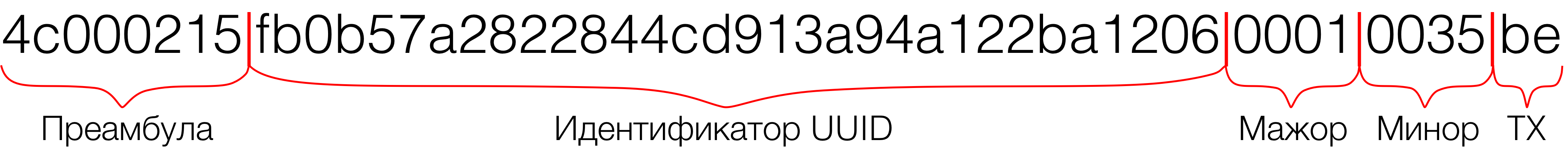
1.2. Форматы передачи данных

1.2.1. Формат данных iBeacon

iBeacon – технология, представленная Apple в iOS 7, которая расширила возможности библиотеки для работы со службами геолокации в iOS. Вместо геолокации по широте и долготе iBeacon использует низкоэнергетический сигнал Bluetooth, который обнаруживается телефоном. Формат данных строго типизирован и детально описан на официальном сайте Apple для разработчкиков.



*Рис 1. Формат данных iBeacon*

Преамбула – занимает 4 байта и является префиксным значением, определяющим, что это iBeacon маячок. Данный префикс всегда 4c000215.

Идентификатор UUID – на него отведено 16 байт, этот идентификатор не уникален и определяет конкретную группу маячков. Представим, нам необходимо установить iBeacon-маячки в главном корпусе ВГУ. Для решения данной задачи нам достаточно сгенерировать один UUID для всех маячков, таким образом, мы сможем подписываться на сообщения только от этих устройств и не будем получать другие, которые также могут быть установлены.

Мажор – на данную составляющую отведено 2 байта, это 2^16 = 65 536 различных значений. Мажор рекомендуется использовать для определения большой группы маячков, иденцифицируемой одним значением UUID. В примере с оборудованием главного корпуса ВГУ оптимально присвоить уникальное значение мажора каждому кабинету.

Минор – на него так же отведено 2 байта, этот параметр используется для однозначного определения маячка внутри группы.

TX – занимает 2 байта в конце протокола и обозначает эталонное значение мощности маячка (значение RSSI), измеренное на заводе-изготовителе на расстоянии 1 метра от маячка. Первый бит - это знак, 1 соответствует

