmartPhone, Ordinateur, etc).

Bluefruit LE Client Apps and Libraries

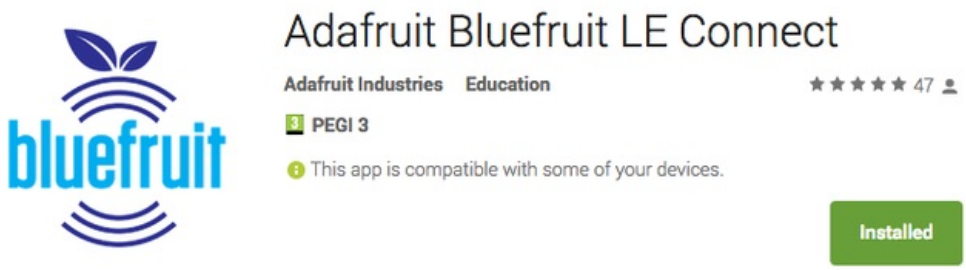
Adafruit à préparé une série d'application desktop et mobile ainsi que des bibliothèques pour faciliter, au maximum, les communication entre votre module Bluefruit LE et votre SmartPhone et Laptop. Les codes sources ont étés publiés où lorsque cela à été possible:

Bluefruit LE Connect (Android/Java)

Le support Bluetooth Low Energy a été ajouté depuis Android 4.3 (mais vraiment stable depuis Android 4.4).

Adafruit met donc à disposition Bluefruit LE Connect sur Play Store https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect .

Les codes source complet https://github.com/adafruit/Bluefruit_LE_Connect_Android de Bluefruit LE Connect pour Android est également disponible sur Github. Cela devrait vous aider à démarrer vos propres apps Android. Vous aurez besoin d'une version récente de Android Studio https://developer.android.com/sdk/index.html pour pouvoir ouvrir ce projet.



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Bluefruit LE Connect (iOS/Swift)

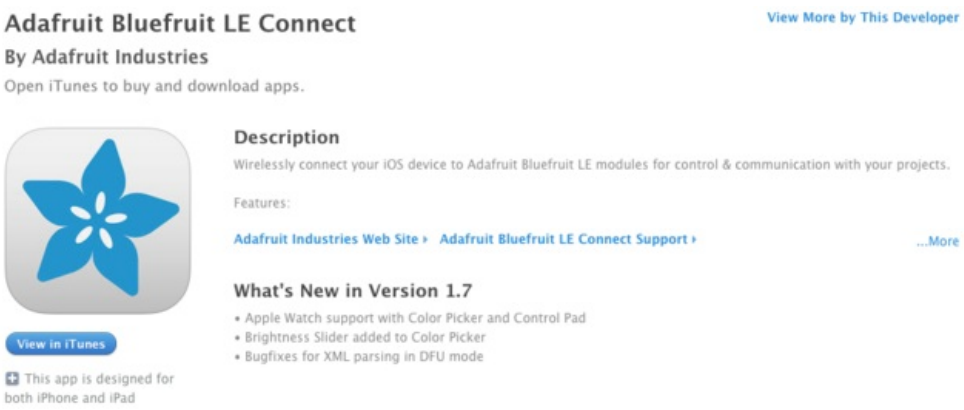
Apple a adopté Bluetooth Low Energy très rapidement et Adafruit propose également une version iOS de Bluefruit LE Connect app https://itunes.apple.com/app/adafruit-bluefruit-le-connect/id830125974?mt=8] (disponible sur l'Apple store).

Le code source swift complet de Bluefruit LE Connect pour iOS est également disponible sur Github. Vous aurez besoin de XCode et un accès au programme développeur d'Apple ("*Apple developper program*") pour utiliser ce project:

- Code source de la Version 1.x: https://github.com/adafruit/Bluefruit_LE_Connect
- Code source de la Version 2.x: https://github.com/adafruit/Bluefruit_LE_Connect_v2

> Version 2.x de l'application à été complètement ré-écrite pour supporter iOS, interface

graphique OS X GUI et ligne de commande OS X à partir d'un seul code.



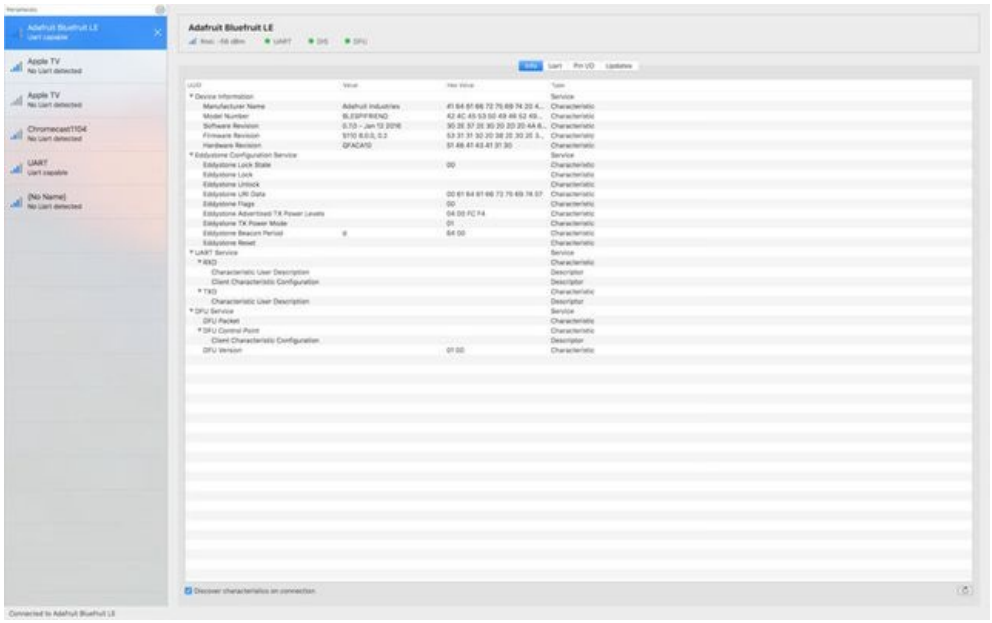
Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Bluefruit LE Connect for OS X (Swift)

Les application bureau OS X sont basées sur le même code que la version 2.x de l'App pour iOS et offre un accès à BLE UART, manipulation des I/O et mise-à-jour OTA du firmware.

