TK3: Mobile & Ubiquitous Computing

Exercise E3: Tagging & Spatial Context

Telecooperation Lab

Submission Date: 15.07.2019 by 14:00 MESZ

By handing in a solution, you confirm that you are the exclusive author(s) of all the materials.

Introduction

This year, the exercises will be all about creating the IoTK, a simplified Internet-of-Things platform, from scratch. In this exercise, we focus on RFID communication and spatial context.

Hand-In

You have to hand-in the code before the submission date specified above. No late submissions will be allowed and result in 0 points.

After the submission, each group will present their solution to one of the tutors. **If you do not present, you get 0 points for the practical part (no matter of the uploaded code).**

Please note the presentation instructions on top of the slot registration (will be available before the presentations here: https://moodle.informatik.tu-darmstadt.de/mod/organizer/view.php?id=20401).

Task 1. Manchester Code & RFID (8 points)

Task 1.1

As introduced in the lecture, an RFID reader uses the Binary Search Anti-Collision algorithm with Manchester Coding to read many tags simultaneously. The tags send their ID in Manchester Code. In Manchester Code a 1 is encoded with a falling edge at the middle of the clock, and a 0 with a rising edge.

a) Draw the signals for each ID

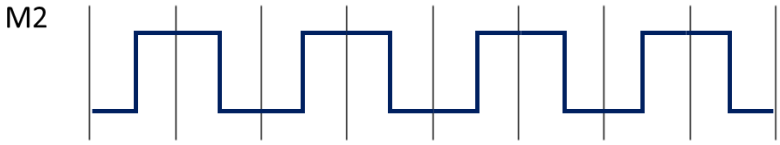
i) T1: ID **158** $(10011110)_2$

ii) T2: ID **222** $(11011110)_2$

b) What signal IDs are encoded by the following Manchester Codes? Please give the ID as **hexadecimal and decimal** value!

i) M1: Decimal: Hex:

ii) M2: Decimal: Hex:



Task 1.2

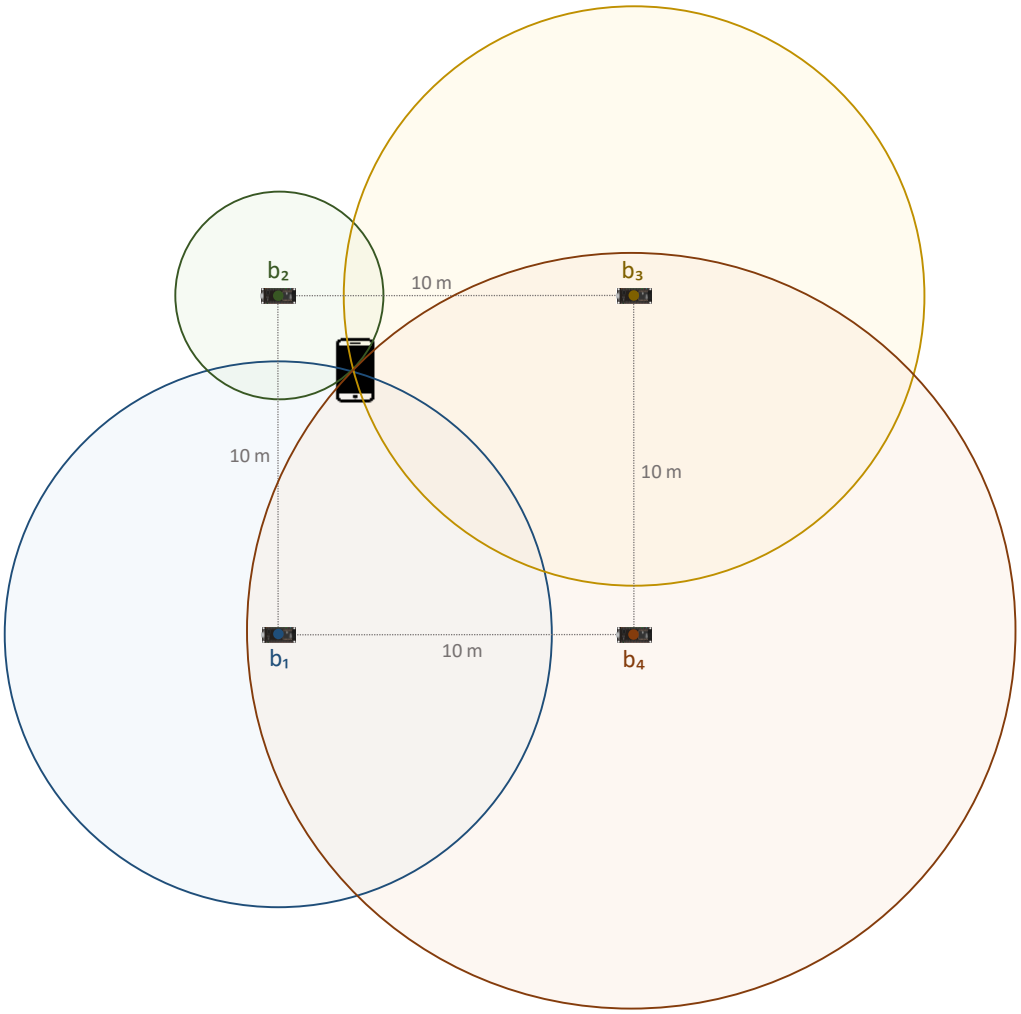
The reader uses the algorithm introduced in the lecture to identify the tags T1 and T2 from above (see Task 1.1), and to extract data. This means after a tag is identified, a READ request is sent to that tag. What REQUEST, SELECT, UNSELECT, READ requests are sent from the reader using what parameters, and what are the responses received? Fill out the following table:

Command	Reply
REQUEST( )	

Task 2. BLE Multilateration (12 points)

As introduced in the lecture, the wireless signal strength can be used to localize a device in space using multilateration. In this task, we utilize the BLE RSSIs of four ESP32s to multilaterate the location of multiple smartphones in a 10m x 10m 2D space (2D for simplicity; four beacons to improve sensing).

