Track&Roll

Outil pour le suivi d'activité physique de sportifs de haut niveau

Conception Capto

24/01/2018

Porteur du Projet
Geoffroy Tijou

Référent Pédagogique
Sébastien Aubin

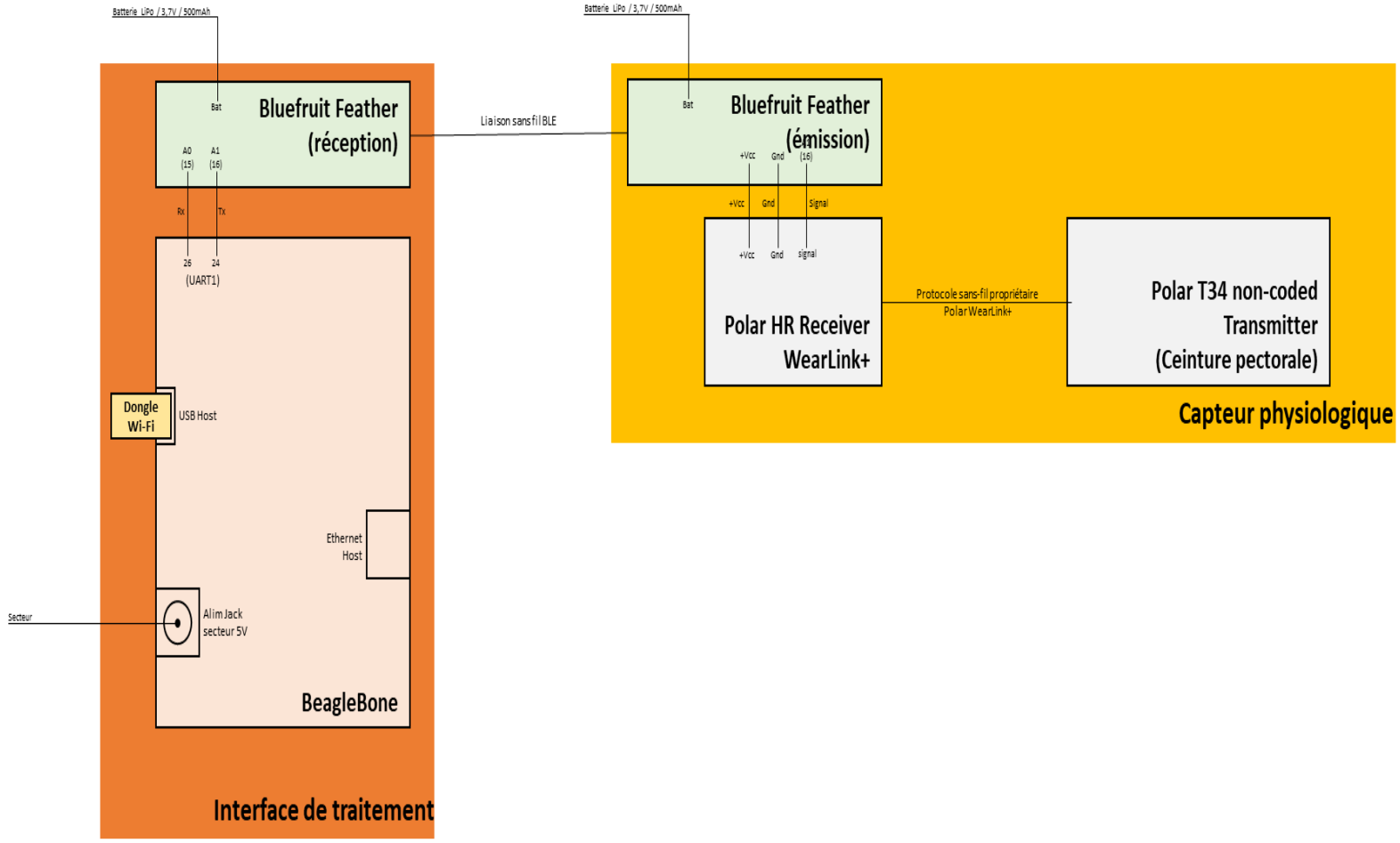
Chef de Projet
François d'Hotelans

Equipe
Marc de Bentzmann
Benoit Ladrange
Guillaume Muret
Antoine de Pouilly
Angéla Randolph