C'est un choix judicieux pour faire du logging des données de senseurs afin de les exporter en fichier CSV, JSON ou XML. Vous pourrez ainsi exploiter ces informations dans une autre application. Cette approche vous permet d'utiliser la matériel BLE disponible sur votre ordinateur, il n'est donc pas nécessaire d'ajouter un clé BLE si vous disposez d'un Mac récent.

Le code source complet est également disponible sur Github https://github.com/adafruit/Bluefruit_LE_Connect_v2 .



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Ligne de commande Bluefruit LE pour mise-à-jour via OS X (Swift)

Cet outil en ligne de commande expérimental (non supporté, disponible à titre de "test de concept") peut être utilisé pour faire une mise-à-jour du Firmware des périphériques Bluefruit.

Cet utilitaire effectue une mise-à-jour automatique du Firmware (comme celle réalisée par la version graphique du logiciel). Elle vérifie la version du firmware installé sur le périphérique Bluefruit (bia le service "Device Information") et le compare avec les versions du firmware disponibles en ligne. Si nécessaure, les fichiers son téléchargés en tâche de fond.

Installez simplement l'outil précompilé via le fichier DMG https://github.com/adafruit/Bluefruit_LE_Connect_v2/releases/tag/OSXcommandline_0.3 et placez le quelque par dans les répertoires systèmes -ou- exécutez le locallement via './bluefruit' (pour voir le message d'aide en anglais):

```
$ ./bluefruit
bluefruit v0.3
Usage:
 bluefruit <command> [options...]

Commands:
 Scan peripherals:   scan
```

```
Automatic update:   update [--enable-beta] [--uuid <uuid>]
Custom firmware:    dfu --hex <filename> [--init <filename>] [--uuid <uuid>]
Show this screen:   --help
Show version:       --version

Options:
--uuid <uuid>    If present the peripheral with that uuid is used. If not present a list of peripherals is displayed
--enable-beta    If not present only stable versions are used

Short syntax:
-u = --uuid, -b = --enable-beta, -h = --hex, -i = --init, -v = --version, -? = --help
```

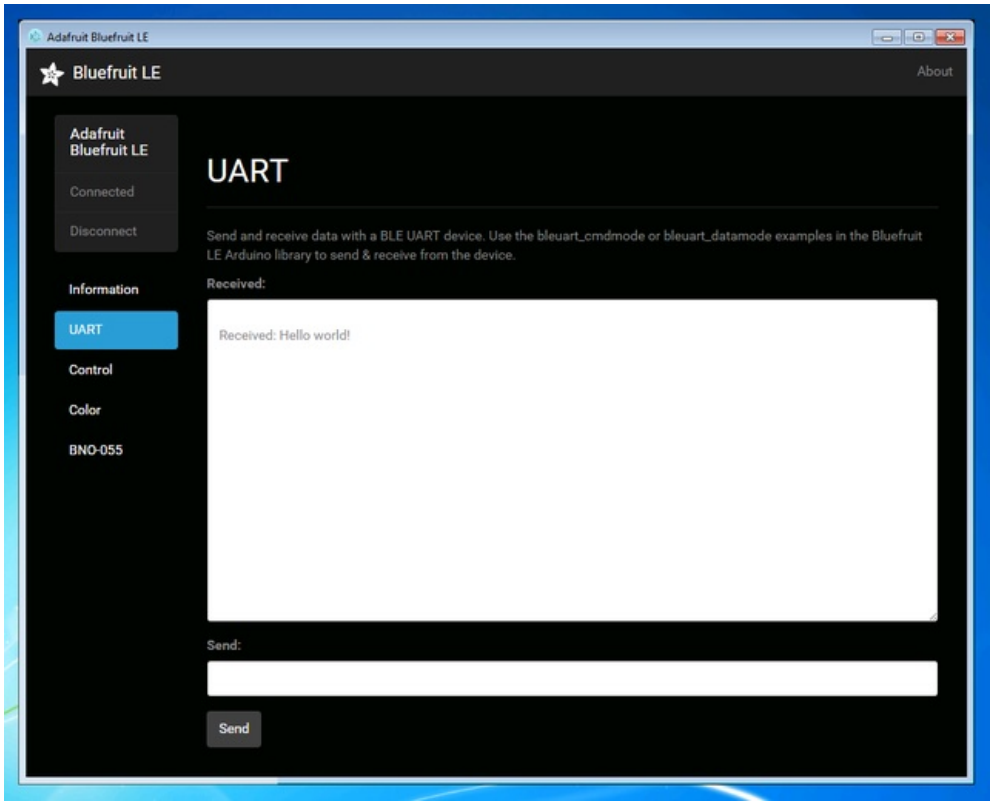
ABLE (Cross Platform/Node+Electron)

ABLE https://github.com/adafruit/adafruit-bluefruit-le-desktop/releases (Adafruit Bluefruit LE Desktop) est une application de bureau cross-platform basé sur la bibliothèque noble https://github.com/sandeepmistry/noble de Sandeep Misty et le projet Electron https://github.com/atom/electron (utilisé par Atom).

Il fonctionne sur OS X, Windows 7+ et une sélectionne de systèmes Linux (Ubuntu testé locallement). Le support Windows 7 est particulièrement intéressant étant qu'il n'a pas de support natif de Bluetooth Low Energy mais la bibliothèque noble permet de dialoguer directement avec des clés Bluetooth 4.0 USB http://shop.mchobby.be/product.php?id_product=566 (supportée) pour émuler BLE sur le système (toujours en BETA, prenez plus de précautions lorsque vous travaillez avec lui).