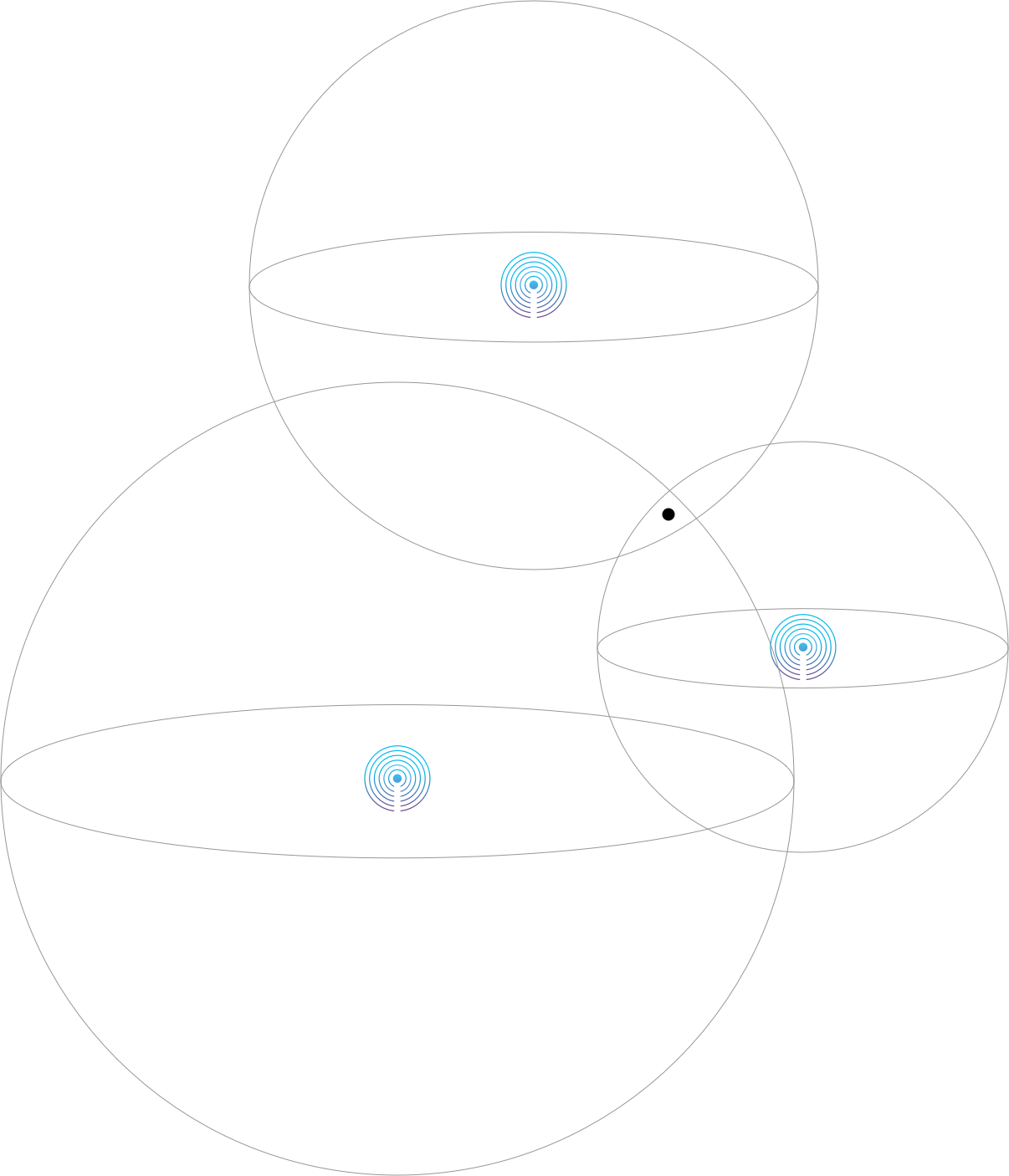
1.3. Алгоритмы определения местоположения.

1.2.1. Введение

Определение местоположения внутри помещений – фундаментальная проблема при программировании робототехники. Но в отличии от решений, применяемых в робототехнике, невозможно напрямую использовать встроенные сенсоры и команды для определения местоположения из-за того, что мобильные устройства приводятся в движение при помощи третьих лиц, в то время как в робототехнике возможно полагаться на команды движения самого робота. Сделав обзор применяемых технологий для определения местоположения, можно сделать вывод, что практически все они основаны на радиоволнах. Выбранная для решения поставленной задачи технология iBeacon так же использует радиоволны.

