К вопросу об архитектуре BLE части SDK

На данный момент, потенциально тестировать production-версию системы будем в одном из учебных корпусов университета ХАИ. Часть его карты показана на рис. 1.

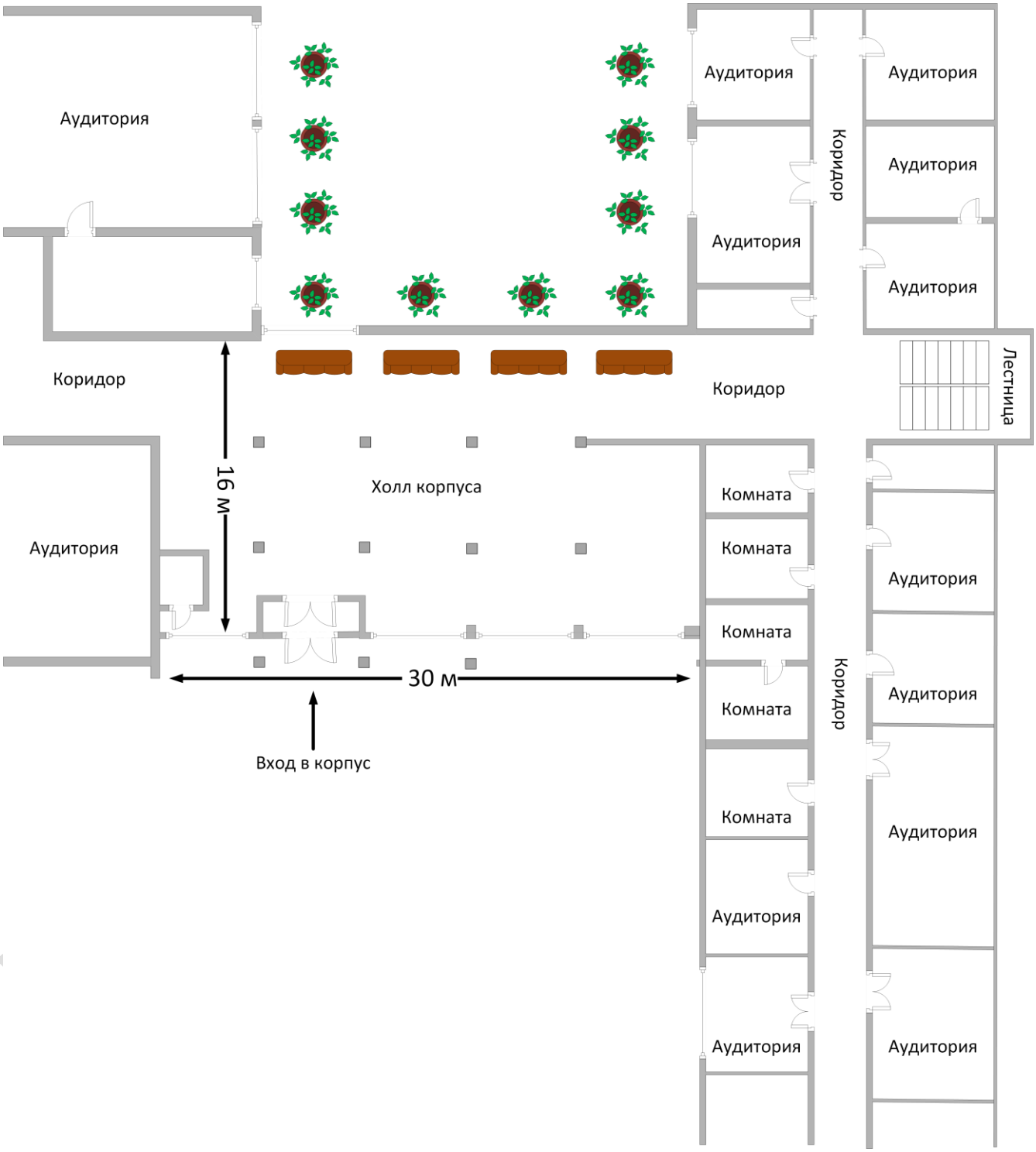

Рис. 1 – Часть карты одного из учебных корпусов ХАИ

Хорошо видно, что территория корпуса включает в себя как большое открытое пространство (холл корпуса, 30 х 16 м, учебные лекционные

аудитории), так и узкие коридоры (шириной около 2 м) и отдельные учебные классы. Предполагаю, что похожие ситуации могут часто встречаться, например, в мед. учреждениях, торговых центрах.

