Описание архитектуры BLE части

На данный момент все случаи навигации по сигналам от BLE маяков можно разделить на два класса:

1. Устройство пользователя принимает сигналы от трех и более маяков, которые относительно пользователя на плоскости образуют треугольник (**Trilat-навигатор**); данный вариант обычно имеет место при больших открытых пространствах, как, например, зал или холл, большая аудитория;
2. Устройство пользователя принимает один, два или более сигналов, которые относительно пользователя на плоскости располагаются на одной линии или близко к ней (**Proximity-навигатор**); обычно позволяет сэкономить на количестве маяков, которые необходимы для организации навигации; применяется для покрытия узких и длинных коридоров, малоразмерных комнат, возможно снаружи зданий.

Примеры положений маяков и помещений для каждого случая.

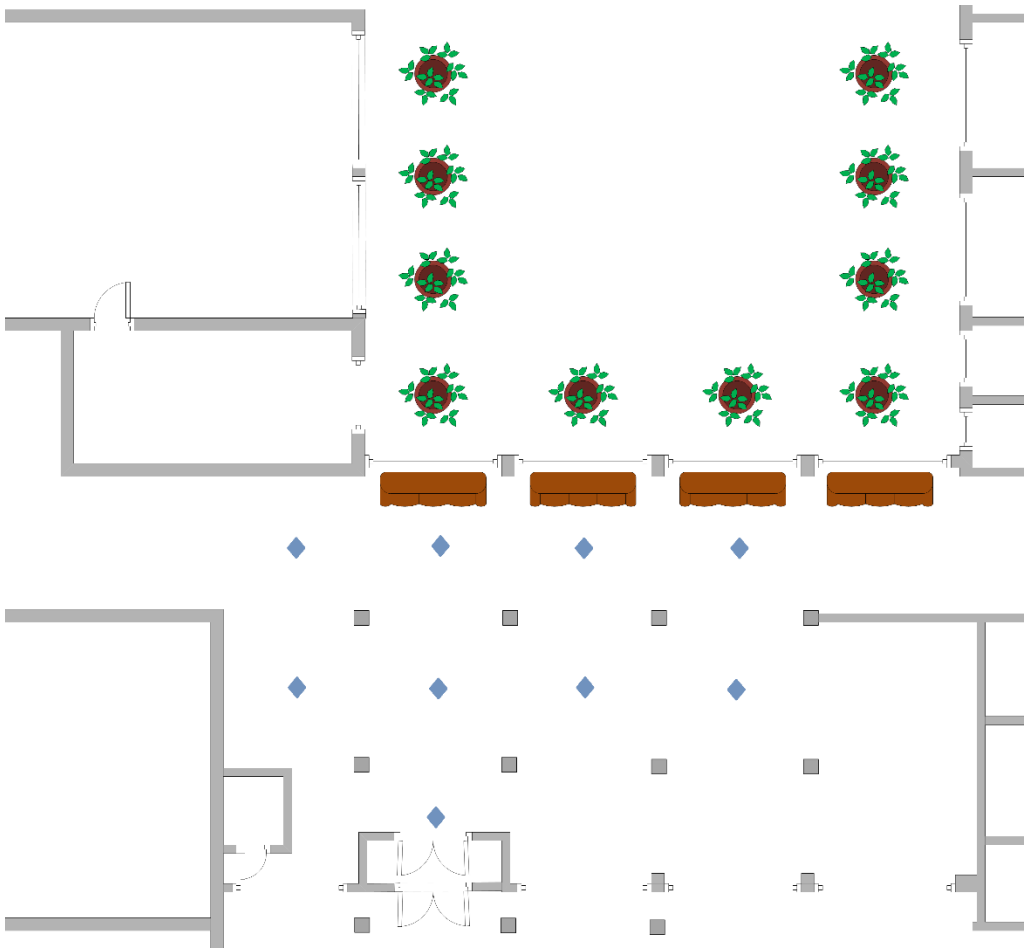

Рис. 1 – Карта и пример расположения маяков для случая 1, Trilat-навигатор (маяки показаны синим ромбом)

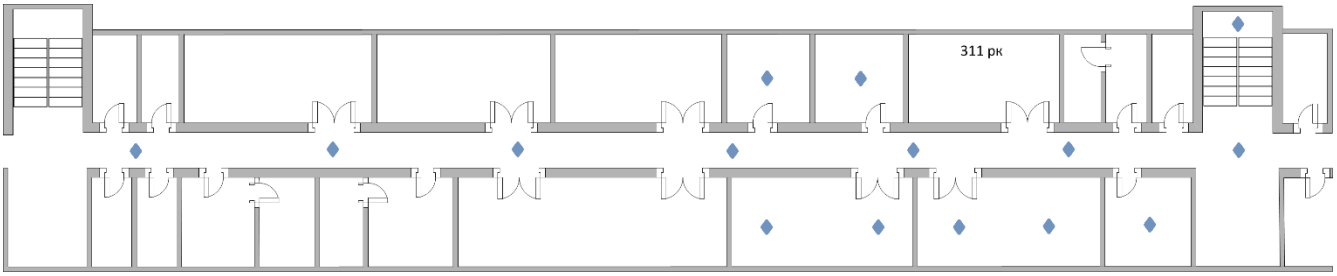

Рис. 2 – Карта и пример расположения маяков для случая 2, Proximity-навигатор (маяки показаны синим ромбом)