1.2.2. Алгоритм трилатерации

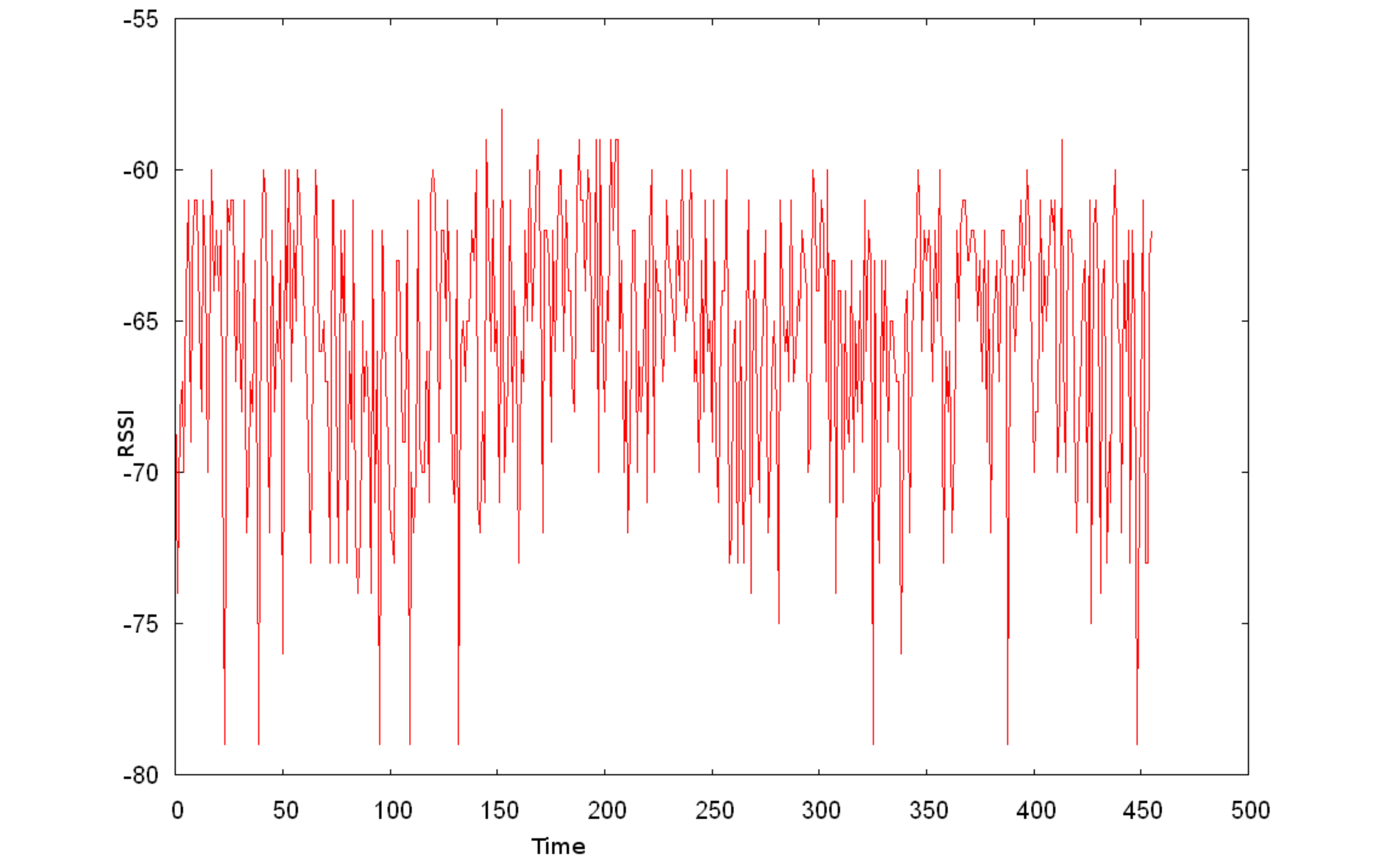
Трилатерация - это метод, позволяющий определить точку пересечения сигналов от передатчиков, путем построения в пространстве трех смежных треугольников, в которых известны длины их сторон. Этот метод основан на линейной засечке. Также часто применяются методы триангуляции и полигонометрии, но для этих методов необходимо обладать информацией об углах.



