iBKS Hello World!

ABSTRACT

The App demo “iBKS Hello World” is a project that contains the most important functions to begin interacting with a Beacon.

In this document is explained how this project, implemented for Android Studio, is structured and what are the functions that can be found on it.

AUDIENCE

This document is focused for App developers who has no experience in beacon communication management.

Revision 0 | July 2016

accent systems

1. Before you start

All you need to start playing with “iBKS Hello World”:

- Android Studio
- Android device with 5.0 version or above
- At least one iBKS Beacon
- Download **iBKS Hello World project**

2. Project iBKS Hello World

After downloading de “iBKS Hello World” project, you only have to open it on Android studio and compile it. All the needed libraries are included as dependencies and downloaded automatically.

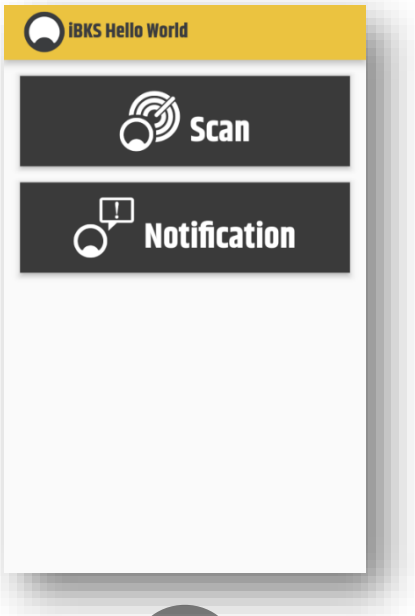
The project is structured to show three important functionalities, each one on a different class:

- **ScanActivity**: scans and list the beacons that are advertising around and allows discovering services and characteristics.

- **NotificationDemo**: Show a notification dialog on the App triggered by a specific beacon packet detected.

- **BackgroundScan:** Starts background scan that allows to detect beacons and does some actions (send notification, open the app, ...) even when the app is stopped.

The first activity started on foreground is “**MainActivity**” that shows the different options of the app and also checks the app permissions.



3. App permissions

In order to manage Bluetooth in Android it's necessary to request some permissions at the user.

3.1 Location

If the Android version is 6.0 or higher, it's necessary to request location permission. To do this it's necessary to add permission in AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

The method which checks the location permissions is `checkLocBT()` in MainActivity.

3.2 Bluetooth

In order to use Bluetooth in Android device, the first thing to do is check if the device that runs the app has Bluetooth Low Energy (beacons work with this type of protocol) and if it is enabled. In order to enable Bluetooth it's necessary to add permission in AndroidManifest.xml

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

The method which checks the location permissions is `inicializeBluetooth()` in MainActivity.

4. Scan Bluetooth devices

All the needed functions used to scan Bluetooth devices are in **ScanActivity**. The steps to achieve that are:

1. Initialize Bluetooth Adapter and set scan settings: `initBT()`

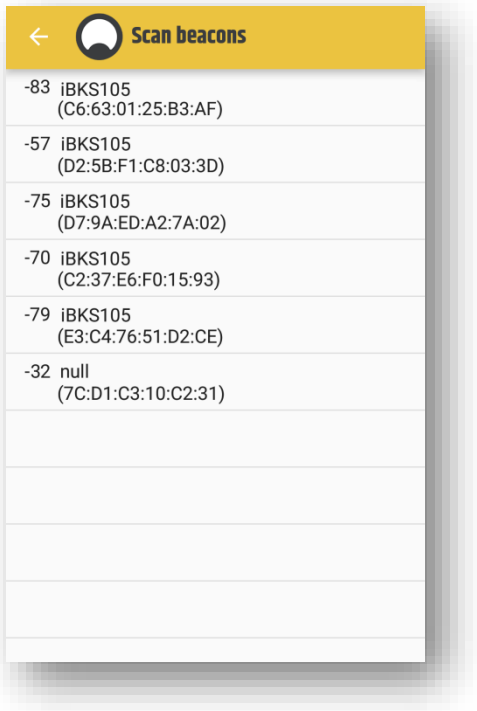
2. Start the BLE Scan with the defined settings and the corresponding ScanCallback: `startLeScan()`

3. Wait for bluetooth packets on `ScanCallback` callback (`onScanResult`) and do some action depending on the RSSI, advertsing, etc.

4. Optionally, connect to the device. In the project, the connection is made when a device of the list is clicked: `onItemClick`[1].

5. Once the connection is made, a `BluetoothGattCallback` callback is received (`onConnectionStateChange`) and it's possible to discover services, read and write the characteristics and enable notifications (all these actions generate a `BluetoothGattCallback` callback). At the end of ScanActivity class there are some examples of these actions.

[1] *Note: On Samsung devices, the connection must be done on main thread*



The app shows a list of all Bluetooth devices detected with its own RSSI and MAC.

