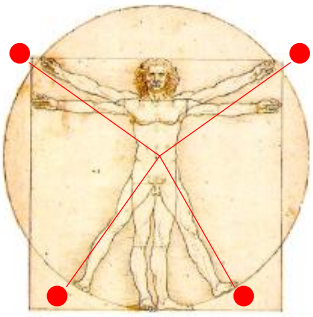
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN XIV. AÑO 2016

SEPARATA



APLICACIÓN ANDROID PARA LA GESTIÓN DE NOTIFICACIONES CON UN DISPOSITIVO TIPO MIBAND

Irene Asunción Yera Pemán, Antonio J. Reinoso

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior
Villanueva de la Cañada (Madrid)

Escuela Politécnica Superior.
Universidad Alfonso X el Sabio.
28691, Villanueva de la Cañada (Madrid).
ISSN: 1696-8085
Editor: Javier Morales Pérez – tecnologia@uax.es

APLICACIÓN ANDROID PARA LA GESTIÓN DE NOTIFICACIONES CON UN DISPOSITIVO TIPO MIBAND

Irene Asunción Yera Pemán[a], Antonio J. Reinoso[b]

[a)] Máster en Desarrollo de Aplicaciones Móviles, Universidad Alfonso X el Sabio.
Avda. De la Universidad nº1, Villanueva de la Cañada, 28691, Madrid. España. irene.yera@gmail.com
[b)]Doctor Ingeniero en Informática, Adjunto a la Jefatura de Estudios
Departamento de Ingenierías TIC, Escuela Politécnica Superior, Universidad Alfonso X el Sabio.
Avda. De la Universidad nº1, Villanueva de la Cañada, 28691, Madrid. España. areinpei@myuax.com

RESUMEN: El Internet of Things (IoT) es una red de dispositivos electrónicos que les permite intercambiar información obtenida a través de sus sensores entre sí. El IoT permite a los usuarios disponer de una malla de dispositivos a su servicio, que obtienen información útil para él, y se comunican con el teléfono móvil que actúa como concentrador de información. Los dispositivos wearables destacan de entre los elementos tecnológicos que recolectan información por su facilidad de uso, comunicación y duración de batería.
En este trabajo realizamos un estudio de la comunicación de las tecnologías wearables y un análisis, diseño e implementación de una aplicación que realiza la comunicación entre un teléfono móvil Android con un wearable tipo MiBand utilizando el protocolo Bluetooth Low Energy (BLE). En concreto la aplicación permite enviar una solicitud de vibrado y encendido de los leds disponibles en la MiBand cuando se reciba una notificación en nuestro dispositivo móvil, y permitir al usuario definir cuándo quiere recibir dicha vibración.

PALABRAS CLAVE: MiBand, notificación, Android, Bluetooth Low Energy, Internet of Things.

ABSTRACT: The Internet of Things (IoT) is an electronic devices mesh that allows the connected devices to exchange information obtained by their sensors. The IoT provides users a mesh of devices at their service: obtaining user centered information and communicating with the mobile phone, that acts as a hub. The wearable devices stand among all these electronic devices for its easy of comunication, user friendliness and batery duration.
In this article we will develop an analysis of the technologies available for synchronization and communication purposes involving wearable devices. Moreover, this paper covers all the aspects related to the design and implementation of an application that performs the communication between an Android mobile phone and a MiBand wearable using the Bluetooth Low Energy (BLE) protocol. Specifically the application is capable of sending a vibrate and led power on signal to the MiBand whenever a notification is received in a mobile phone, and allowing the user to define when he wants to receive the vibration.

KEY-WORDS: MiBand, notification, Android, Bluetooth Low Energy, Internet of Things.

1. Introducción

El "Internet de las cosas" ("Internet of things" en inglés, o "IoT") es una red de dispositivos físicos formada por dispositivos electrónicos, sensores, redes y software, que les permite obtener e intercambiar información entre sí. El IoT ofrece la posibilidad de que los dispositivos estén interconectados, puedan monitorizar la actividad del usuario y obtener información útil para él, y también puedan procesar dicha información y presentarla al usuario cuando lo requiera.

Esta malla digital ha sido identificada por Gartner como la principal tendencia tecnológica del 2016 en su conferencia de prensa de [7], *«Las 10 tendencias estratégicas en tecnología en el 2016»*.

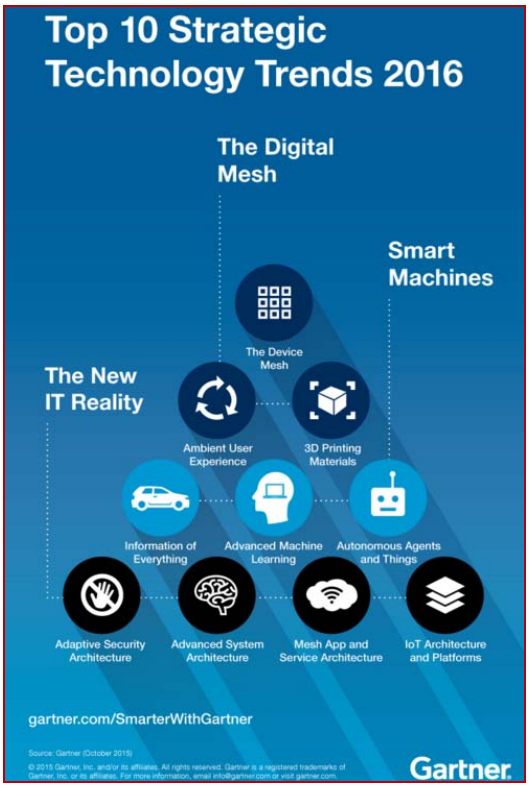


Imagen 1.1: Las 10 tendencias estratégicas en tecnología en el 2016. Fuente: Gartner [7]

Para la implementación de la malla de dispositivos se requiere una gran cantidad de componentes electrónicos embebidos en objetos cotidianos, el uso masivo de sensores y una gran capacidad de interconexión entre todos a través de tecnologías móviles, Bluetooth, NFC y Wifi. Entre estos dispositivos hay uno que destaca por su función principal: el teléfono móvil. Esto es debido a que no sólo se utilizará su potencia de cálculo y sus sensores para la obtención de la información como cualquier otro

dispositivo de la malla de dispositivos, sino que además servirá de nexo de unión (o hub) entre el resto de dispositivos de la red y la Web.

