Lab 5: Bluetooth Low Energy

Design and Development of Mobile Applications
E. De Coninck, S. Leroux, P. Simoens
2015-2016

1 Bluetooth low energy

Bluetooth Low Energy (BLE) is the next generation Bluetooth standard. Compared to traditional Bluetooth, BLE allows for much lower power requirements while maintaining a similar communication range. BLE was described in the Bluetooth 4.0 specification. BLE is often marketed as Bluetooth Smart.

Figure 1: The BLE logo

While traditional Bluetooth is mostly used to transfer files between devices or to connect devices to headsets or speakers, BLE has a different target market. The low energy consumption of a BLE device allows it to operate months or even years on a small button cell battery. Thanks to the low cost and the small size of a BLE device it is usable in various sectors such as health care, sport/ fitness, smart homes, ... Some possible applications include:

- Communication with blood pressure sensors, heart rate monitors, ...
- Communication between a smartphone and sensor in running shoes measuring running speed and cadence.
- Indoor localization or proximity detection.

2 BLE Beacons

Indoor localization has always been difficult for GPS. In 2013 Apple introduced iBeacons. iBeacons are simple BLE devices that can advertise their presence to nearby devices. They continually transmit a small frame following a strict format. Every beacon can have a unique ID which allows a mobile device to approximate its current location. In most cases we are not interested in an absolute location but rather in a relative location (i.e. close to a certain beacon). Possible use cases for Bluetooth beacons are:

- Notifying customers of promotions when they are in a certain section of a shop.
- Showing information on the device of a visitor in a museum when they are in a certain room.
- Location based games.

It is important to note that Bluetooth beacons do not contain the content shown on a device. They only advertise themselves using an unique ID. It is up to the device of the user to obtain the content.

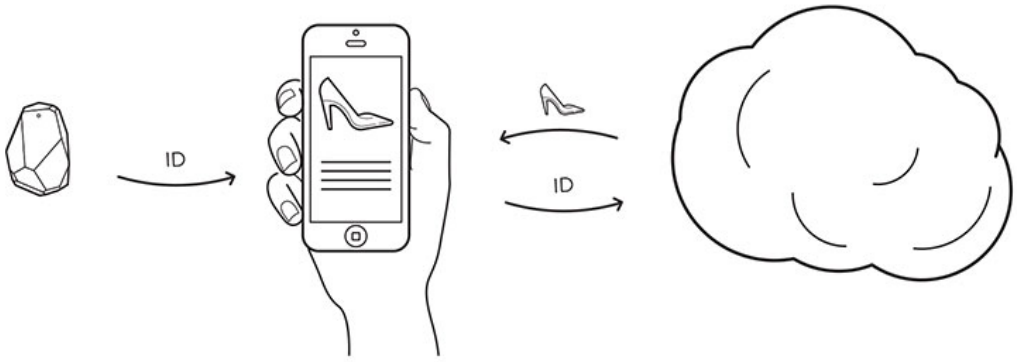

Figure 2: Obtaining information based on the relative position.

Bluetooth beacons are available from different vendors for prices as low as $5. In this lab session we will explore the usage of Bluetooth beacons with Android. We will use Estimote beacons, currently one of the most common beacons.

Figure 3: The iBeacon logo

Figure 4: Estimote beacons

3 BLE and Android

Android 4.3 (API level 18) introduces support for Bluetooth Low Energy and provides APIs that apps can use to discover devices. Android 5.0 makes it possible to turn an Android device into a beacon transmitter.

4 BeaconHunt

In this lab session we will develop a simple location based game: "BeaconHunt". Imagine having a city with several BLE beacons hidden in public places. The goal is to find all beacons. Every beacon will guide you to the next one. We will keep it simpler and show just a list of beacons and their approximate distance. When the user is close to a certain beacon, the beacon is marked as found.

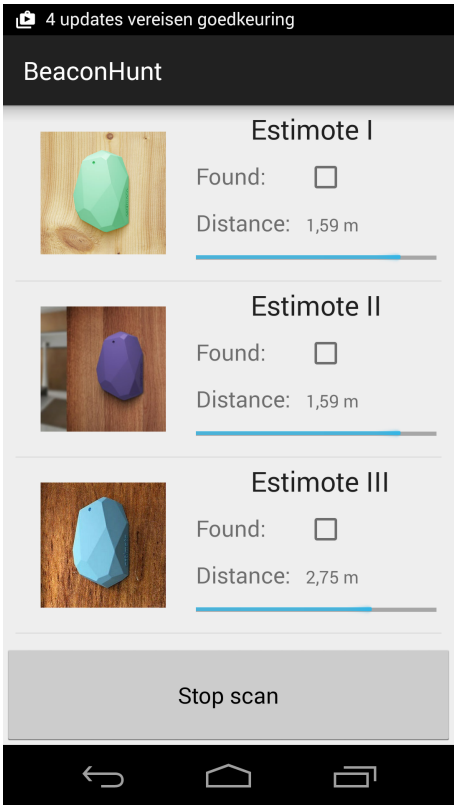

Figure 5: Looking for nearby devices ...