Для указанной карты все случаи навигации по сигналам от BLE маяков можно условно разделить на два класса (рис. 2):

1. Устройство пользователя принимает сигналы от трех и более маяков, которые относительно пользователя на плоскости образуют треугольник (**Trilat-навигатор**); данный вариант обычно имеет место в 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.

Моя идея для выбора, какое решение для навигации использовать, состоит в следующем:

- Нашли маяк с максимальным RSSI для текущего ts (на выходе Kalman RSSI); обозначим его через B0;
- По графу смотрим, с какими ближайшими маяками B0 связан; по определению, ребер-соседей будет минимум одно (например, если маяк находится в конце коридора), максимум – зависит от конфигурации;
- Если ребер-соседей больше одного (минимум 2), проверяем формируется ли треугольник из маяка B0 и маяков соседей; возможно, что треугольников с участием B0 будет больше одного;
    - Если треугольника нет, то применяем Proximity решение; **конец**;
    - Если минимум один треугольник есть, то проверяем его через TripletsFIlter (на углы между гранями треугольника) из TrilatNavigator;
        - Если треугольник есть и проходит через TripletsFIlter, то применяем TrilatNavigator; **конец**;
        - Если треугольников несколько, проверяем до первого корректного треугольника и применяем TrilatNavigator; **конец**;
        - Если все треугольники не проходят проверку в TripletsFIlter, то применяем Proximity решение; **конец**;

На рис. 3 показана блок-схема, которую рисовал для будущей архитектуры BLE Navigator. После описания своей идеи понимаю, что вид на рис. 3 будет не таким. Больше блоков из TrilatNavigator будут идти до ветвления.