Los dispositivos wearables también son dispositivos que destacan en la malla de dispositivos. Un dispositivo wearable es aquel dispositivo electrónico, incorporado en una prenda de ropa o accesorio, que incorpora una tecnología electrónica que permite una funcionalidad adicional. Los dispositivos wearables son un ejemplo claro de implementación de “Internet of things” ya que adquieren información y pueden interactuar con otros componentes electrónicos sin requerir de intervención humana.

En este artículo mostraremos un resumen de un análisis, diseño e implementación de una aplicación móvil capaz recibir notificaciones en un dispositivo móvil y solicitar al wearable que muestre un conjunto de vibraciones y luces al usuario, en el caso de que el usuario haya indicado previamente que quiere ser notificado cuando se reciba dicha notificación. Para ello analizaremos la estructura de paquetes de la aplicación móvil, indicaremos los principales elementos necesarios para el diseño de la aplicación, estudiaremos el protocolo Bluetooth Low Energy (BLE) y mostraremos las partes más interesantes del código de la aplicación. Como dispositivo wearable utilizaremos la MiBand del fabricante chino Xiaomi.

El artículo está estructurado de la siguiente forma: en el capítulo 2 se indican el conjunto de artículos relevantes ya publicados en el tema, en el capítulo 3 se expone el funcionamiento deseado de la aplicación informática, en el capítulo 4 se especifican los aspectos más importantes de la arquitectura de la solución, al igual que el diseño y la implementación, y finalmente en el capítulo 5 se explican las conclusiones obtenidas durante la elaboración de la aplicación y el trabajo futuro, para por último indicar la bibliografía utilizada en el capítulo 6.

2. Estado del arte

Los dispositivos wearables no son un tipo de tecnología nueva en el mercado. Aunque su uso está considerablemente menos extendido que los teléfonos móviles, los beneficios que se obtienen de sus ventas son suficientemente elevados para que las compañías los consideren como un nicho de mercado a explotar. Gartner predijo en Enero del 2016 en [6] que las ventas en el 2016 de dispositivos wearables serán de 274.59 millones de unidades, frente a los 374 millones de smartphones esperados para ese mismo año:

Device	2015	2016	2017
Smartwatch	30.32	50.40	66.71
Head-mounted display	0.14	1.43	6.31
Body-worn camera	0.05	0.17	1.05
Bluetooth headset	116.32	128.50	139.23
Wristband	30.15	34.97	44.10
Smart garment	0.06	1.01	5.30
Chest strap	12.88	13.02	7.99
Sports watch	21.02	23.98	26.92
Other fitness monitor	21.07	21.11	25.08
Total	**232.01**	**274.59**	**322.69**

Imagen 2.1: Predicción de ventas por dispositivo para 2016 y 2017. Fuente: Gartner [4]

Académicamente, el uso de los dispositivos wearables ha sido estudiado previamente por otros autores. Así, en [1] se realiza un análisis de los efectos en la vida cotidiana de introducir la tecnología wearable, en [4] y [11] se analiza el impacto de los sensores wearables en la ingeniería biomédica, y cómo éstos pueden influir en la monitorización médica.

Desde una perspectiva informática se han diseñado wearables específicos para mejorar la alimentación de los usuarios, como se explica en [12] y se han investigado diferentes diseños para wearables que permitan monitorizar la actividad física de los usuarios, tal y como se explica en [9]. También destacamos los estudios realizados en [10] para diseñar una arquitectura software de comunicación con todo tipo de sensores que incluyan APIs propietarias, como es el caso que nos ocupa. En nuestro caso, realizaremos la implementación completa de una pequeña aplicación que permitirá realizar la comunicación con un wearable propietario (MiBand), indicando los aspectos más relevantes de la arquitectura, diseño e implementación.

La comunicación con los dispositivos wearables se realiza utilizando la tecnología Bluetooth Low Energy (BLE), también llamada Bluetooth Smart. Esta tecnología se utiliza para crear una pequeña área inalámbrica personal alrededor del dispositivo que permite la recepción y envío de información. Antes de estudiar el diseño del paquete, indicaremos algunos aspectos de la comunicación necesarios para comprender el diseño.

La transmisión de la información en Bluetooth se realiza a través de perfiles o profiles, tal y como se describe en [2]. Un perfil es una definición de alto nivel que describe un aspecto de la comunicación entre dispositivos. Si dos dispositivos se quieren comunicar entre sí, deben soportar el mismo perfil de comunicación para hacerlo.

El perfil que se utiliza en BLE es el denominado “Generic Attribute Profile” o GATT y describe, precisamente, cómo dos dispositivos BLE transfieren la información entre sí para proporcionar una aplicación práctica indicando las operaciones básicas

necesarias para la comunicación. Un perfil GATT está formado por uno o varios servicios, que son funcionalidades y datos básicos proporcionados por el dispositivo BLE con el que nos queremos comunicar.

Los servicios se pueden distinguir entre sí ya que incorporan un identificador único universal (UUID). Aquellos servicios estandarizados cuentan con un UUID de 16 bits, mientras que los servicios no estandarizados y propios de fabricantes tienen un UUID de 128 bits. En la definición de cada servicio estandarizado se encuentra, además de la descripción del mismo, el UUID que lo identifica. Utilizando la herramienta "nRF Master Control Panel" ([8]) podemos obtener los servicios de la MiBand, que son los siguientes:

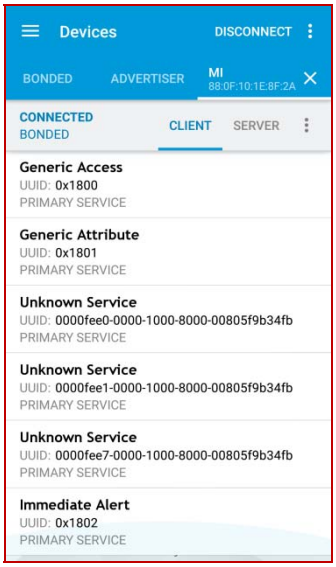


Imagen 2.2: Servicios proporcionados por la MiBand. Fuente: producción propia.

Al igual que un perfil GATT está formado por uno o varios servicios, los servicios a su vez están compuestos por los datos que se pueden enviar o recibir por el dispositivo BLE, lo que se conoce como característica. Una característica es un dato utilizado por un servicio, junto con sus propiedades e información acerca de las operaciones que se pueden realizar con el dato ([3]). Utilizando técnicas de sniffing podemos obtener las distintas características de la MiBand; en la siguiente imagen se muestra la característica y el dato necesario para que se ilumine la MiBand de color azul

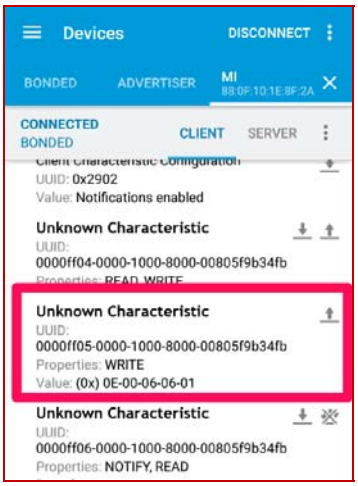


Imagen 2.3: Característica propia de MiBand para mostrar LEDs de colores. Fuente: producción propia.