The following figure shows the general setup: Four ESP32 beacons are distributed in a 10x10m rectangle. Depending on the smartphone's location, the measured signal strength (RSSI) vary and can be used to multilaterate its position.



In the end, each smartphone should display a map of its own position and the positions of all other devices that are currently tracked in the 2D space.

Task 2.1 Beaconize the ESP32s

In order to multilaterate, the ESP32s need to be configured as BLE beacons (b1 – b4 in the figure above). To get started, you may use the following example by espressif (for arduino): https://github.com/espressif/arduino-esp32/blob/master/libraries/BLE/examples/BLE_iBeacon/BLE_iBeacon.ino

Some hints to get started:

- In order to compute the correct distance in meters, each ESP32 needs to be calibrated. To that end, you have to set the ESP32's signal power (tx) to the measured signal strength in RSSI at 1 meter away (you may use an BLE scanner app like "Beacon Scanner" to get the appropriate RSSI value for your code)

- For real-time usage, you should broadcast the messages more often (<= 1s) than in the example
- To comply with the iBeacon standard, you should use major 19202 and minor 1105 in the code (see setMajor and setMinor)
- The UUID should be unique per beacon (see https://www.uuidgenerator.net/ for new ones)

As the location of each beacon is not always globally known, each beacon should connect to a Wi-Fi and MQTT broker fully automatically (see E2), and publish its geo location and active presence in MQTT as follows:

- The beacon's name (a readable standard string like "b1") in /laterator/beacons/<UUID> /name
- Its last active unix timestamp in /laterator/beacons/<UUID>/lastActivity
- Its x, y position (in meters) in /laterator/beacons/<UUID>/x and /laterator/beacons/<UUID>/y

Each beacon should use a unique <UUID> (replace above for each beacon). You may assume artificial coordinates for your beacons that fit the requirements of the figure (e.g. 0,0; 0,1; 1,0; 1,1) as long as the setup also fits.

You should take care that:

- As we only investigate 2D, place all ESP32 beacons b1-b4 such that they are in the same plane.
- Depending on your room arrangement, you may place the beacons differently (e.g. shorter distances) to test your solution.

The following requirements must be fulfilled for this task:

- Your code should work with a variable number of beacons (min. of 3 up to n)
- All beacons discover the MQTT broker and publish their beacon information automatically
- The beacons comply with the iBeacon standard and are calibrated correctly

Task 2.2 Multilateration App

Each smartphone computes its position via multilateration based on all available beacons (see Task 2.1). As there is not always a single mathematical solution due to noise, you should use a non-linear solver to find an optimal position (see https://github.com/lemmingapex/trilateration).

Each smartphone should publish its current position via the same MQTT broker as in Task 2.1 as follows:

- The smartphone's last active unix timestamp in /laterator/devices/<UUID>/lastActivity
- The smartphone's name (a readable string like "Luke") in /laterator/beacons/<UUID>/name
- Its x, y position (in meters) in /laterator/devices/<UUID>/x and /laterator/devices/<UUID>/y

Each device should use a unique <UUID> (replace above for each device) and publish its position at least every 5 seconds.

Moreover, a user interface should display a map with all beacons (with correct position) and the locations of all smartphones currently active in /laterator/devices/ (rule out devices with timestamps >= 30 seconds). The user interface may look like in the sample figure.

Please note:

- You are free to use any suitable smartphone. We recommend Android. If you need an Android device, please contact us.
- We will not grade the appearance of the user interface. However, all information (i.e. location/names of all beacons & phones) should be visualized physically correct in 2D.
- You need to add the beacon type to the beacon library:

```
BeaconManager.getInstanceForApplication(this).
getBeaconParsers().add(new
BeaconParser().setBeaconLayout("m:2-3=0215,i:4-
19,i:20-21,i:22-23,p:24-24"));
```

The following requirements must be fulfilled for this task:

- The locations and names of all beacons and smartphones are displayed on a map (i.e. a simple rectangular) in the app
- The smartphone automatically discovers the MQTT broker, and publishes and receives all required information.
- Your code operates with all beacons and smartphones that are also present and fulfill the requirements of the iBeacon & MQTT specification above

Links & Tutorials

ESP32 & BLE

- BLE Library https://github.com/espressif/arduino-esp32/tree/master/libraries/BLE

Android, BLE & MQTT

- MQTT Library https://www.eclipse.org/paho/clients/android/
- Beacon Library https://github.com/AltBeacon/android-beacon-library
- Multilateration Library https://github.com/lemmingapex/trilateration
- Beacon Scanner App https://play.google.com/store/apps/details?id=com.bridou_n.beaconscanner
- Getting started with drawing in Android
    - https://alvinalexander.com/android/how-to-extend-android-view-class-custom-view-ondraw
    - https://developer.android.com/training/custom-views/custom-drawing