Table des matières

I. Introduction

Dans le cadre du projet Track&Roll, une communication en Bluetooth Low Energy a été sélectionnée entre l'interface de traitement (BeagleBone) et le capteur physiologique mesurant la fréquence cardiaque d'un joueur. Pour ce faire, deux cartes électroniques Bluefruit Feather nRF52 ont été choisies. L'une est placée au niveau du joueur (le module émetteur) qui se charge de récupérer la mesure de fréquence cardiaque et de la transmettre en BLE. La deuxième carte (le module récepteur) reçoit les trames en provenance du capteur physiologique et les transferts par liaison série à la BeagleBone pour traitement.



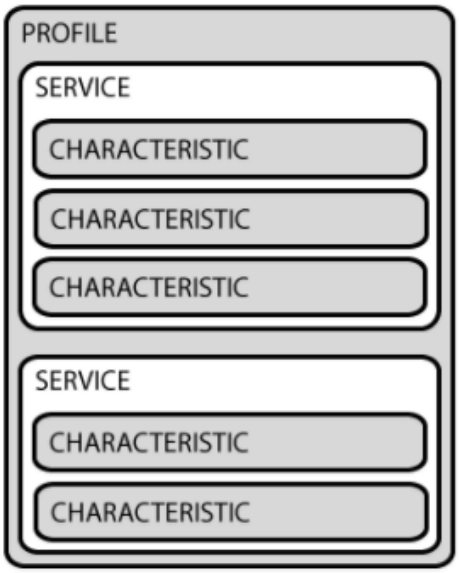
Nous expliquerons dans ce document le code implémenté dans chacun des deux modules. Auparavant, il faut savoir que les cartes Bluefruit sont compatibles avec l'IDE Arduino avec laquelle elles ont été programmées grâce à un câble nano USB utilisé comme sonde de debug.

II. Principe général

Afin de mettre en place une communication efficace entre les deux modules, le fabricant, Adafruit, met à disposition des utilisateurs un certain nombre d'exemples directement exploitables. L'un d'eux consiste à établir une communication de type « BLEUart » entre les deux cartes. Le BLEUart est défini comme une encapsulation du protocole propriétaire de Nordic Semiconductors intitulé « NUS » pour « Nordic UART Service ». Le BLEUart repose donc sur l'utilisation de deux modules Bluetooth Low Energy, l'un jouant le rôle du « central device » (maître) et l'autre celui de « peripheral device » (esclave).

Dans un premier temps, le périphérique émet en continue une trame de données appelée « advertising paquet ». De son côté, le module maître effectue un scan des appareils BLE environnant et analyse les trames diffusées. Suite à cela, le maître peut décider de se connecter au périphérique qui arrêtera alors de diffuser son « advertising paquet » jusqu'à la déconnexion.

Dans l'architecture du protocole BLE, la phase de connexion entre deux appareils est gérée par le protocole GATT (Generic Attribute). Une communication GATT s'effectue forcément entre un appareil central jouant le rôle de serveur et un appareil périphérique jouant le rôle de client. D'autre part, une communication GATT s'appuie sur un ensemble d'éléments appelés « profils », « services » et « caractéristiques ».



Un profil correspond à un ou plusieurs services, nécessaires pour la mise en place d'un cas d'utilisation. Un service est constitué d'un certain nombre de caractéristiques ou de références à d'autres services. Enfin, les caractéristiques de chaque service est une sorte de structure comprenant un type (l'UUID, Universal Unique Identifier), une valeur et un ensemble de propriétés indiquant les opérations que l'appareil BLE peut effectuer ainsi que ses permissions.