BLE permite la conexión entre dos dispositivos siguiendo una relación cliente/servidor y el protocolo "ATTribute Protocol" o ATT. El protocolo ATT es el mecanismo de bajo nivel que determina cómo se transfieren las unidades de información (las características). El dispositivo wearable almacena la información a la que se quiere acceder, contiene los servicios y características y actúa como servidor GATT en la comunicación. El dispositivo móvil o tableta actúa como cliente GATT y envía peticiones al servidor utilizando ATT. En este tipo de transmisión la comunicación es iniciada por el cliente GATT que recibe respuestas del servidor GATT alojado en el wearable.

Adicionalmente, la mayoría de los fabricantes, incluido Samsung, incluyen restricciones extra en la comunicación utilizando GATT que también debemos tener en cuenta en el diseño ([10]). Estos fabricantes implementan la comunicación GATT de forma síncrona. Esto implica que por cada mensaje que enviemos desde nuestro teléfono móvil al wearable, éste responderá indicando que ha recibido el mensaje de forma correcta o incorrecta. Pero, desgraciadamente, si se envía un mensaje antes de haber recibido la confirmación de otro mensaje enviado previamente, éste se pierde.

3. Análisis de la funcionalidad de la aplicación

Tal y como hemos indicado previamente, vamos a diseñar una aplicación móvil Android para que la MiBand asociada a un teléfono móvil sea notificada cuando un usuario recibe una notificación en su terminal. La aplicación permitirá que los usuarios puedan configurar cómo quieren que esa notificación se muestre en la MiBand, siempre teniendo en cuenta que este dispositivo sólo cuenta con tres leds y un vibrador. A cada una de las configuraciones que el usuario podrá definir en nuestra aplicación, y que

determinan los parámetros de notificación hacia la MiBand, lo definiremos como subscripción de notificación o subscripción.

El funcionamiento que nuestra aplicación permite es el siguiente:

3.1. Gestión de subscripciones

La aplicación debe ser capaz de proporcionar un conjunto de pantallas para el usuario pueda gestionar y configurar las subscripciones, o lo que es lo mismo, cómo quiere que la notificación sea mostrada en la MiBand. El usuario debe ser capaz de insertar una nueva subscripción de notificación, editar una subscripción de notificación ya insertada, ver cada una de las subscripciones de notificación y eliminar una subscripción:

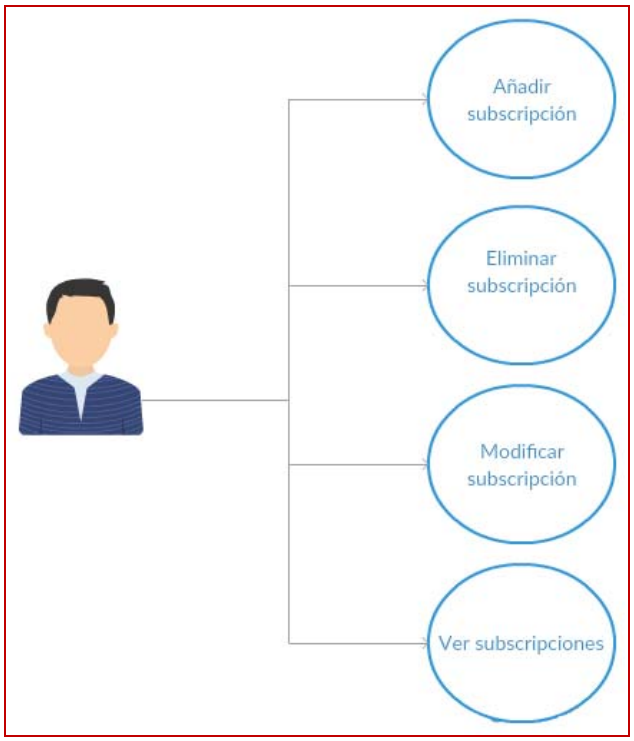


Imagen 3.1: Gestión de subscripciones. Fuente: elaboración propia.

Para cada una de las subscripciones el usuario puede definir:

1) La aplicación que debe enviar la notificación en Android para que la MiBand sea notificada.
2) El usuario o texto que debe constar en la notificación.

3) El color que quiere que se muestre en la MiBand cuando se reciba dicha notificación.
4) El número de veces que quiere la MiBand vibre cuando se reciba dicha notificación.

También se debe permitir definir subscripciones a llamadas entrantes en el teléfono móvil, a pesar de que la notificación recibida en el terminal es distinta.

3.2. Recepción de una notificación

Cuando el terminal móvil recibe una notificación Android, éste lo muestra en el área de notificación del teléfono por defecto. En este caso nuestra aplicación debe determinar si la notificación recibida cumple los criterios de subscripción definidos previamente por el usuario. En el caso de que la notificación recibida sea aplicable a la aplicación definida por el usuario, y se refiera al usuario o texto indicado en la subscripción, nuestra aplicación debe a su vez notificar a la MiBand de esa notificación recibida, haciendo vibrar la MiBand el número de veces definido con el color indicado en la subscripción.

El diagrama de casos de uso para este escenario es el siguiente:

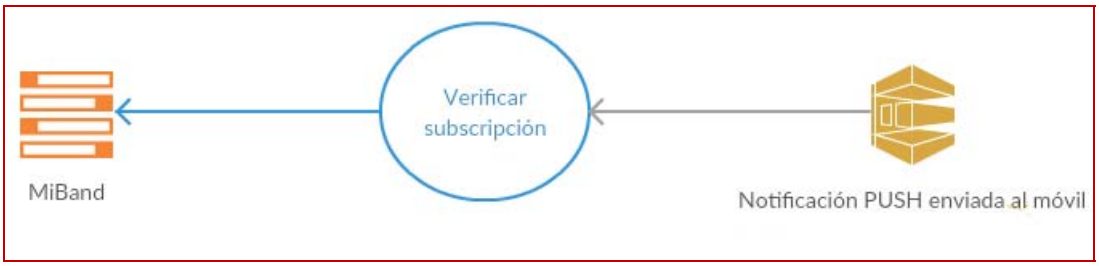


Imagen 3.2: Recepción de una notificación. Fuente: elaboración propia.