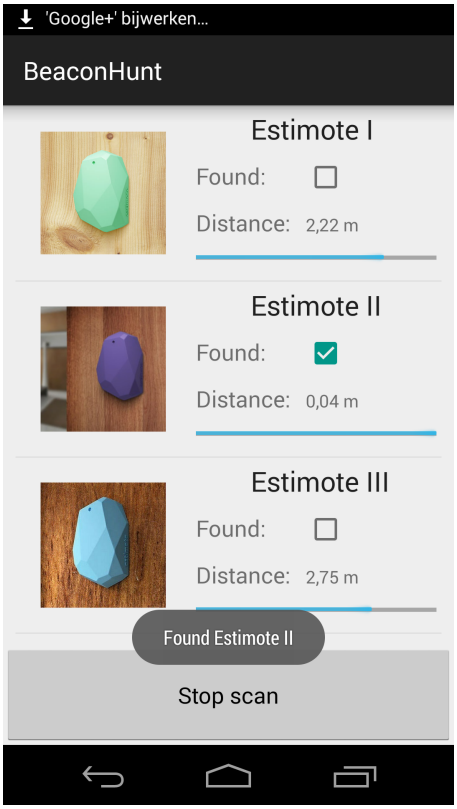

Figure 6: Found !

5 Implementation

The front-end of the application is provided:

- `MainActivity`: The launcher (and only) activity. It shows a list (`ListView`) of beacons and a `Button`.
- `BeaconListAdapter`: Generates a custom list item for each beacon. The layout is described in `Beacon_list_element.xml`. Every list item shows an image (this could show a visual clue of the location of the beacon), an approximate distance in meters, a progress bar giving a visual representation of the signal strength and a checkbox that indicates whether the beacon is found or not.
- Every beacon is represented by an object of type `Model.Beacon`.

Your task is to implement all the functionality needed to scan for Bluetooth LE beacons and to update the list view when new information is available.

- Declare the Bluetooth permission(s) in your application manifest file.
- Obtain the `BluetoothAdapter`.
- Check whether Bluetooth is enabled, ask the user to enable Bluetooth when this is not the case.
- Add a method to handle the pressed event of the button.
- Make efficient use of logging during the development.
- Start scanning when the user pressed the start button
- Process the result of the scan, update the listview

- The result of the scan does not contain an absolute distance to the beacon, we have to use the received signal strength to calculate an approximate distance. The theoretical relationship between the distance and the signal strength is given by $RSSI = -10n\log_{10}(d) + A$ with n a propagation constant, depending on the environment and A the received signal strength at 1m in -dBm. The value of n is usually between 2 and 4, a value of 2 is obtained when a clear line-of-sight is available, when there is a lot of interference, a value of 4 is not uncommon. See this webpage for some common values for n. Implement this function in the *Beacon* class and experiment with different constants.

Question: The received signal strength can fluctuate a lot, what techniques could be used to smoothen out the high frequency changes ?

- When the distance to the beacon is below a certain threshold (0.5m), mark the beacon as found
- Keep the user up-to-date, use `Toast`-messages when appropriate
- Keep scanning until the user pressed "Stop" or the app is moved to the background.
- Scanning for new devices drains the battery. Make sure you stop scanning when it is no longer needed. Check all cases. What happens when the app is moved to the background because the user pressed "Home" or when the user rotates the device?
- Implement extra functionality to improve the accuracy.

Question: The use case presented here is relatively straight-forward. Can you come up with more complex situations where bluetooth beacons can add context awareness to your app ?

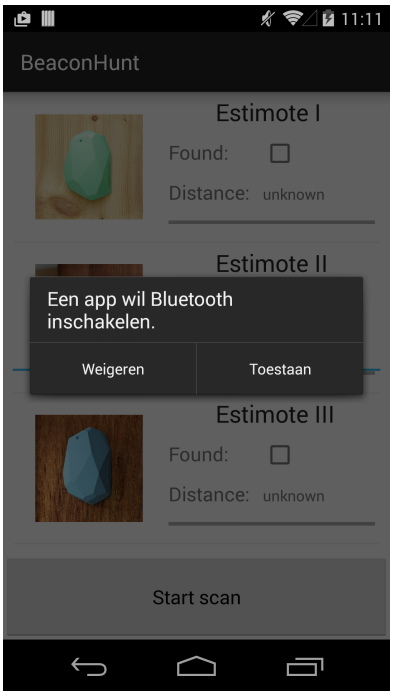

Figure 7: Ask the user to enable Bluetooth.