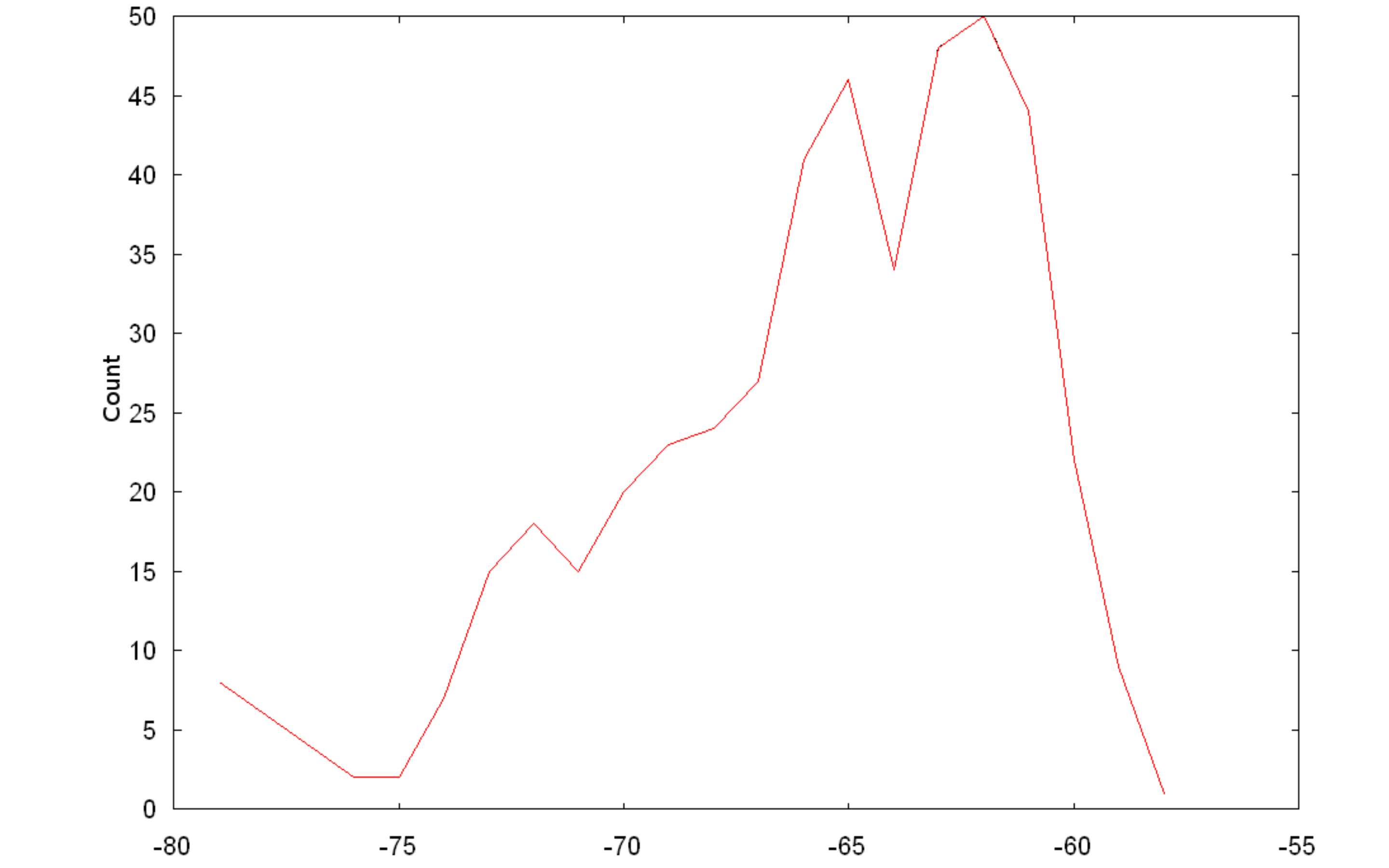
*Рис 3. Графическое представление задачи трехмерной трилатерации*