3.3. Recepción de una llamada

Este caso es muy similar al anterior, pero recibimos una llamada en el dispositivo móvil en lugar de una notificación PUSH. De forma similar la aplicación verifica si el usuario ha indicado previamente que quiere recibir una notificación en la MiBand cuando se reciba una llamada por parte de un determinado contacto. En ese caso, la aplicación indica a la MiBand el número de vibraciones y el color de los leds para notificar al usuario final de dicho evento.

El diagrama de casos de uso para este escenario es el siguiente:

Imagen 3.3: Recepción de una llamada. Fuente: elaboración propia

3.4. Requisitos de la aplicación

Así pues, nuestra aplicación debe ser capaz de:

1) Descubrir y comunicarse con la MiBand, utilizando la especificación Bluetooth y parámetros definidos para dicho dispositivo, y los mensajes que éste soporte.
2) Escuchar las notificaciones que pudiera recibir el teléfono móvil.
3) Escuchar las llamadas que pudiera recibir el teléfono móvil.
4) Proporcionar un mecanismo de gestión de subscripciones para que el usuario indique las condiciones que deben darse para que la MiBand sea notificada. En este apartado se debe permitir:
    a. La creación de una subscripción.
    b. La eliminación de una subscripción.
    c. La modificación de los datos de una subscripción.
    d. La visualización de las subscripciones realizadas.
5) Para cada una de las subscripciones, el usuario puede indicar:
    a. La aplicación (o llamada) que recibe la notificación.
    b. El contacto que se incluye en la notificación o que realiza la llamada, o bien el texto que debe incluir la notificación (opcional).
    c. El número de veces que debe vibrar la MiBand en el caso de recibir una notificación de esta aplicación y contacto.
    d. El color de los leds para esta subscripción.

4. Diseño de la solución

La aplicación que se pretende desarrollar debe proporcionar tres funciones básicas desde una perspectiva de alto nivel. Estas son:

1) Obtener las notificaciones que se reciben en el teléfono móvil.
2) Comunicarse con la pulsera MiBand.
3) Gestionar las subscripciones

Por ello la estructura de paquetes de nuestra aplicación es la siguiente:

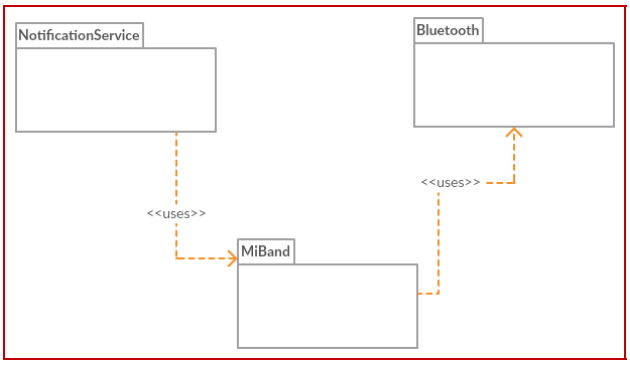


Imagen 4.1: Diagrama de paquetes de la aplicación. Fuente: elaboración propia

4.1.- Paquete bluetooth

Este paquete gestiona toda la comunicación con el dispositivo wearable. Si queremos comunicarnos con un wearable utilizando el protocolo GATT. que describimos en la sección 2, deberemos realizar una serie de pasos previos. Estos son:

1) Realizar una conexión inicial con el dispositivo. En el caso de que el dispositivo wearable no se haya conectado anteriormente con nuestro teléfono, este dispositivo no habrá sido emparejado con nuestro terminal. Se deberá entonces iniciar un proceso para "descubrir" los dispositivos BLE que se encuentren en modo visible para que nos podamos conectar con él.
2) Conectarnos con el dispositivo, lo que permitirá crear una unión entre el cliente GATT y el servidor GATT, y hacer que el servidor GATT pase a modo invisible y no pueda ser detectado por otros clientes.
3) Descubrir los servicios. Como hemos visto en la sección anterior, un dispositivo puede proporcionar múltiples servicios. La comunicación con un dispositivo wearable no se considera completamente establecida hasta que se han descubierto los servicios proporcionados por el wearable.
4) Envío/Recepción de información. Una vez establecido el canal de comunicación, podremos enviar el valor de la característica que solicitemos a un determinado servicio, u obtener el valor de una característica de un servicio indicado. Conviene recordar que para algunos fabricantes la implementación de esta operación es síncrona.
5) Desconexión de la comunicación. Esta operación permite desvincular el wearable del cliente GATT para que pueda ser descubierto por otro cliente GATT.

Veamos un diagrama de secuencia para observar todas estas acciones de forma resumida:

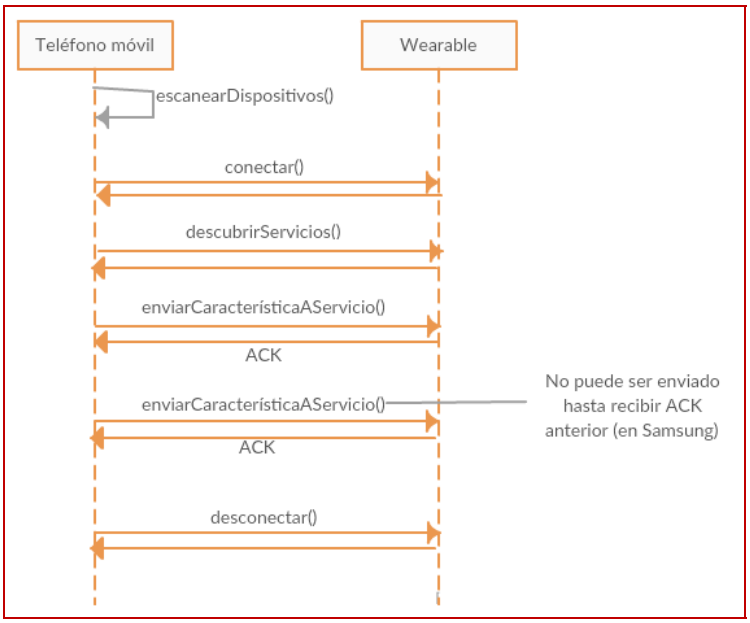


Imagen 4.2: Diagrama de secuencia a alto nivel de la comunicación BLE. Fuente: elaboración propia.