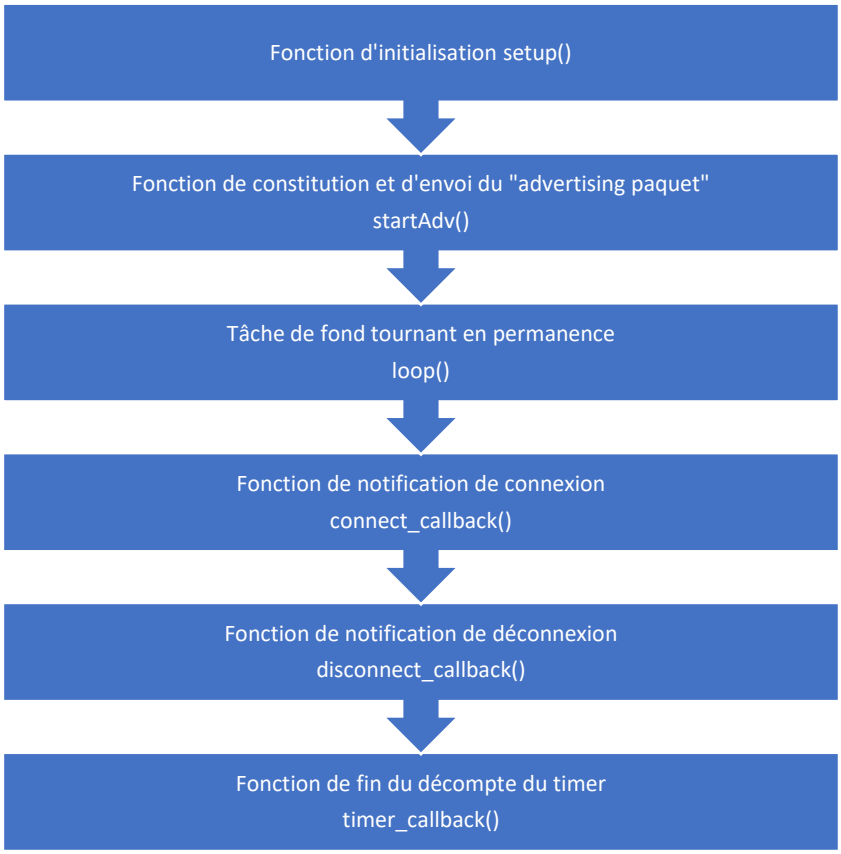
De manière générale, un profil ou service GATT représente un certain cas d'utilisation du protocole BLE pour une certaine application. Dans notre cas, Nordic a créé le profil NUS intégrant un service UART constitué d'un « Rx » (une caractéristique avec une propriété d'écriture) et d'un Tx (caractéristique avec une propriété de lecture) comme une liaison UART classique, mais sans fil.

Ainsi, le module émetteur du système Track&Roll diffusera en continue une trame de données contenant le code correspondant au service « BLEUart ». Le module récepteur effectuera un scan des trames BLE environnante et les analysera, à la recherche de celles contenant le service BLEUart. Une fois détecté, l'appareil central (module récepteur) se connectera au périphérique en question en vue d'un échange d'information suivant le modèle Rx et Tx d'une liaison UART.

III. Module émetteur

A. Architecture du code

Le programme implémenté dans le module émetteur contient l'ensemble des fonctions suivantes :



B. Explication des fonctions

1. setup()

La première fonction appelée par le programme tournant sur le module émetteur est une fonction d'initialisation permettant d'initialiser les GPIO et le timer puis de fixer les propriétés hardware de la carte Bluefruit (puissance d'émission, nom de l'appareil…). Le timer est ensuite lancé pour un décompte de 10 secondes.

C'est également cette fonction qui se charge de configurer et de démarrer les services BLE utilisés par l'application. Ceux-ci sont au nombre de 3 : BLEUart que nous avons vu précédemment, BLEDis (Device Information Service), permettant de véhiculer des

informations spécifiques au fournisseur de l'appareil Bluetooth, et BLEBas (Battery Service), service permettant d'accéder à l'état de la batterie de l'appareil.