Cette application permet de collecter des données de senseur et permet de réaliser de nombreuses fonctionnalités similaires à celles de l'apps Bluefruit LE Connect mais dans un environnement de bureau.

L'application est toujours en BETA mais le code source complet https://github.com/adafruit/adafruit-bluefruit-le-desktop est disponible sur GitHub ainsi que des binaires pré-compilés https://github.com/adafruit/adafruit-bluefruit-le-desktop/releases (pour faciliter leur utilisation).



Crédit: AdaFruit Industries www.adafruit.com http://www.adafruit.com

Toujours en guise de "test de concept", Adafruit à produit du code pour utiliser Python avec les APIs Bluetooth natives sur sur OS X et la dernière versions de Bluez pour Linux (disponibles sur certaines distributions Linux).

Il y a un croquis d'exemple qui montre comment recevoir des données BLE UART ainsi que des informations complémentaires en provenance du service Bluetooth DIS ("*Device Information Service*").

Il ne s'agit pas d'un projet supporté activement mais plutôt d'un projet d'expérimentation. Si vous avez un Macbook récent ou un Raspberry Pi et quelques connaissances Python vous pourriez trouver les sources Adafruit_Python_BluefruitLE https://github.com/adafruit/Adafruit_Python_BluefruitLE (github d'Adafruit intéressante).

Outil de débogage

Si vous êtes un grand aventurier et que, dans durant vos tribulations, votre module Bluefruit LE à totalement perdu le nord alors les outils suivants pourraient se montrer utiles pour revenir en terrain connue.

Ces outils de débogages sont uniquement fournit pour les utilisateurs avancés désirant restaurer l'état de leur périphérique. Il n'est pas recommander de les utiliser sauf si vous savez ce que vous faites car une utilisation incorrecte peut bloquer votre board.

L'utilisez de ces outils se fait à vos propres risques.

AdaLink (Python)

Outil en ligne de commande basé sur le wrapper Python. Il permet de programmer le MCU ARM en utilisant soit un Segger J-Link https://www.adafruit.com/search?q=J-Link ou un STLink/V2 https://www.adafruit.com/product/2548 . Vous pouvez l'utiliser pour reflasher votre module Bluefruit LE en utilisant de dernier firmware disponible sur le dépôt des firmware Bluefruit LE https://github.com/adafruit/Adafruit_BluefruitLE_Firmware .

Les détails concernant l'utilisation de cet outil sont disponible dans le fichier readme.md du dépôt Github Adafruit_Adalink https://github.com/adafruit/Adafruit_Adalink .

Le reflashage complet du module Bluefruit LE avec AdaLink nécessitera 4 fichiers et ressemblera à quelque-chose comme ceci (en utilisant un JLink):

```
adalink nrf51822 --programmer jlink --wipe
  --program-hex "Adafruit_BluefruitLE_Firmware/softdevice/s110_nrf51_8.0.0_softdevice.hex"
  --program-hex "Adafruit_BluefruitLE_Firmware/bootloader/bootloader_0002.hex"
  --program-hex "Adafruit_BluefruitLE_Firmware/0.6.7/blefriend32/blefriend32_s110_xxac_0_6_7_150917_blefriend32.hex"
  --program-hex "Adafruit_BluefruitLE_Firmware/0.6.7/blefriend32/blefriend32_s110_xxac_0_6_7_150917_blefriend32_signature.hex"
```

Vous pouvez également utiliser l'outil AdaLink pour obtenir des informations de diagnostic a propos de votre module. A l'aide de l'option `--info`, vous pourrez, par exemple, obtenir des information sur le *SoftDevice* programmé dans le module ou la révision du circuit intégré (16KB SRAM ou 32KB SRAM).

```
$ adalink nrf51822 -p jlink --info
Hardware ID : QFACA10 (32KB)
Segger ID   : nRF51822_xxAC
SD Version  : S110 8.0.0
Device Addr : **:**:**:**:**:**
Device ID   : ****************
```

Adafruit nRF51822 Flasher (Python)

Adafruit nRF51822 Flasher est un outil python utilisé en interne par Adafruit. Il est utilisé en production pour flasher les cartes avec le Firmware Adafruit au moment où elles passent la procédure de test (avant de quitter la ligne d'assemblage).

Cet outil est également utilisé pour tester les différentes releases du firmware durant les phases de débogages.

Il s'appuie sur AdaLink ou OpenOCD (voir ci-dessous) mais vous pouvez utiliser cet outil en ligne de commande pour flashet votre nRF51822 avec un *SoftDevice* spécifique, combinaison alternatives de Bootloader et firmware Bluefruit.

Actuellement, il supporte soit un J-Link de Segger ou STLink/V2 par l'intermédiaire d'AdaLink, ou GPIO d'un Raspberry Pi https://github.com/adafruit/Adafruit_nRF51822_Flasher#rpi-gpio-requirements si vous n'avez pas accès à un débugger ARM SWD traditionnel. (une version d'OpenOCD précompilé pour RPi est inclus dans le dépot - recompiler cet outils depuis les sources nécessite un temps considérable.)

Adafruit n'offre pas de support actif pour cet outil (puis qu'il s'agit d'un outil à usage interne). Il est rendu public pour aider les utilisateurs aventureux qui désirent se débrouiller par eux-même.

```
$ python flash.py --jtag=jlink --board=blefriend32 --softdevice=8.0.0 --bootloader=2 --firmware=0.6.7
jtag       : jlink
softdevice  : 8.0.0
bootloader  : 2
board      : blefriend32
firmware    : 0.6.7
Writing Softdevice + DFU bootloader + Application to flash memory
adalink -v nrf51822 --programmer jlink --wipe --program-hex "Adafruit_BluefruitLE_Firmware/softdevice/s110_nrf51_8.0.0_softdevice.hex" --program-hex "Adafruit_BluefruitLE_Firmware/bootloade
...
```

Foire Aux Questions

Sommaire

Mon Bluefruit LE peut-il dialoguer avec un périphérique 'Classic Bluetooth' ?