La conexión con la MiBand se puede realizar en una única clase, que contendrá la lógica de la conexión con el wearable, el descubrimiento de servicios, el envío de la información que se requiera y la desconexión. Además incluiremos otra clase que permita identificar a los colores de forma más simple. Para realizar estas acciones Android proporciona un conjunto de clases que deberemos utilizar. Éstas son:

1) *BluetoothAdapter*: permite realizar las tareas fundamentales con Bluetooth, como saber los dispositivos emparejados, iniciar el descubrimiento de dispositivos u obtener una instancia de un dispositivo emparejado.
2) *BluetoothManager*: nos permite obtener una instancia de *BluetoothAdapter*.
3) *BluetoothDevice*: representa un dispositivo Bluetooth remoto y nos permite iniciar una conexión GATT.
4) *BluetoothGATT*: permite la conexión con un dispositivo BLE. Para realizar la conexión se debe crear un callback que recibirá los datos solicitados o el estado de la MiBand (por ejemplo: estamos descubriendo los servicios de la MiBand, o la MiBand ha recibido correctamente el mensaje que hemos enviado previamente).

En el paquete “bluetooth” de nuestra aplicación tendremos el siguiente diagrama de clases:

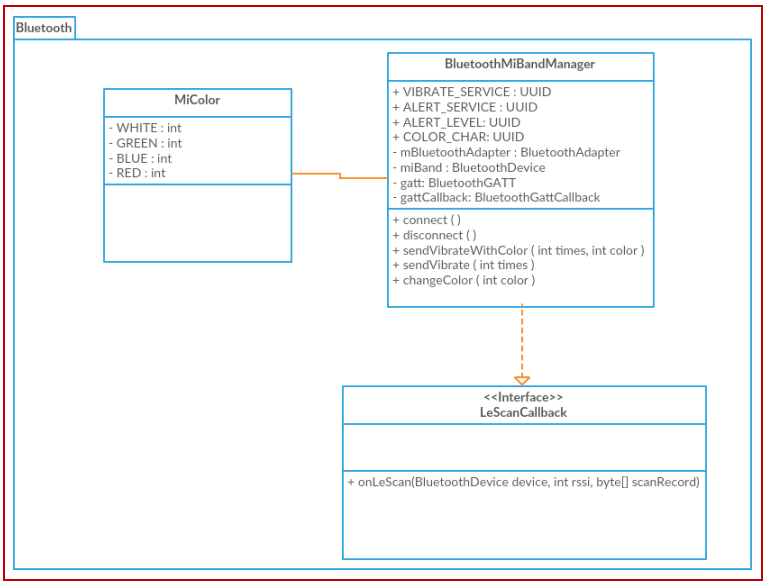


Imagen 4.3: Diagrama de clases del paquete “Bluetooth”. Fuente: producción propia.

Con respecto al diseño e implementación se debe tener en cuenta que, como hemos mencionado anteriormente, la comunicación debe ser síncrona, por lo que se requiere la utilización de un *countdownLatch* que permita bloquear el envío de mensajes hasta que la MiBand indica que ha recibido el mensaje anterior. Se muestra el diagrama de secuencia para el envío de un vibrado seguido de un color, en el que se observa que hasta que no se reciba la confirmación de escritura por parte de la MiBand (*onCharacteristicsWrite*) no se envía ningún otro mensaje:

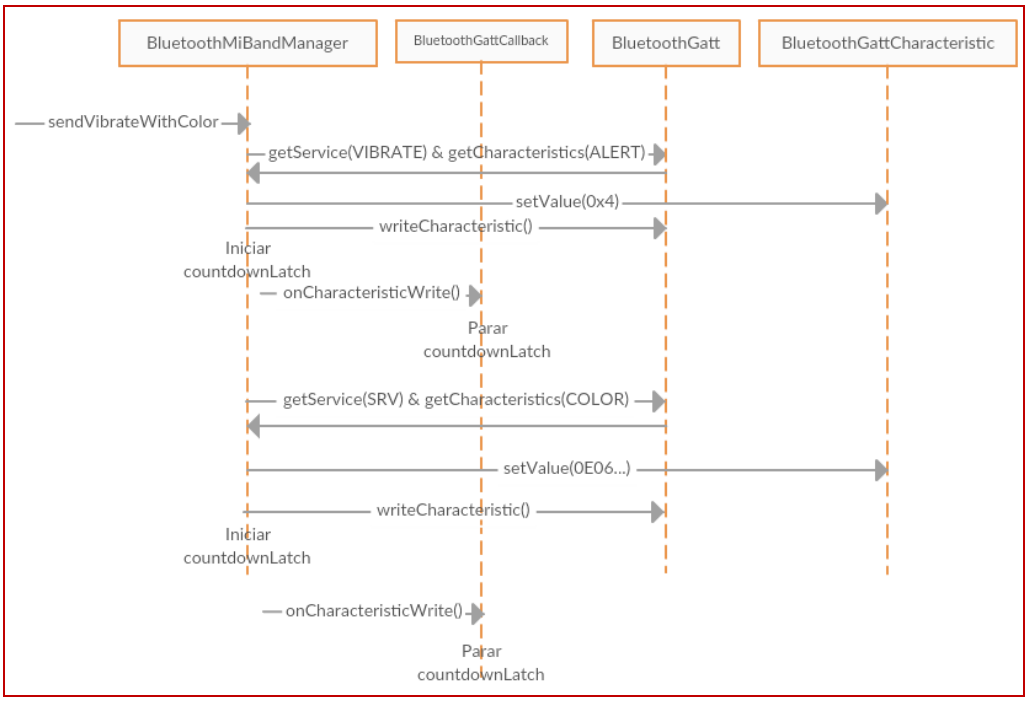


Imagen 4.4: Diagrama de secuencia: envío de un mensaje. Fuente: elaboración propia.

Siendo éste el código del envío de un vibrado:

```
public synchronized void sendVibrate() {
    // Paso 1: Verificar si tenemos posibilidad de enviar mensajes
    if (checkAndRetryConnection()) {
        // Paso 2: Obtenemos el servicio de vibrado de la MiBand
        BluetoothGattCharacteristic chara = gatt.getService(UUID_VIBRATE).getCharacteristic(ALERT_LEVEL);

        if (null == chara) {
            return;
        }

        chara.setValue(STRONG_ALERT);

        try {
            writeLatch = new CountDownLatch(1);

            if (!this.gatt.writeCharacteristic(chara)) {
                Log.i("MiBand", "No existe posibilidad de enviar vibrado 2");
            }

            writeLatch.await(2, TimeUnit.SECONDS);
        } catch (InterruptedException e) {

            e.printStackTrace();
        }

    }
}
```

Imagen 4.5: Implementación de envío de vibrado a la MiBand. Fuente: elaboración propia.