La fonction setup() se termine par l'appel à la fonction startAdv().

2. startAdv()

Cette fonction est entièrement dédiée à la construction de la trame de données BLE que l'appareil diffusera à son entourage jusqu'à ce qu'une connexion s'établisse. Ainsi, la fonction se charge de remplir les différents champs de la trame comme le service principal utilisé (BLEUart), le nom de l'appareil, l'intervalle séparant deux envois du « advertising paquet » … Puis, la diffusion de la trame est démarrée, rendant ainsi le module visible aux autres appareils BLE environnants.

Une fois cette fonction achevée, le programme exécutera la tâche de fond.

3. loop()

Le module émetteur est connecté en filaire au récepteur de fréquence cardiaque Polar qui lui indique, par un niveau logique haut sur une de ses pins numériques, qu'un battement a été détecté par la ceinture pectorale.

Ainsi, la tâche de fond effectue une scrutation sur le changement d'état de la pin et, dès qu'un 1 logique est reçu, le programme incrémente un compteur servant alors de compteur de battements cardiaques.

4. connect_callback()

Dès qu'une connexion est entreprise par un appareil BLE fonctionnant en mode maitre, cette fonction est exécutée. Elle permet de notifier à l'appareil qu'une connexion est survenue.

5. disconnect_callback()

Même principe que la fonction précédente, mais cette fois-ci pour indiquer à l'appareil que la connexion a été interrompue.

6. timer_callback()

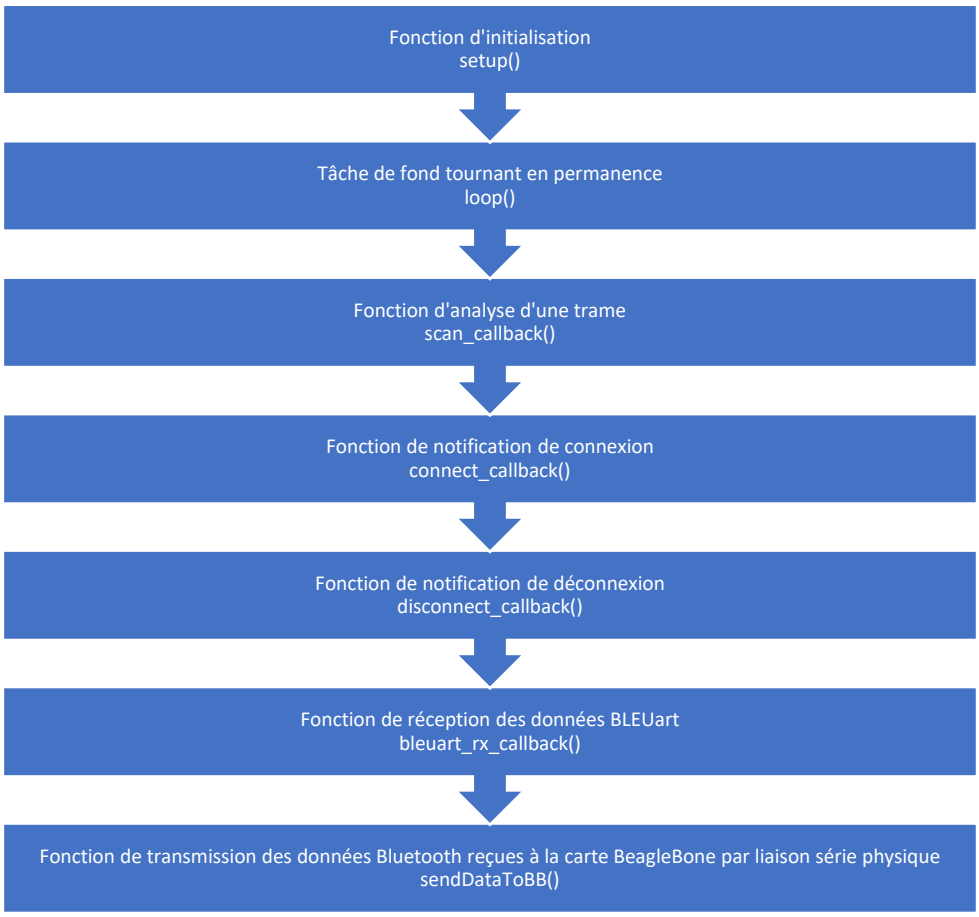
Une fois que le timer a fini de compter ses 10 secondes, cette fonction est exécutée. Elle se charge d'envoyer une trame de données, constituée de l'identifiant du module émetteur et du compteur de battements cardiaques, via le service BLEUart.