На данный момент есть несколько идей, как переключаться между двумя данными навигаторами, т.е., определять, когда какой из них необходимо использовать. Однако уже можно сказать, что структурная схема BLE части будет выглядеть в общем виде как:

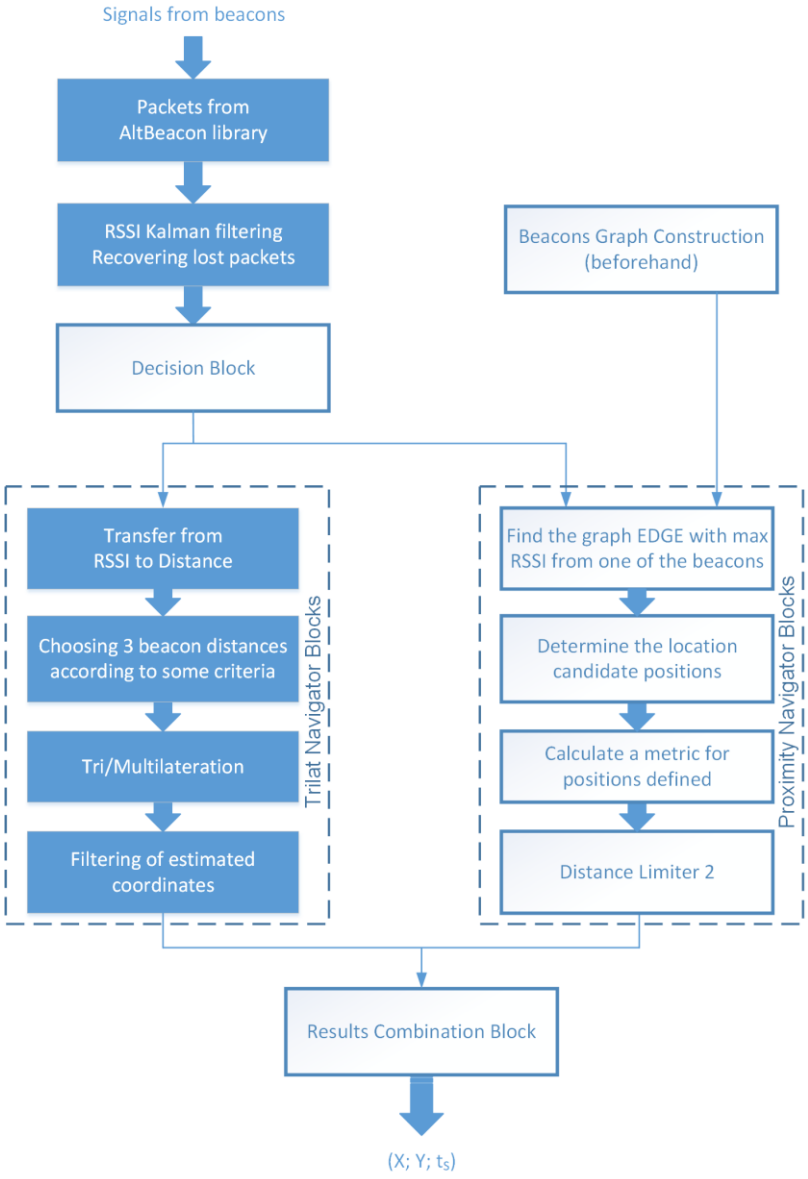

Рис. 3 – Структурная схема BLE части

В ближайшее время в SDK будем делать два отдельных и не связанных между собой навигатора: один – Trilat Navigator (старый), второй – Proximity Navigator (новый). Пользователь будет явно определять на экране (Settings), какой из навигаторов применять для BLE сигналов (и для Particle Filter).

У Proximity Navigator - общий блок фильтрации принятых значений RSSI, а именно Kalman RSSI filter.

Beacons Graph Construction блок будет строить граф, узлами которого являются положения маячков на текущей карте, а ребрами – возможные для пользователя (с учетом мелкого мэша) перемещения между маячками.

Далее, по принятым (и отфильтрованным) RSSI значениям алгоритм определяет маяк (узел графа) с максимальным RSSI. Затем на смежных с ним (узлом) ребрах находит маяк со следующим по мощности значением RSSI и по этому критерию делает выбор этого ребра.

Далее находятся пересечения сигналов (окружностей) от остальных маяков с выбранным ребром. Другими словами, определяются потенциальные возможные положения пользователя. Затем ко всем обнаруженным на ребре положениям применяется некоторая метрика, например, усреднение. И вычисляется конечное положение пользователя.

К полученной оценке может применяться Distance Limiter 2 фильтр. Алгоритмически он будет отличаться от одноименного фильтра, уже реализованного в SDK для Trilat Navigator.

Затем полученные координаты выдаются, как и раньше либо на экран UA, либо в PF.