Non. Bluetooth Low Energy et Bluetooth 'Classic' sont deux parties d'une même noyau de spécification Bluetooth (*Bluetooth Core Specification*) -- définie et maintenue par le Bluetooth SIG -- mais sont des protocoles complètements différents fonctionnant avec des contraintes et des exigences différentes. Les deux protocoles ne peuvent pas dialoguer directement.

Mon module Bluefruit LE peut-il se connecter sur d'autres périphériques Bluefruit LE ?

Non, le firmware Bluefruit LE d'Adafruit est uniquement un périphérique et ne fonctionne pas en mode Central (comme un SmartPhone ou support BLE d'un laptop).

Si vous avez besoin du mode Central alors vous devriez vous pencher sur les nouveaux produits basés sur nRF52832 tel que l'Adafruit Feather nRF52 Bluefruit LE https://www.adafruit.com/product/3406 qui contient un firmware capable de fonctionner soit en mode Central ou soit en mode Périphérique.

Les produits à base de **nRF51**8322 (comme celui utilisé dans ce guide) ne sont **PAS** capable de fonctionner en mode Centrale parce qu'il n'est pas supporté par le firmware qu'il utilise. Il n'est pas possible de faire une mise-à-jour en toute sécurité sans utiliser un matériel spécial.

Pourquoi mes changements ne persiste t'il pas lorsque j'utilise un croquis d'exemple ?

La plupart des croquis effectuent une réinitialisation d'usine (FACTORY RESET) pour être certain que le module Bluefruit LE soit dans un état connu avant d'exécuter le croquis de démo.

C'est utile pour assurer le bon fonctionnement d'un croquis mais a comme effet de board d'effacer toutes les données personnalisées de la NVM pour y remettre le paramétrage d'usine à chaque fois que le croquis est redémarré (ou Arduino réinisitalisé, ce qui redémarre le croquis).

Pour désactiver la réinitialisation d'usine, ouvrez le croquis de démo et trouvez la ligne qui initialise le flag/drapeau **FACTORYRESET_ENABLE** et fixer la valeur à '0' (cela empêchera la réinitialisation d'usine au démarrage du croquis).

Si vous ne voyez pas le flag/drapeau 'FACTORYRESET_ENABLE' dans le fichier .ino c'est que vous avez probablement une ancienne version de la bibliothèque. Vous pourriez avoir besoin de faire une mise-à-jour de la bibliothèque via le gestionnaire de bibliothèque Arduino.

Ai-je besoin de CTS et RTS avec mon module UART Bluefruit LE ?

Utiliser CTS et RTS n'est strictement nécessaire lorsque l'on utilise une connexion série matérielle (*HW serial*) mais elle devrait être utilisé lors d'une connexion série logicielle (*SW serial*) pour réguler le flux de donnée et éviter de perdre des octets. Il est également recommandé d'utiliser CTS et RTS si vous avez beaucoup de données à transférer.

La raison derrière le besoin des signaux CTS et RTS c'est que le block UART du nRF51822 n'est pas très robuste. Les toutes premières versions du composant avaient un FIFO extrêmement petit, ce qui signifie que le périphérique UART était rapidement submergé.

Utiliser CTS et RTS améliore significativement la fiabilité de la connexion UART puisque ces deux broches indiquent au périphérique distant qu'il faut attendre que la mémoire tampon soit traitée.

Pour activer le support CTS et RTS, il faut ouvrir le fichier BluefruitConfig.h et assigner les broches appropriées aux macros dédiées à ces fonctions (elles sont fixées à -1 si elles ne sont pas utilisées).

Activer ces deux broches devrait résoudre les problèmes de fiabilité que vous pourriez rencontrer avec les grandes commandes, ou les transmissions de nombreuses commandes d'affilé.

Comment puis-je faire une mise-à-jour pour utiliser la dernière version du firmware?

La façon la plus facile de maintenir vos modules Bluefruit LE à jour est d'utiliser l'app Bluefruit LE Connect pour Android https://play.google.com/store/apps/details?id=com.adafruit.bluefruit.le.connect ou Bluefruit LE Connect pour iOS https://itunes.apple.com/app/adafruit-bluefruit-le-connect/id830125974?mt=8 .

Ces deux apps inclus une fonctionnalité de mise-à-jour du firmware qui vous permet de facilement télécharger le dernier Firmware et de lancer une mise-à-jour via la connexion Bluetooth. En cas de problème, vous pouvez également utiliser l'app pour revenir à une version précédente du firmware Bluefruit LE.

Quelle version du firmware supporte 'xxx'?

Adafruit publie régulièrement des images Firmware pour Bluefruit LE https://github.com/adafruit/Adafruit_BluefruitLE_Firmware corrigeant des bugs et ajoutant de nouvelles fonctionnalités. Chaque commande AT reprise dans le guide affiche la version du Firmware nécessaire pour utiliser la commande.

Pour une vue plus globale des changements du Firmware (d'une version à l'autre), vous pouvez consulter la page *firmware history* https://learn.adafruit.com/introducing-adafruit-ble-bluetooth-low-energy-friend/history maintenu par Adafruit.

Est-ce que mon périphérique Bluefruit LE supporte ANCS?

ANCS https://developer.apple.com/library/content/documentation/CoreBluetooth/Reference/AppleNotificationCenterServiceSpecification/Introduction/Introduction.html (*Apple Notification Center Service*) fait partie des projets d'Adafruit (probablement durant la famille des releases 0.7.x). ANCS n'est actuellement pas supporté par Adafruit étant donné qu'il y a quelques effets de bord particuliers lorsqu'il est implémenté comme un service.