5. Notifications

In some cases, it's useful to show a notification to the user when a specific beacon is detected in a particular distance range. This example is shown in **NotificationDemo**.

The most common way to detect a specific beacon is by checking the identifier advertised on the bluetooth packet, either an iBeacon or Eddystone-UID packet. The steps to achieve that are:

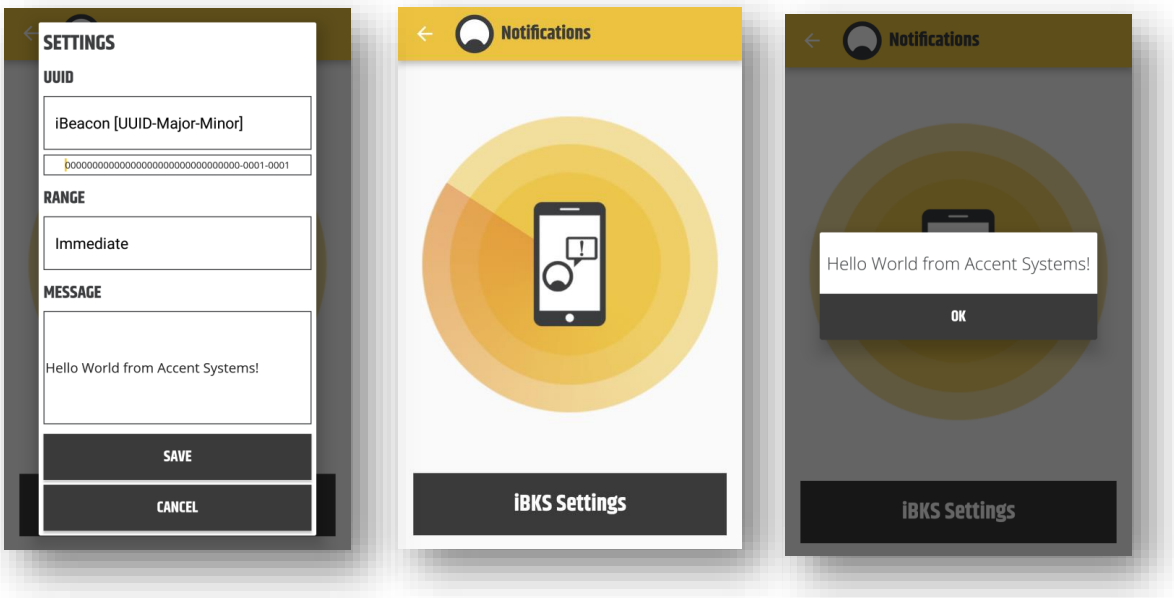
1. Start scanning as seen on ScanActivity.

2. Wait for bluetooth packets on ScanCallback callback.

3. Check if advertising data contains the desired identifier. In this project, the identifier introduced in the settings dialog is used.

4. Optionally, check the RSSI in order to trigger the action only for a particular distance range between the beacon and the user. In this project we defined three different ranges (Immediate, Near, Far)

5. If all the conditions are true, show the dialog with the desired information: `showDialog()`



6. Background Scan

Another interesting function is the background scan. This one allows to scan advertisings when the app is in background and also start some actions when the beacon is detected.

The example of a background scan is in **BackgroundScan** class. In order to execute the background process, it is necessary to add it to AndroidManifest.xml (already added in the project):

```
<application
    android:name=".BackgroundScan"
        . . .
</application>
```

The background scan uses an external open source library. In order to use this library it's necessary to add to the app build.gradle the following dependency (already added in the project):

```
compile 'org.altbeacon:android-beacon-library:2+'
```

The steps to configure the background scan are:

1. Set Beacon Layouts. In the project there's an example of iBeacon and Eddystone-UID layouts, but in the open source library you can find more frame types.

2. Set the scan period in foreground and background. Please be careful in choosing the period because if it's too short the Android device will have a high battery consumption due to Bluetooth is scanning very often.

3. Create the regions. It is the same as creating a list of identifiers that will be checked in every scan period.

4. Binds the BackgroundScan Activity to the BeaconService (`mBeaconManager.bind(this)`). When the *onBeaconServiceConnection* callback is received, enable the regions and start the scan (`enableRegions()`).

5. Wait for the callback called on every scan period, through the function `didRangeBeaconsInRange`, and check if the beacons have been found.

6. Optionally, a notification can be send to the user when a beacon is detected. There's an example code to create a notification in the project.

Revision History

The following revision history table summarizes changes contained in this document.

Revision Number	Revision Date	Description of Changes
Rev 0	07/2016	Initial Release

Bluetooth® SMART iBeacon Eddystone

accent systems

www.accent-systems.com

Accent Systems

Office
Avda. Francesc Macia 46-50, 7th Floor
08208 Sabadell
Barcelona - Spain

Factory
Terra Alta, 1-3 (Pol. Ind. Can Carner)
08211 Castellar del Vallès
Barcelona - Spain