4.2.- Paquete notificationService

Esta aplicación móvil debe ser capaz de recibir las notificaciones o llamadas que se reciban en nuestro teléfono móvil. Toda la funcionalidad relacionada con la obtención de las notificaciones que reciba el terminal la incluiremos en el paquete "notificationservice" que hemos identificado previamente.

La clase de Android *NotificationListenerService*, disponible en la API 19, nos permite diseñar fácilmente parte de la funcionalidad descrita. Esta clase implementa un servicio que recibe llamadas del sistema cuando las notificaciones se reciben o se eliminan. Para utilizar esta clase se debe heredar de ella, definir los permisos correspondientes y sobreescribir los métodos sobre los que queramos añadir funcionalidad. También requeriremos otra clase que permita obtener los eventos referentes a las llamadas recibidas en el teléfono móvil. El diagrama de clases es el siguiente:

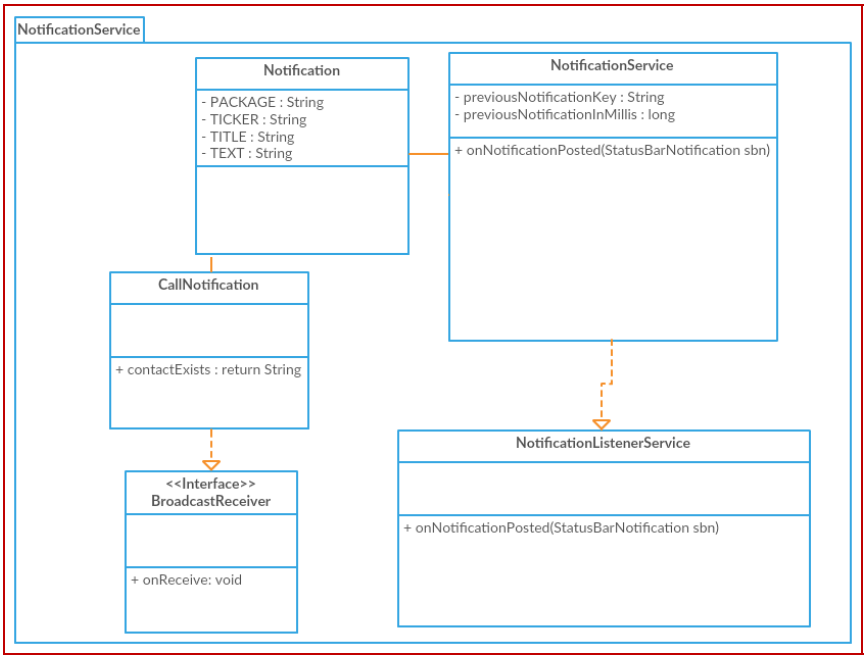


Imagen 4.6: Diagrama de clases del paquete "NotificationService". Fuente: elaboración propia.

Con respecto al diseño e implementación, las clases *CallNotification* y *NotificationService* informarán de la recepción de una llamada o notificación mediante un Intent. El código que permite realizar esta acción es el

siguiente:

```
public void onNotificationPosted(StatusBarNotification sbn) {
    String pack ="", ticker = "", title = "", text = "";
    Bundle extras;

    long currentTimeInMillis = System.currentTimeMillis();

    if (!(sbn.getPackageName().equals(previousNotification)) || (previousTimeInMillis+60000 < cu

        pack = sbn.getPackageName();
        if (sbn.getNotification() != null) {
            if (sbn.getNotification().tickerText != null) {
                ticker = sbn.getNotification().tickerText.toString();
            }
            extras = sbn.getNotification().extras;
            if (extras != null) {
                title = extras.getString("android.title");
                if (extras.getCharSequence("android.text") != null) {
                    text = extras.getCharSequence("android.text").toString();
                }
            }
        }

        Intent msgrcv = new Intent("com.ireneyera.miband.bluetoothAction");
        msgrcv.putExtra(Notification.PACKAGE, pack);
        msgrcv.putExtra(Notification.TICKER, ticker);
        msgrcv.putExtra(Notification.TITLE, title);
        msgrcv.putExtra(Notification.TEXT, text);
        sendBroadcast(msgrcv);
```

Imagen 4.7: Implementación de la recepción de una notificación. Fuente: elaboración propia.

4.3.- Paquete miBand

Finalmente, el paquete MiBand es el que proporciona la funcionalidad de conexión entre el paquete "*notificationService*" y "*bluetooth*", y también incluye la funcionalidad necesaria para que el usuario gestione las subscripciones para ser notificado. Este paquete debe:

1) Almacenar las subscripciones del usuario en una base de datos de la aplicación. El diagrama de clases de este apartado es el siguiente:

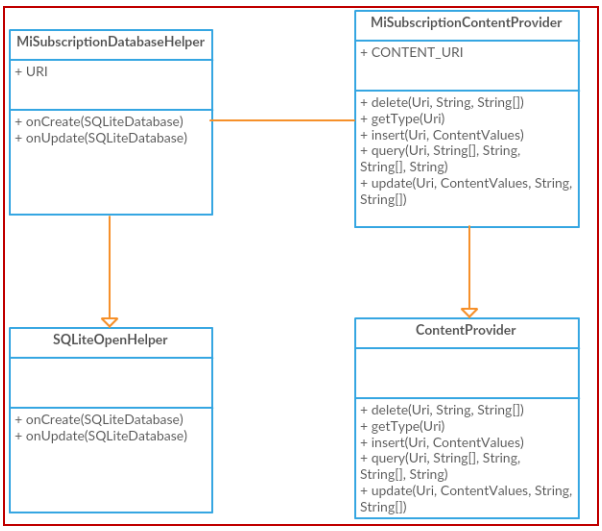


Imagen 4.8: Diagrama de clases de la gestión de BD del paquete "MiBand".
Fuente: elaboración propia.