1.2.3. Практическая реализация алгоритма трилатерации

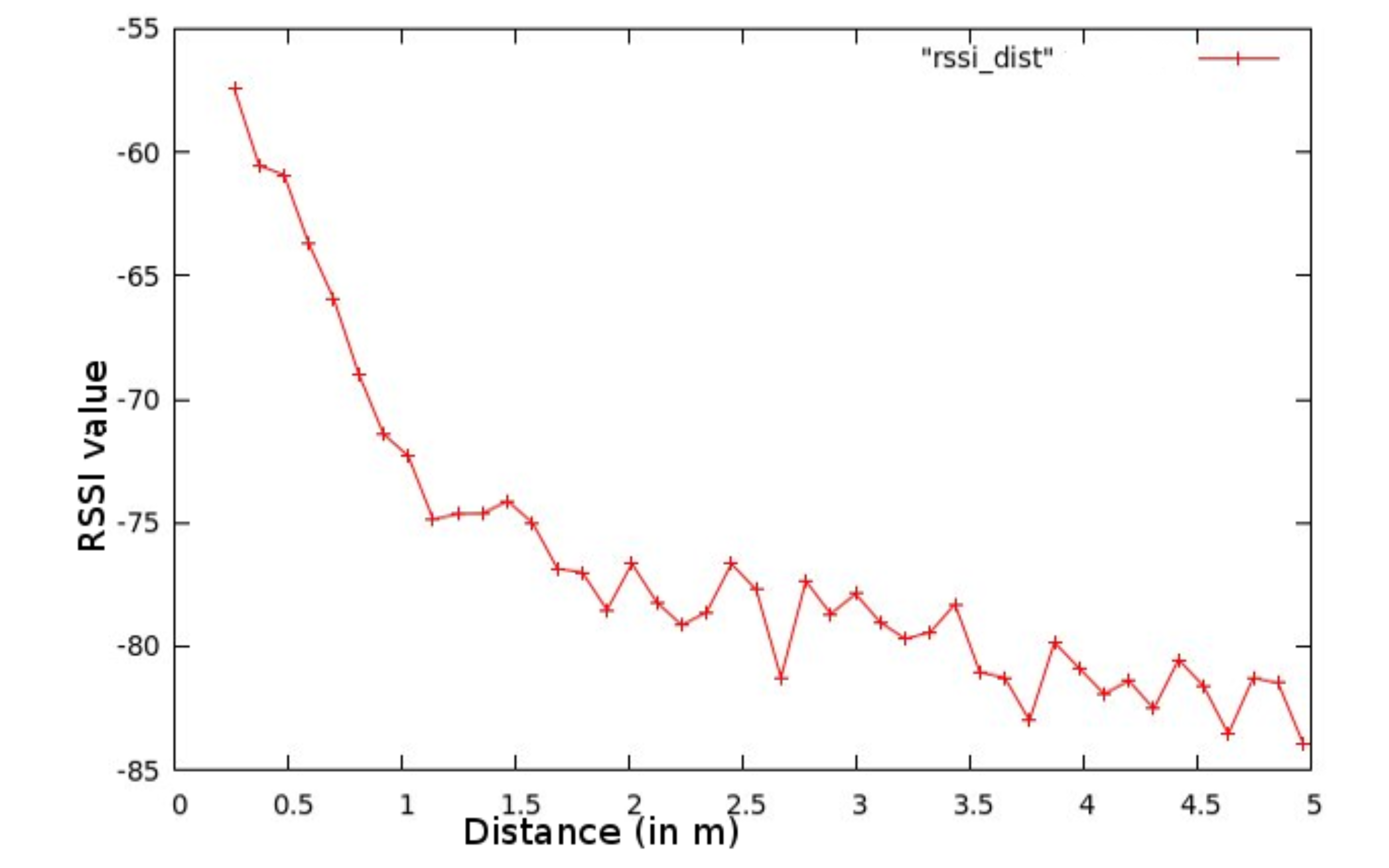
Получаемый от маячков параметр RSSI (Received Signal Strength Indicator) – параметр, позволяющий определить удаленность маячка от приемника сигнала. Это происходит следующим образом: маячок широковещает свой уникальный идентификатор, в хвосте которого находится параметр TX. Используя параметр, обозначающий эталонный RSSI на расстоянии 1 метр, производится деление текущего значения RSSI на эталонный. В теории мы должны получить точное расстояние в метрах между приемником – мобильным устройством и передатчиком (Bluetooth-маячком) с форматом передачи данных iBeacon. Однако из-за физического эффекта интерференции сигналов значение RSSI между стационарно-установленными приемником и передатчиком непостоянно, что можно увидеть на графиках.