Le compteur de battements est ensuite remis à zéro, tout comme le timer qui est réinitialisé et relancé.

IV. Module récepteur

A. Architecture du code

Le programme implémenté dans le module récepteur contient l'ensemble des fonctions suivantes :



B. Explication des fonctions

1. setup()

La première fonction appelée par le programme tournant sur le module émetteur est une fonction d'initialisation permettant de fixer les propriétés hardware de la carte Bluefruit (puissance d'émission, nom de l'appareil…).

C'est également cette fonction qui se charge de configurer et de démarrer les services BLE utilisés par l'application. Ceux-ci sont au nombre de 2 pour le module récepteur : BLEUart que nous avons vu précédemment et BLEDis (Device Information Service), permettant de véhiculer des informations spécifiques au fournisseur de l'appareil Bluetooth.

De plus, la fonction d'initialisation permet de configurer le mode scanner de l'appareil BLE qui va ainsi pouvoir détecter les trames BLE environnantes.

2. loop()

La tâche de fond du programme se charge de vérifier si l'appareil est connecté à un périphérique et, si tel est le cas, la fonction récupère les données reçues, les stocks dans un buffer puis les renvoies au périphérique. Il s'agit d'une forme d'accusé de réception permettant d'indiquer au périphérique les informations qui ont été effectivement reçues par le module récepteur.

3. scan_callback()

Dès que le module récepteur fonctionnant en mode scanner détecte une trame BLE, cette fonction est exécutée. Celle-ci permet d'analyser la trame de données détectée afin de vérifier si elle contient l'identifiant correspondant au service BLEUart. Si l'identifiant est bel et bien présent, le module récepteur se connecte à l'appareil périphérique.

4. connect_callback()

Dès qu'une connexion est réalisée par le module récepteur, cette fonction est exécutée. Plusieurs étapes sont alors réalisées, la première étant la récupération des différentes informations contenues dans la trame diffusée par l'appareil émetteur (informations du fabricant, nom de l'appareil, numéro de série…). Ensuite, si le service BLEUart est détecté, le programme se met en écoute sur le lien virtuel Bluetooth UART existant entre les deux modules. L'appareil récepteur est alors prêt à recevoir les données en provenance du capteur physiologique.

Si le service BLEUart n'est pas détecté dans le « advertising paquet » du module distant, une déconnexion est entreprise.

5. disconnect_callback()

Cette fonction est exécutée dès qu'une connexion est abandonnée afin de notifier l'appareil de la déconnexion avec le module périphérique.

6. bleuart_rx_callback

Lorsque l'appareil a été préalablement placé en écoute sur la réception de données, le programme exécutera cette fonction dès que des informations seront reçues. Tant que des données seront reçues, la programme fera appel à la fonction sendDataToBB() en lui fournissant l'octet reçu en paramètre.

7. sendDataToBB()

C'est dans un deuxième onglet du code dans l'IDE Arduino que se trouve cette fonction. Celle-ci se charge de configurer une connexion série de type UART utilisant deux pins de la carte Bluefruit qui sont déclarées en début de code. La liaison UART est ensuite configurée afin de transmettre l'octet préalablement reçu par BLE à la carte BeagleBone. La donnée transmise est envoyée en format hexadécimal afin de faciliter le traitement réalisé par la carte BeagleBone en réception.