2) Proporcionar "*Activities*" que permitirán al usuario listar las subscripciones ya realizadas, crear una nueva subscripción, editar una subscripción ya creada y borrar una subscripción. Para ello creamos una activity principal (*MiBand)* que mostrará las subscripciones a través de un *Adapter*, y otra activity secundaria para poder editar subscripciones. El diagrama de secuencia parcial de esta funcionalidad es el siguiente:

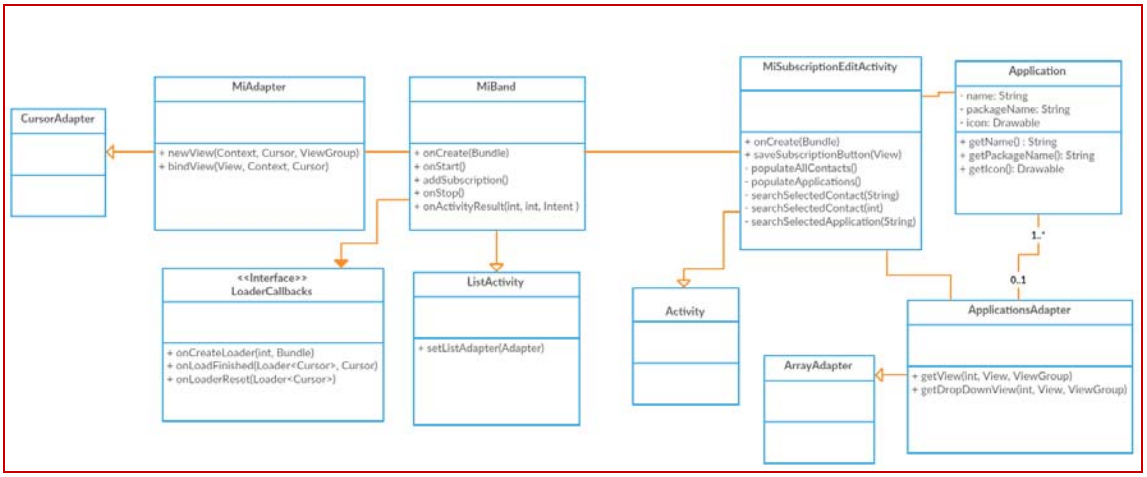


Imagen 4.9: Diagrama de clases que soportan el interfaz gráfico del paquete "MiBand". Fuente: elaboración propia.

3) Recibir los "*Intents*" enviados desde el paquete *NotificationService*, comprobar si la información contenida en el Intent se ajusta a alguna de las subscripciones de la base de datos, y solicitar al paquete "*bluetooth"* que envíe un mensaje de

vibración a la MiBand en caso de que se encuentre alguna subscripción que cumpla los criterios. Para ello diseñamos un servicio, *MiService*, que estará siempre activo y tendrá una instancia de la clase *BluetoothMiBandManager* para poder solicitar envíos de vibrados y colores. Para evitar que este servicio interfiera con el interfaz gráfico, éste se ejecutará en un proceso distinto y la comunicación con la MiBand se realizará en un Thread distinto al hilo principal (*BackgroundWorker).* Este servicio definirá un *BroadcastReceiver* que escuchará constantemente los Intents que se reciban de la clase *NotificationService*, accederá a la base de datos y notificará a la MiBand si fuera necesario.

Como parte del diseño, destacamos la funcionalidad de recepción de una notificación por parte del paquete *NotificationService*, su comprobación con la base de datos y posterior solicitud al paquete *bluetooth* para que sea enviado a la MiBand:

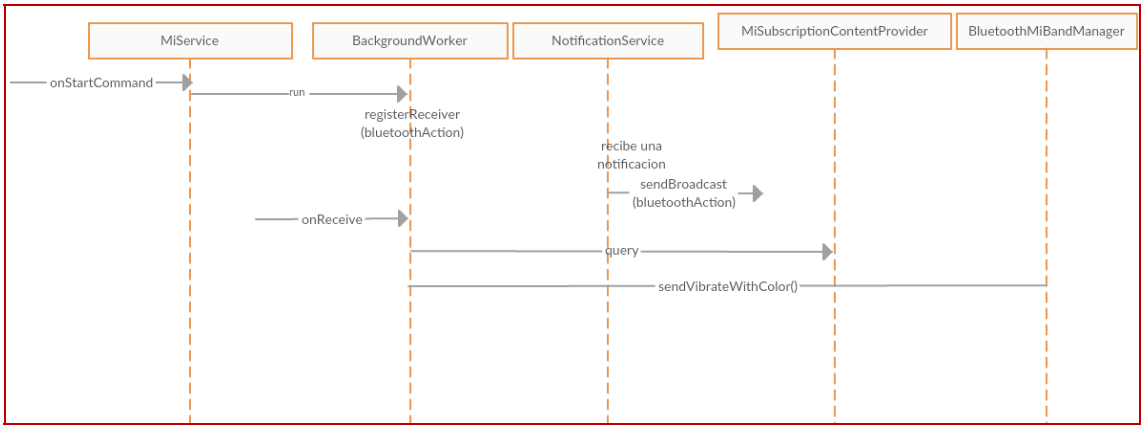


Imagen 4.10: Diagrama de secuencia para la recepción de una notificación e información a la MiBand. Fuente: elaboración propia.

El código que implementa este diagrama de secuencia es el siguiente:

```
public void onReceive(Context context, Intent intent) {
    // Aquí comprobaríamos en BD qué tipo de acción se debe realizar
    Log.i("MiBand", "->El servicio Mi ha recibido una notificación!!!!!");
    String packageData = intent.getStringExtra(Notification.PACKAGE);
    String ticker = intent.getStringExtra(Notification.TICKER);

    // Obtenemos, de la base de datos de la aplicación, la subscripción para esta notifica

    Uri selectedUri2 = Uri.parse("content://" + MiSubscriptionContentProvider.AUTHORITY +

    Cursor c = getContentResolver().query(selectedUri2, null, null, null, null);

    if ((c!= null) && (c.moveToFirst())) {
        do {
            String aContact = c.getString(c.getColumnIndex("contact"));
            String anApplication = c.getString(c.getColumnIndex("application"));

            Log.i("MiBand", " Comprobando con aplicacion " + anApplication + " y contacto
            if (packageData.contains(anApplication) && ticker.contains(aContact)) {
                int color = c.getInt(c.getColumnIndex("color"));
                int number = c.getInt(c.getColumnIndex("number"));
                bTMiBand.sendVibrateWithColor(number, color);
                break;
            }

        } while (c.moveToNext());
        c.close();
    }
```

1 2 3 4

Imagen 4.11: Recepción de una notificación en *MiService* y envío a la *BluetoothMiBandManager*. Fuente: elaboración propia.