*Рис 4. График изменения значения RSSI при статичном размещении приемника и передатчика*

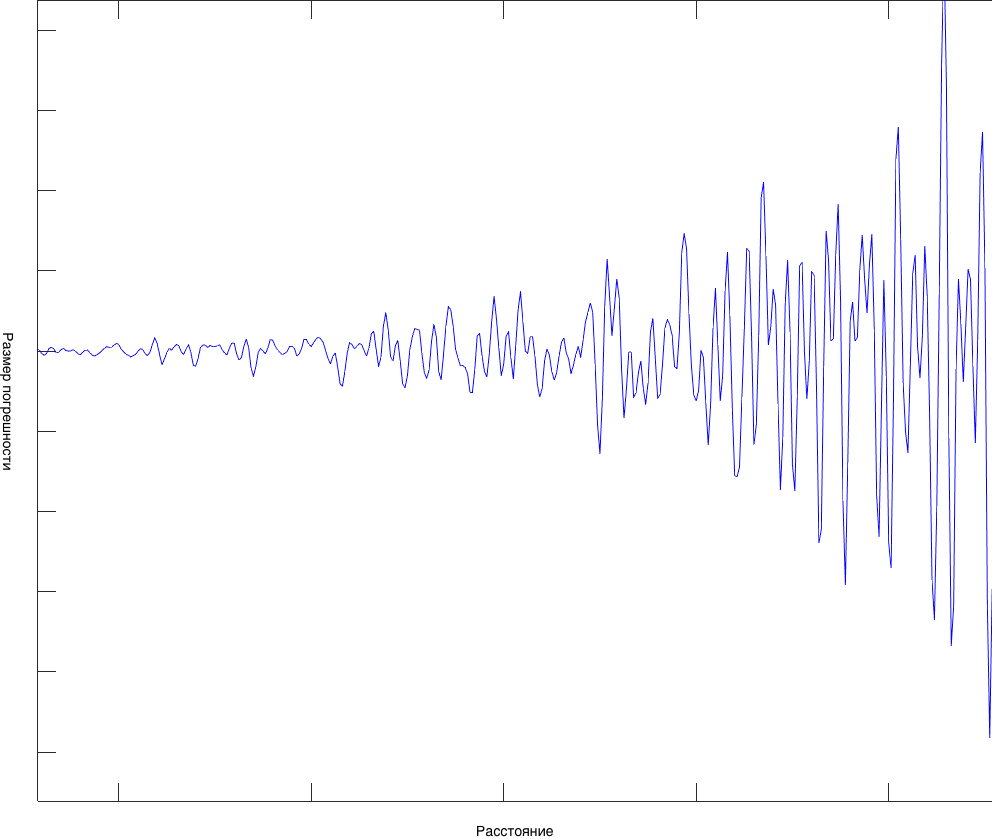


*Рис 5. График моды изменения значения RSSI при статичном размещении приемника и передатчика*



1.2.5. Практическая реализация алгоритма итеративной трилатерации.

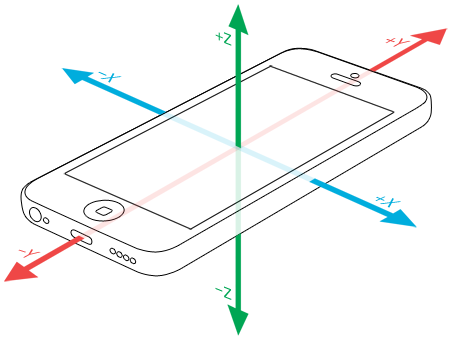
В результате практической реализации алгоритма итеративной трилатерации удалось решить проблемы отсутствия решений системы уравнений или же получения области решений уравнения. Для достижения целевой точности необходимо четко определить минимальное доверительное значение RSSI. Это необходимо сделать, исходя из технических характеристик маячков. Более простые решения, применяемые в дешевых маячках, имеют недостаточную силу сигнала. Точность определения расстояния обратно пропорциональна растоянию, то есть чем дальше мы находимся, тем меньше точность определения. Стоит отметить, что отношение значения погрешности к расстоянию выглядит очень похожим на экспоненциальную функцию.