Mon Bluefruit LE est planté en mode DFU ... que puis-je faire ?

Si votre périphérique est planté en mode DFU (pour une raison quelconque) avec un firmware est corrompu alors il vous reste plusieurs options.

Premièrement, essayer de faire une réinitialisation d'usie (*factory reset*) en maintenant le bouton DFU enfoncé pendant environ 10 secondes (jusqu'à ce que la LED **CONN** commence a flasher) puis relâchez le bouton DFU pour effectuer la réinitialisation d'usine.

Si cela ne fonctionne pas alors vous pourriez avoir besoin de reflasher votre firmware en mode DFU. Cela peut être réalisé d'une des façons suivantes:

Bluefruit LE Connect (Android)

- Placez le module en mode DFU (LED clignotant constamment)
- Ouvrez Bluefruit LE Connect

- Connectez vous sur le périphérique en mode DFU (*DfuTarg*)
- Une fois connecté, vous verrez un écran avec des informations de base. Cliquer sur '...' dans le coin en haut à droite et sélectionnez **Firmware Updates**
- Cliquez sur le bouton **Use Custom Firmware** (*Utiliser un Firmware personnalisé*)
- Selectionner les fichiers .hex et .init approrpriés (copiés depuis le dépôt des firmware Bluefruit LE https://github.com/adafruit/Adafruit_BluefruitLE_Firmware) ... pour le firmware BLEFRIEND32 version 0.6.7, cela devrait être:
  - Fichier Hex: blefriend32_s110_xxac_0_6_7_150917_blefriend32.hex
  - Fichier Init: blefriend32_s110_xxac_0_6_7_150917_blefriend32_init.dat
- Cliquer sur **Start Update**

L'app iOS ne supporte pas encore la mise-à-jour de firmware personnalié depuis le mode DFU mais Adafruit fait le nécessaire pour rendre la fonctionnalité disponible.

La boîte à outil Nordic nRF

Vous pouvez également utiliser l'application "Nordic's nRF Toolbox" pour faire une mise-à-jour du firmware en utilisant le bootloader OTA (*Over The Air*).

Sur **Android**:

- Ouvez nRF Toolbox (utiliser la dernière version de l'outil)
- Cliquer sur l'icone **DFU**
- Cliquer sur le bouton **Select File** pour sélectionner un fichier
- Selectionner **Application** depuis dans la liste des option PUIS cliquez sur **OK**
- Localiser le fichier .hex approprié (ex: 'blefriend32_s110_xxac_0_6_7_150917_blefriend32.hex')
- Lorsque l'App vous demande s'il faut un **Init packet** (*paquet d'initialisation*), répondez Oui en pressant sur **Yes**. Sélectionner le fichier *_init.dat approprié (par exemple: 'blefriend32_s110_xxac_0_6_7_150917_blefriend32_init.dat').
- Cliquer sur le bouton **Select Device** pour sélectionner un périphérique (voir en bas du l'écran principale) et sélectionner le périphérique en mode DFU (**DfuTarg**) en cliquant dessus
- Cliquez sur le bouton **Upload** (*téléverser*) qui devrait maintenant être disponible sur l'écran principal.
- Cela devrait débuter le processus de mise-à-jour DFU, ce qui devrait faire une mise-à-jour du Firmware et restaurer l'état du module Bluefruit LE

Sur **iOS**:

- Créer un fichier .zip contenant ke fichier .hex et le fichier init.dat que vous allez utiliser pour faire la mise-à-jour du firmware. Par exemple:
  - Renommer 'blefriend32_s110_xxac_0_6_7_150917_blefriend32.hex' en **application.hex**
  - Renommer 'blefriend32_s110_xxac_0_6_7_150917_blefriend32_init.dat' en **application.dat**
- Téléverser le **fichier .zip** contenant les fichiers `application.hex` et `application.dat` sur votre iPhone en utilisant, comme décrit ci-dessous:
- Ouvrer l'app nRF Toolbox (utilisez la dernière version)
- Cliquer sur l'icone **DFU**
- Cliquer sur le libellé texte **Select File** (*sélectionner fichier*)
- Basculer vers **User Files** (*fichier utilisateur*) pour voir le fichier .zip que vous avez téléchargé ci-dessus
- Sélectionner le fichier .zip (ex: blefriend32_065.zip)
- Sur l'écran principal, sélectionner **Select File Type** (*sélectionner le type de fichier*)
- Sélectionner **application**
- Dans l'écran principal, cliquer sur **SELECT DEVICE** (*sélectionner périphérique*)
- Sélectionner le périphérique en mode DFU (**DfuTarg**)
- Cliquer sur le bouton **Upload** (*téléverser*) qui devrait maintenant être disponible.
- Cela débutera le processus de mise-à-jour DFU et votre module Bluefruit LE sera réinitialisé une fois l'opération terminée
- Si vous voyez une schéma normal de 2 ou 3 clignotement alors la mise-à-jour à fonctionné!

Avec **Adafruit_nRF51822_Flasher**:

En dernier ressort, si vous avez accès à un Raspberry-Pi, un Segger J-Link ou un STLink/V2, vous pouvez manuellement reflasher le périphérique entier, comme décrit ci-dessus dans la foire aux questions, avec plus de détails sur la page ressources logicielles.

Comment est ce que je reflash mon module Bluefruit LE via SWD?

Reflasher le module Bluefruit LE via SWD (ex: passer à la version sniffer du firmware et inversement) est une opération **réalisée à vos propre risque et peu bloquer votre périphérique, et Adafruit ne peut pas offrir de support pour une telle opération!**. Vous êtes seule pour réaliser cette opération et il y a malheureusement 1,000,000 de choses qui peuvent mal tourner. C'est pour cela qu'Adafruit propose deux cartes différentes:

- Le Bluefruit LE Friend en version sniffer

- Et le Bluefruit LE Friend en version normale (avec le firmware *non-sniffer*) qui fournit un bootloader avec un fonctionnalité de récupération qui empêche la corruption de la carte lors d'une mise-à-jour OTA défaillante.