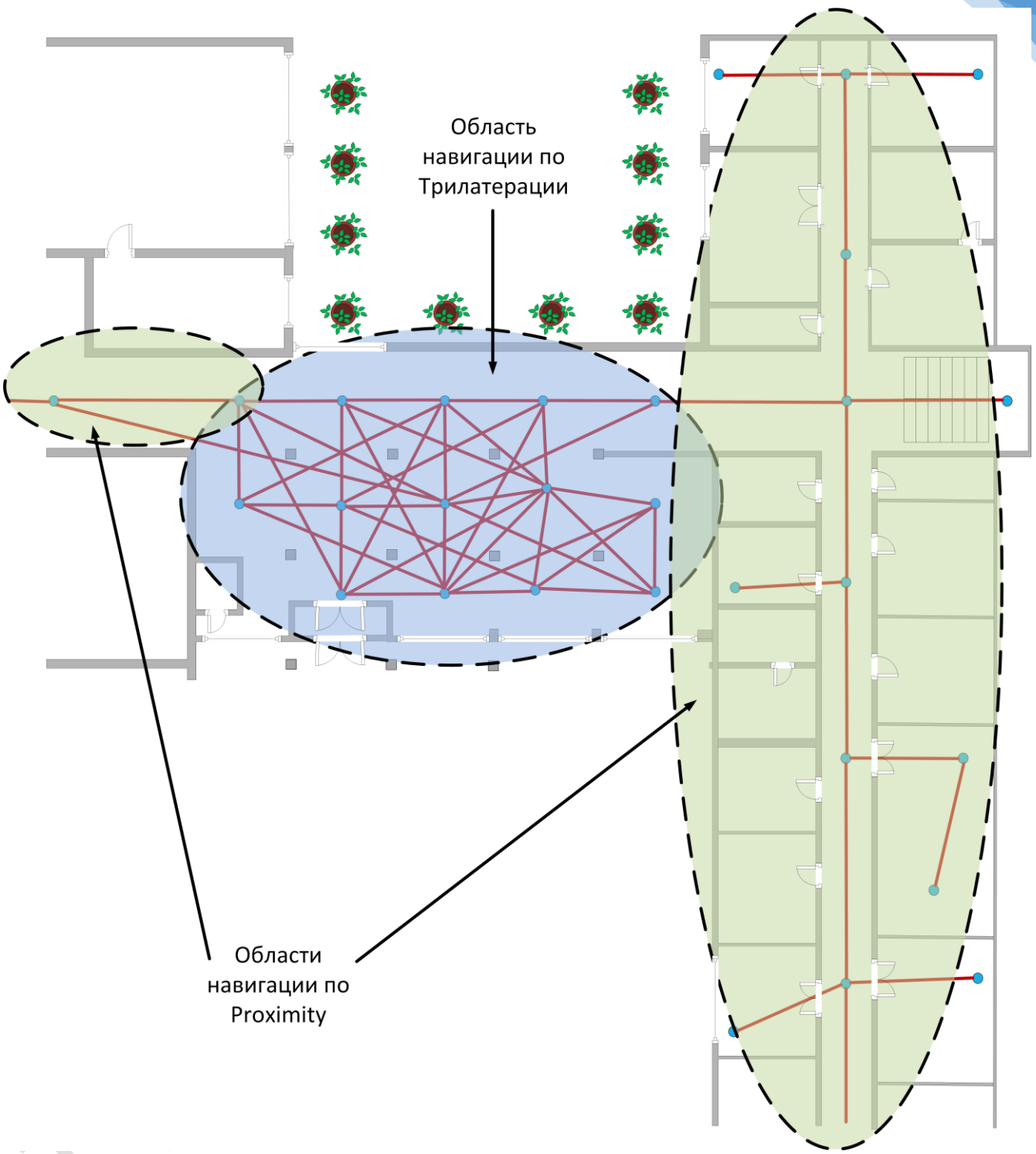

Рис. 2 – Карта и пример расположения маяков и ребер графа
(маяки показаны синими кругами, ребра графа – красными жирными линиями)

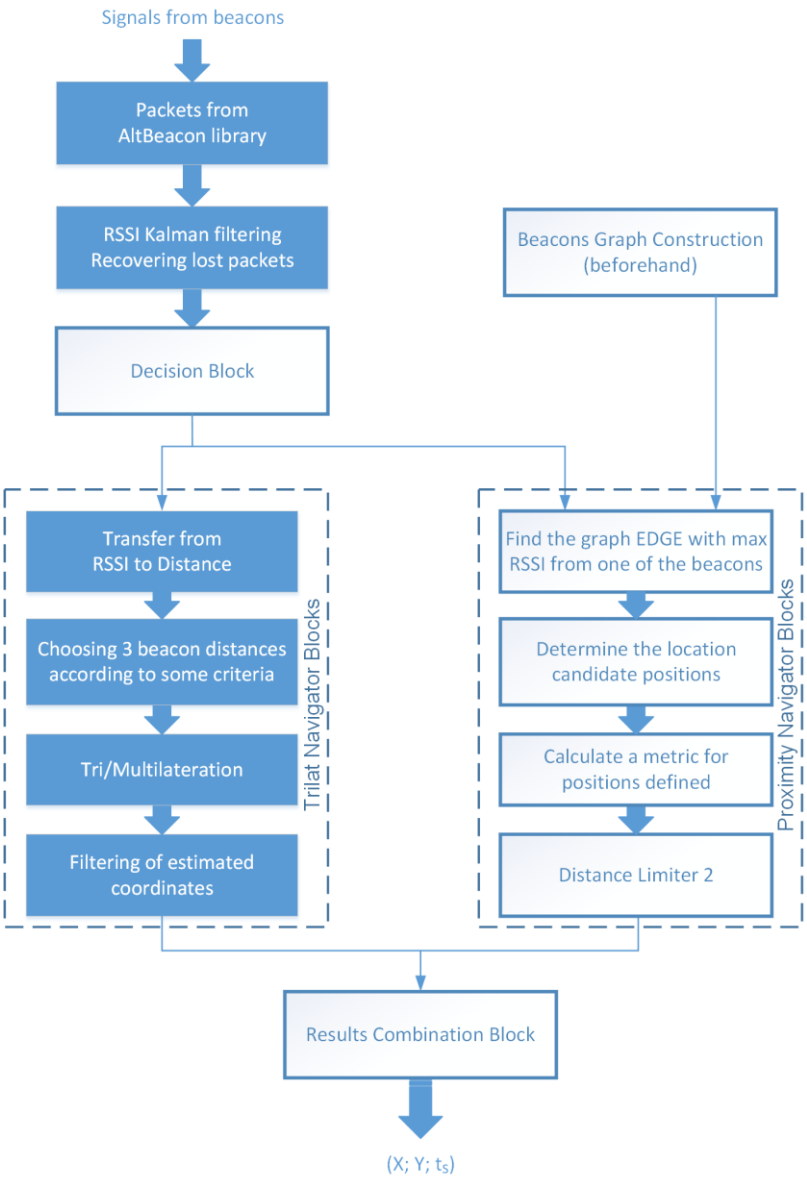

Рис. 3 – Структурная схема BLE части