2.3. Организация взаимодействия со встроенными сенсорами

Встроенная библиотека Core Motion в операционной системе iOS позволяет получать и обрабатывать события, зафиксированные широким набором сенсоров. На данный момент основной набор сенсоров для телефона — это акселерометр, гироскоп и магнетометр. Каждый из перечисленныйх датчиком имееет особенности в его отслеживании. В случае, если телефон оборудован сопроцессорами M7 или M8, обрабатывающими постоянно информацию с сенсоров, то можно использовать информацию об активности пользователя, такую как шаги, количество пройденных этажей и тип активности (прогулка, бег, поездка на велосипеде…). Логически можно объединить акселерометр и гироскоп в одну группу, оба этих сенсора представляют получаемую информациию в виде трехосевой системы координат, центр которой проходит через само устройство. Например, для iPhone в портретной ориентации:

ось Ox проходит через устройство слева направо, Oy снизу-вверх, а Oz проходит перпендикулярно сзади вперед:



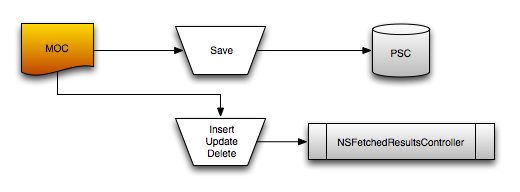
*Рис 8. Расположение системы координат в портретной ориентации*

Для работы со всеми сенсорами необходимо создать объект класса CMMotionManager. Этот класс предоставляет доступ ко всей информации о движениях мобильного устройства. Как и все API, предоставляющие персональную информацию, приложение получит доступ к информации и движениях только в

***2.4. Способы организации работы с CoreData***

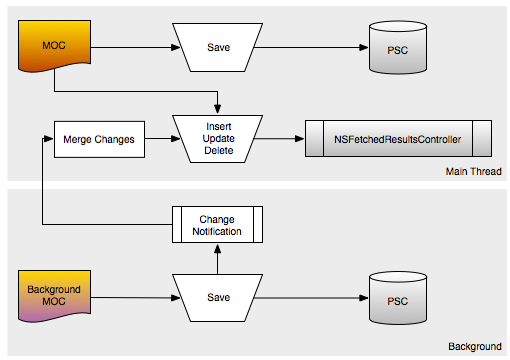
В данном проекте были рассмотрены несколько способов работы с CoreData, от самого простого, до самого быстродейственного:

1. Работа в главном потоке - самый простой способ, но относительно логики - неправильный, так как модель меняется до сохранения, кроме того, сохраняя в главном потоке, замораживается интерфейс пользователя на время сохранения



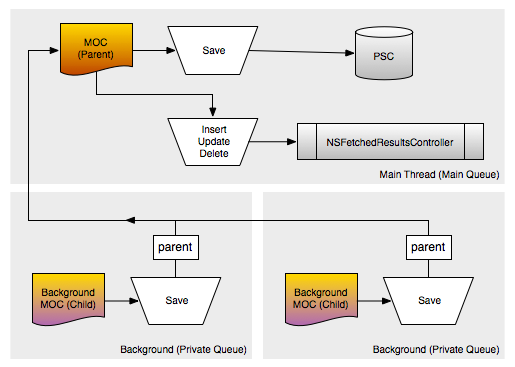
*Рис 9. Схема работы в главном потоке*

1. Работа в многопоточном режиме - паттерн, рекомендуемый apple в ее документации для параллельной работы с CoreData. Каждый поток имеет свой private контекст, каждый раз создает managed context в потоке, в котором он должен использоваться. Эта модель себя хорошо зарекомендовала, но в следующем обновлении iOS появилась более совершенная модель.