AdaLink (wrapper de débogage SWD/JTAG)

La transition entre les deux types de carte (sniffer et module Bluefruit LE) n'est malheureusement pas une opération sans risque et nécessite du matériel et logiciel complémentaires et du savoir faire. Cette technique avancé n'est pas prise en charge par l'équipe support d'Adafruit.

Cela étant... si vous êtes déterminé à poursuivre seul sur cette voie et que vous disposez d'un Segger J-Link https://www.adafruit.com/search?q=J-Link (ce qu'Adafruit utilise en interne pour le développement et la production), ou si vous avez déjà effacé vitre périphérique Bluefruit LE, vous pouvez jeter un oeil sur AdaLink https://github.com/adafruit/Adafruit_Adalink (qui est l'outil utilisé par Adafruit pour flasher les 4 fichiers nécessaires pour restaurer un module Bluefruit LE module).

(Note: Les versions plus récentes d'AdaLink supporte également STLink/V2 https://www.adafruit.com/products/2548 par l'intermédiaire de J-Link (J-Link est généralement plus robuste si vous achetez un débogger pour un usage à long terme.)

Le fichiers Hex Intel requis sont disponibles sur le dépôt des firmwares Bluefruit LE https://github.com/adafruit/Adafruit_BluefruitLE_Firmware . Vous aurez également besoin de flasher:

- Une image bootloader adéquate
- Une image SoftDevice adéquate
- Une image du Firmware Bluefruit LE
- Le fichier de signature correspondant contenant un CRC de vérification pour que le bootloader accepte l'image firmware ci-dessus (ce fichier est localisé dans le même répertoire que l'image du Firmware)

Les fichiers adéquats sont généralement listés dans le fichier .xmf de contrôle de version https://github.com/adafruit/Adafruit_BluefruitLE_Firmware/blob/master/releases.xml (dans le dépôt du firmware).

Si vous essayez de flasher le firmware sniffer (à vos propres risques!), vous aurez besoin de flasher un seul fichier .hex, que vous pourrez trouver ici https://github.com/adafruit/Adafruit_BluefruitLE_Firmware/tree/master/sniffer/1.0.1 . Le sniffer ne nécessite pas d'image SoftDevice et n'utilise pas le bootloader fail-safe -- c'est la raison pour laquelle cette modification est à sens unique et une opération risquée si vous ne disposez pas d'un débogger SWD supporté.

Adafruit_nF51822_Flasher

We also have an internal python tool available that sits one level higher than AdaLink (referenced above), and makes it easier to flash specific versions of the official firmware to a Bluefruit LE module. For details, see the Adafruit_nRF51822_Flasher https://github.com/adafruit/Adafruit_nRF51822_Flasher repo.

Comment puis-je accéder aux firmwares BETA ?

Les dernières versions de l'application Bluefruit LE Connect pour iOs et Android vous permet de faire une mise-à-jour de votre module Bluefruit LE avec de nouveaux firmwares en pre-release ou firmware BETA.

Cette fonctionnalité est fournie pour les opérations de débogages et mécanismes de test pour le support de problèmes particuliers sur le forum. Vous devriez uniquement utiliser ces firmwares pour identifier et résoudes des problèmes particuliers sur votre module!

Activer les Releases BETA pour iOS

- Assurez vous d'avoir une version récente de BlueFruit LE connect (au moins la **version 1.7.1**)
- Allez dans la page "Settings" (*paramètres*)
- Faites défiler l'écran des paramètres jusqu'à ce que vous trouviez le libellé **Bluefruit LE**
- Cliquez sur l'icone Bluefruit LE et activez le l'option **Show beta releases** (*affiché les releases beta*)
- Maintenant, lorsque vous utilisez BlueFruit LE connect, vous devriez être capable de voir toutes les releases Beta disponibles dans le dépôt des firmwares.

Activer les Releases BETA pour Android

- Assurez-vous d'avoir la dernière version de Bluefruit LE Connect
- Ouvrir l'application Bluefruit LE Connect
- Cliquer sur l'icone "..." dans le coin en haut à droite de l'écran principal de l'application.
- Selectionner **Settings** (*Paramètres*)
- Faites défiler l'écran jusqu'à la section **Software Updates' *(*mise-à-jour logiciel*) et activer l'option* Show beta releases *(*afficher les releases Beta**)
- Maintenant, vous devriez être capable de voir toutes les releases BETA disponible dans le dépôt github.

Pourquoi ne puis-je pas/plus voir mon périphérique Bluefruit LE après un upgrade à Android 6.0?

Il y a eu quelques changement de sécurité important http://developer.android.com/about/versions/marshmallow/android-6.0-changes.html#behavior-hardware-id dans Android 6.0 et cela affecte les périphériques Bluetooth Low Energy. Si les services de localisations ne sont pas disponibles (signifiant que le GPS est désactivé) vous ne serez pas capable de détecter les paquets d'annonce des périphériques Bluetooth Low Energy. Voyez les détails ici https://code.google.com/p/android/issues/detail?id=190372&q=GPS&colspec=ID%20Type%20Status%20Owner%20Summary%20Stars .

Assurez-vous d'avoir activé le service de localisation sur votre Android 6.0 lorsque vous utilisez Bluefruit LE Connect ou autres applications Bluetooth Low Energy avec le module Bluefruit LE d'Adafruit.

Quel est la vitesse théorique limite du BLE?

Cela dépends de plusieurs facteurs, et déterminé plus les capacités du périphérique centrale (votre smartphone, etc) que par le périphérique.