5. Conclusiones y trabajo futuro

El diseño de aplicaciones móviles para la comunicación con wearables propietarios no es una tarea simple, y requiere una gran investigación para conocer qué tipo de comunicación y parámetros se requieren en cada escenario. En esta investigación destacamos los siguientes problemas encontrados:

1) El principal problema a la hora de diseñar esta solución ha sido la dificultad de encontrar información de la comunicación con los dispositivos wearables, al igual que con la MiBand. Si bien sí existen APIs para la comunicación con los dispositivos BLE, nos encontramos con que cada fabricante utiliza implementaciones propias que dificultan el estudio genérico de las soluciones. Ha sido especialmente tedioso el estudio de la MiBand, ya que no existen ni APIs oficiales, ni documentación oficial ni foros en inglés con información válida ni contrastada (algunos de ellos incluso contienen información errónea). Prácticamente el 80% del tiempo empleado en el diseño se ha concentrado en encontrar las características que se pueden utilizar en la MiBand, debiendo utilizar técnicas de sniffing y herramientas de terceros para encontrar la información.

2) Con respecto al código, el envío síncrono de la comunicación en GATT ha supuesto implementar una solución utilizando *countdownLatch* cuyo timeout puede ser dependiente del dispositivo. El problema principal reside en que, en ocasiones, el dispositivo wearable no devuelve confirmación de la recepción del mensaje, por lo que se debe establecer un timeout por defecto para que el contador sea puesto a cero automáticamente pasado ese tiempo. Si no, la comunicación se detiene a la espera de un mensaje de confirmación que nunca se recibe. Este problema es dependiente completamente del dispositivo wearable y aunque en la mayoría de las ocasiones se recibe el mensaje de vuelta se ha tenido que incluir un timeout para aquellas pocas ocasiones en las que no se recibe.
3) Finalmente, y más importante, existe un problema de implementación en la pila de BLE para Android en la versión sobre la que hemos implementado esta aplicación, si se mantiene Wifi activado, tal y como se describe en [5]. El problema reside en este caso en que el wearable no envía a tiempo el mensaje de recepción del primer mensaje, por lo que la pulsera vibra pero no se muestran los colores. Simplemente desactivando la Wifi se solventa el problema y se obtienen los colores y vibrados de la subscripción.

La implementación realizada puede ser completada con el resto de servicios proporcionados por la MiBand, e igualmente diseñar un API libre para que otras aplicaciones no tengan que estudiar el comportamiento de la MiBand desde cero.

También se quiere destacar que la aplicación realizada se puede combinar con algunos canales de IFTTT para conseguir una funcionalidad mayor. Por ejemplo, incluir en nuestro canal de IFTTT la recepción de un correo electrónico si la previsión meteorológica indica que va a llover, e incluir una subscripción en nuestra aplicación incluyendo el texto "weather" o "rain", o ser avisado cuando una subasta para la que pujamos está a punto de terminar. Con este canal IFTTT se puede conseguir una mayor funcionalidad y flexibilidad en nuestra aplicación básica.

Además se pueden implementar algunas aplicaciones nuevas específicas para móviles:

- Incluir un conjunto de localizaciones en un mapa y ser avisado en la MiBand cuando nos encontremos en la proximidad de alguna de esas localizaciones. Aunque esta funcionalidad se puede conseguir con IFTTT, la notificación se recibiría más rápidamente.
- Incorporar notificaciones según el estado del teléfono. Por ejemplo, indicar si la batería se encuentra por debajo del 10% de carga o que la batería ha sido totalmente cargada y puede ser desenchufada del cargador.
- Dado que la MiBand cuenta con un acelerómetro, se podría verificar su precisión para crear una pequeña aplicación que pudiera comunicar dos MiBands entre sí: en un extremo se movería la MiBand tantas veces como vibraciones se quisiera

enviar. Al entrar en reposo el teléfono enviaría el número de veces deseado a otro móvil mediante, por ejemplo, un correo electrónico. El código de nuestra aplicación podría ser modificado para leer de la notificación el número, y vibrar tantas veces como se indique. Bastaría con que el usuario emisor y el receptor se pusieran de acuerdo en el significado de los vibrados para poder transmitir el mensaje.

- En el caso de que dispongamos de una MiBand con lector de ritmo cardíaco, podríamos leer de la MiBand cada cierto tiempo el ritmo cardíaco y avisar a los servicios de emergencia o contactos configurados cuando se alcance cierto límite durante un periodo largo.

6. Bibliografía

[1] BABER, C (2001): «Wearable Computers: A Human Factors Review» en *International Journal of Human-Computer Interaction*, volumen 13, artículo 2.

[2] Bluetooth (2016): *Visión general de los perfiles*. Recuperado el 16 de abril de 2016, desde https://www.bluetooth.com/specifications/profiles-overview.

[3] Bluetooth developer (2016). *Generic Attribute Profile GATT*. Recuperado el 30 de abril de 2016, desde https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx.

[4] BONATO, P (2003): «Wearable Sensors/Systems and Their Impact on Biomedical Engineering» en *IEEE Engineering in Medicine an Biology Magazine*, volume 22, artículo 3.

[5] E-howtogeek (2015). *La implementación BLE es instable para Android 4.3*. Recuperado el 28 de mayo de 2016, desde http://www.e-howtogeek.com/111199/android-4-3-bluetooth-low-energy-unstable.

[6] GARTNER (2016). *Gartner anuncia que las ventas mundiales de los dispositivos wearables aumentarán un 18,4% en el 2016.* Recuperado el 1 de abril de 2016, desde http://www.gartner.com/newsroom/id/3198018.

[7] GARTNER (2016). *Las 10 tendencias estratégicas en tecnología en el 2016*. Recuperado el día 15 de marzo de 2016, desde http://www.gartner.com/newsroom/id/3143521.

[8] Google Play (2016): *Aplicación nRF Master Control Panel*. Recuperado el 8 de mayo de 2016, desde https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp.

[9] Intille SS, Albinali F, Mota S, Kuris B, Botana P, Haskell WL (2011): «Design of a wearable physical activity monitoring system using mobile phones and accelerometers» en *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*.

[10] Mzan (2013). *¿Tiene la implementación de Android BLE GATT una naturaleza síncrona?* Recuperado el 30 de abril de 2016, desde http://www.mzan.com/article/18011816-has-native-android-ble-gatt-implementation-synchronous-nature.shtml.

[11] Rodgers, M.M., Pai, V.M, Conroy, R.S. (2015): «Recent Advances in Wearable Sensors for Health Monitoring» en *Sensors Journal, IEEE*, volume 15, artículo 6, pp 3119 - 3126.

[12] Sun M. et al. (2010): «A Wearable Electronic System for Objective Dietary Assessment» en J. *Am. Diet Assoc. review*, volumen 110.