*Рис 10. Схема работы в многопоточном режиме*

1. Родительские контексты - Apple так же добавила возможность соединять в цепь managed object contexts вместе, устанавливая parentContext property для другого context. Таким образом, можно сделать изменения в Core Data и потом определиться, нужно ли их сохранить или откатить. При сохранении child context изменения автоматически распространяются на уровень выше, в представленном случае - на родительский контекст. Такой подход дает возможность обрабатывать добавление большого количество данных в фоновом потоке без блокировки главного потока(интерфейса).

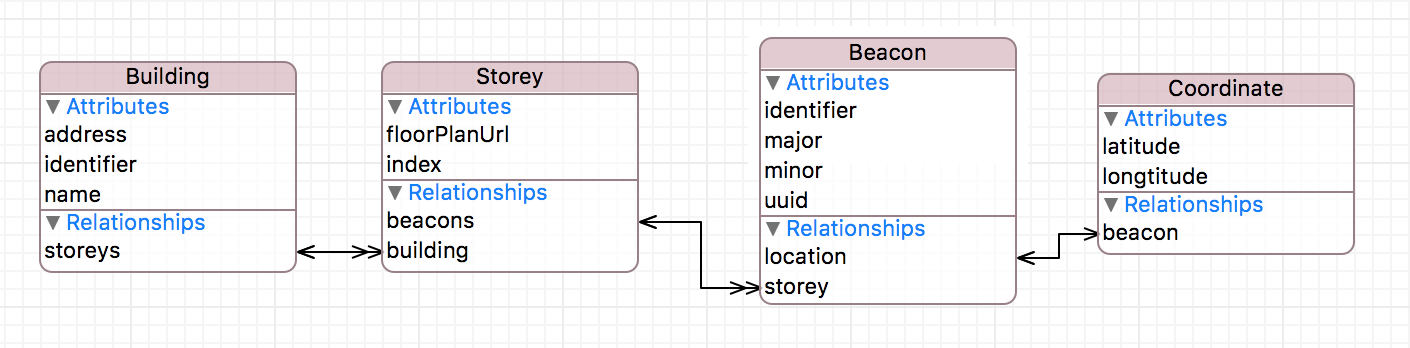


*Рис 11. Схема работы с родительскими контекстами*

Но любой fetch request приведет к операциям чтения или записи с диска в главном потоке, сохранению данных в фоновом потоке и потом сохранению в главном потоке, что все равно заблокирует главный поток. После долгих поисков был найден паттерн лучше. Этот паттерн используется Apple UIManagedDocument и было найдено много рекомендаций использования этого паттерна, даже в известной книге Marcus Zarra "CoreData Book".

***2.5. Схема базы данных***

Схема базы данных достаточно компактная, но это гарантирует всю нужную функциональность при максимальной производительности.



*Рис 13. Схема базы данных*

На рисунке 17 представлена схема локальной базы данных, основная сущность Beacon. Все атрибуты сущности являются обязательными, атрибут identifier является конкатенацией трех сторк uuid+major+minor, обеспечивающим уникальный идентификатор каждого маячка, этот атрибут используется для построения индекса в таблице. Маячок связан с вспомогательной сущностью Coordinate, это сделано для удобства работы с объектным представлением схемы. Сущность Storey – представляет информацию об этаже, а именно ссылку на схему этажа и индекс этажа. Связь этажа с маячками каскадная, следовательно при удалении этажа вызовется удаление всех маячков, которые с ним связаны. Сущность Building представляет собой здание, индекс построен по атрибуту identifier, который равняется идентификатору beacon, на который нужно подписаться для навигации в этом здании. Связь с этажами также каскадная, при удалении здания, будут удалены все этажи и все маячки каскадом.

ПРИЛОЖЕНИЕ 1. ЭКРАНЫ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