En prenant en compte la limitation matérielle du nR51822 (6 paquets max par intervalle de connexion et un in intervalle de connexion minimum de 7.5ms), nous atteignons les limites théoriques suivantes sur différents systèmes d'exploitation mobile :

- **iPhone 5/6 + IOS 8.0/8.1**
  6 paquets * 20 octets * 1/0.030 s = 4 kB/s = 32 kbps
- **iPhone 5/6 + IOS 8.2/8.3**
  3 paquets * 20 octets * 1/0.030 s = 2 kB/s = 16 kbps
- **iPhone 5/6 + IOS 8.x with nRF8001**
  1 paquets * 20 octets * 1/0.030 s = 0.67 kB/s = 5.3 kbps
- **Nexus 4**
  4 paquets * 20 octets * 1/0.0075 s = 10.6 kB/s = 84 kbps
- **Nordic Master Emulator Firmware (MEFW) with nRF51822 0.9.0**
  1 paquets * 20 octets * 1/0.0075 = 2.67 kB/s = 21.33 kbps
- **Nordic Master Emulator Firmware (MEFW) with nRF51822 0.11.0**
  6 paquets * 20 octets * 1/0.0075 = 16 kB/s = 128 kbps

Il y a également une limite imposée par le firmware du Bluefruit LE (Adafruit travaille activement pour améliorer le débit dans la release 0.7.0).

Les chiffres ci-dessus sont des maximums théoriques utiles pour décider si BlueFruit Le est un matériel approprié pour votre projet.

Mise-à-jour: Voyez ce billet très utile de Nordic Semiconductors https://devzone.nordicsemi.com/blogs/1046/what-to-keep-in-mind-when-developing-your-ble-andr/ pour des détails spécifiques concernant les limitations concernant les différentes versions d' **Android** et téléphones.

Est ce que ma carte Bluefruit peut détecter d'autres cartes Bluefruit ou périphériques centrals ?

Non. Tous les modules Bluefruit LE d'Adafruit fonctionne actuellement en mode périphérique. Cela signifie que les cartes peuvent uniquement annoncer leur propre existence (via les payload d'avertissement). Le périphérique centrale (habituellement un smartphone ou laptop) a pour tâche d'écouter ces paquets d'annonce, puis démarrer un processus de connexion et initier les transactions entre les périphériques.

Le module BlueFruit LE n'as pas le moyen de détecter d'autres modules Bluefruit (ou périphériques centrales). Ils peuvent uniquement envoyer leurs propres paquets d'annonce et attendre après requête de connexion.

Comment puis-je déterminer la distance (en m) entre entre mon module Bluefruit et mon téléphone ?

La réponse courte est: cela n'est pas possible.

Les périphériques RF mesure normalement la puissance du signal en utilisant RSSI qui signifie "Received Signal Strength Indicator" (*indicateur de puissance du signal de réception). Cette puissance de signal se mesure en dBm (Décibel par milliwatt).*

Plus près est le périphérique et plus fort sera la valeur de RSSI (par exemple: -90dBm est plus faible que -60dBm). Dans le monde réel, il n'y a cependant pas de relation fiable entre les valeurs de RSSI en dBm et la distance entre mes périphériques. Par exemple, le RSSI chutera, s'il y a un mur entre les périphériques. Le RSSI chutera également s'il y a beaucoup d'interférence sur la même bande 2.4GHz. Le RSSI peut également chuter en fonction du périphérique ou de l'orientation de l'antenne.

Vous pouvez lire la valeur de RSSI entre deux périphériques connectés en utilisant la commande `AT+BLEGETRSSI`. La valeur de RSSI ne permet pas de tirer de résultat ou conclusion valable concernant la distance. Elle permet juste de donner une idée

d'éloignement général (éloigné ou proche) en fonction de la force du signal.

A quelle distance de mon téléphone puis-je capter mon module Bluefruit LE ?

Cela dépend de plusieurs facteurs au-delà du module comme l'orientation de l'antenne, la conception de l'antenne du SmartPhone, la puissance de transmission sur le noeud d'envoi, le trafic des différents appareils sur la bande de 2.4GHz, les obstacles entre l'émetteur et le récepteur, etc.

Cette distance peut être aussi réduite que quelques mètres jusqu'à une dizaine de mètres en terrain découvert. Bluetooth Low Energy est conçu pour des communications courtes distances et fonctionnera dans les meilleures conditions entre 5-6 mètres ou moins (pour un communication fiable et un paramétrage normal du Bluefruit LE).

Combien de services GATT et caractéristiques puis-je créer ?

Les limitations suivantes sont applicable au firmware 0.7.0 ou supérieur :

- Nombre maximum de services: 10
- Nombre maximum de caractéristiques: 30
- Taille maximum de la mémoire tampon pour chaque caractéristique: 32 octets
- Nombre maximum de CCCDs: 16

Est-il possible de modifier ou désactiver les services GATT et caractéristiques (DIS, DFU, etc.)?

Non, ce n'est malheureusement pas possible. Adafruit s'appuie sur la spécification du "Device Information Service https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.service.device_information.xml " (DIS) pour déterminer la versions du firmware et la version du bootloader fonctionnant sur le module BlueFruit LE. Bluefruit LE Connect ne pourra pas offrir de mise-à-jour Firmware sans disposer d'information fiable (comme celles offertes par DIS), raison pour laquelle ces deux information sont obligatoire et en lecture seule.

De même, le service DFU est également nécessaire pour maintenir la fonctionnalité de mise-à-jour OTA (over-the-air). Désactiver ce service créerait plus de problèmes que sa présence n'en provoque.

Comment puis-je utiliser BlueZ et gatttool avec les modules Bluefruit ?

BlueZ nécessite un peu d'étude pour pouvoir l'utiliser. Vous trouverez ci-dessous quelques notes sur une approche permettant d'envoyer et recevoir des données en utilisant le service BLE UART disponible sur tous les modules Bluefruit LE et toutes les cartes Bluefruit LE.

Ces commandes pourraient changer avec une version différente de BlueZ. La version 5.21 était utilisée ci-dessous.

```
# Initialiser la clé USB
$ sudo hciconfig hci0 up

# Scan des périphériques BLE UART
$ sudo hcitool lescan
  D6:4E:06:4F:72:86 UART

# Démarrer gatttool, pointant sur le périphérique UART trouvé ci-dessus
$ sudo gatttool -b D6:4E:06:4F:72:86 -I -t random --sec-level=high

  [D6:4E:06:4F:72:86][LE]> connect
  Attempting to connect to D6:4E:06:4F:72:86
  Connection successful

# Scan des services GATT principaux
  [D6:4E:06:4F:72:86][LE]> primary
  attr handle: 0x0001, end grp handle: 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
  attr handle: 0x0008, end grp handle: 0x0008 uuid: 00001801-0000-1000-8000-00805f9b34fb
  attr handle: 0x0009, end grp handle: 0x000e uuid: 6e400001-b5a3-f393-e0a9-e50e24dcca9e
  attr handle: 0x000f, end grp handle: 0xffff uuid: 0000180a-0000-1000-8000-00805f9b34fb

# Obtenir les handles pour les entrées dans le service UART (handle 0x0009)
  [D6:4E:06:4F:72:86][LE]> char-desc
  handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
  handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
  handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
  handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
  handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
  handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
  handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
  handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
  handle: 0x0009, uuid: 00002800-0000-1000-8000-00805f9b34fb
  handle: 0x000a, uuid: 00002803-0000-1000-8000-00805f9b34fb
```

```
handle: 0x000b, uuid: 6e400002-b5a3-f393-e0a9-e50e24dcca9e
handle: 0x000c, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 6e400003-b5a3-f393-e0a9-e50e24dcca9e
handle: 0x000e, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002a27-0000-1000-8000-00805f9b34fb

# 6e400002 (handle 0x000b) = TX characteristic
# 6e400003 (handle 0x000d) = RX characteristic

# Optionel (mais peut être utile) ... scan des entrées CCCD
[D6:4E:06:4F:72:86][LE]> char-read-uuid 2902
handle: 0x000e   value: 00 00

# Activer les notifications pour la caractéristique RX (CCCD handle = 0x000e)
# 0100 = get notifications / obtenir des notifications
# 0200 = get indications / obtenir des indications
# 0300 = get notifications + indications / Obtenir des notification et indications
# 0000 = désactiver les notifications + indications
[D6:4E:06:4F:72:86][LE]> char-write-req 0x000e 0100
Characteristic value was written successfully

# S'assurer que la caractéristique à été mise-à-jour
[D6:4E:06:4F:72:86][LE]> char-read-hnd 0x000e
Characteristic value/descriptor: 01 00

# Ecrire "test" dans le moniteur série du croquis Arduino. Cela
# devrait cette sortie.
Notification handle = 0x000d value: 74 65 73 74

# Ecrire quelque chose sur la caractéristique TX (handle = 0x000b)
# Cela devrait faire apparaître les caractères E F G H dans le
# moniteur série d'Arduino
[D6:4E:06:4F:72:86][LE]> char-write-cmd 0x000b 45
[D6:4E:06:4F:72:86][LE]> char-write-cmd 0x000b 46
[D6:4E:06:4F:72:86][LE]> char-write-cmd 0x000b 47
[D6:4E:06:4F:72:86][LE]> char-write-cmd 0x000b 48

# Envoyer plusieurs octets
[D6:4E:06:4F:72:86][LE]> char-write-cmd 0x000B 707172737475

# Si vous exécutez le croquis callbackEcho et que les notifications
# sont activée alors vous devriez voir cette réponse après avoir
# exécuté la commande ci-dessus:
Notification handle = 0x000d value: 70 71 72 73 74 75

----------

# Si vous désirez seulement une écoute permanente, saisissez la
# commande suivante dans le CLI:
$ sudo gatttool -b D6:4E:06:4F:72:86 -t random --char-write-req -a 0x000e -n 0100 --listen

# Cela devrait produire les sorties suivantes lorsque des données sont
# saisie dans Arduino IDE. Nous ne pouvons cependant pas renvoyer
# de données:
Characteristic value was written successfully
Notification handle = 0x000d value: 74 65 73 74
Notification handle = 0x000d value: 6d 6f 72 65 20 74 65 73 74
```

Puis-je utiliser la broche IRQ pour sortir mon MCU du mode veille lorsque l'UART BLE à des données disponibles ?

Non, sur les cartes à base de SPI, la broche IRQ est utilisé pour indiquer qu'une réponse SDEP est disponible pour une commande SDEP.

Par exemple, lorsque vous envoyez une commande "AT+BLEUARTRX" comm message SDEP, le firmware Bluefruit fonctionnant sur le nRF51822 fera une analyse du message, préparera une réponse SDEP et déclenchera la broche d'interruption (pour indiquer au MCU que la réponse est prête).

Le broche IRQ est complètement indépendante du service BLE UART qui ne dispose pas de capacité d'intérruption (pour le moment).

Téléchargement

Téléchargement

- Fiche technique MDBT https://cdn-shop.adafruit.com/product-files/2267/MDBT40-P256R.pdf
- Objet Fritzing dans la bibliothèque Fritzing d'Adafruit https://github.com/adafruit/Fritzing-Library
- Fichier EagleCAD de la carte sur GitHub https://github.com/adafruit/Adafruit-Bluefruit-LE-Shield-PCB

Schéma

- Schéma du Bluefruit LE Shield https://learn.adafruit.com/assets/28024
- Fabrication Print https://learn.adafruit.com/assets/28023 (dimensions en pouce)
