МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВПО «ВГУ»)

Факультет прикладной математики, информатики и механики

Кафедра математического обеспечения ЭВМ

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ IBEACONS ДЛЯ ПОЗИЦИОНИРОВАНИЯ МОБИЛЬНЫХ ДЕВАЙСОВ ВНУТРИ ПОМЕЩЕНИЙ

Магистерская диссертация
По направлению 01.04.02 Прикладная математика и информатика
Магистерская программа Программирование для мобильных устройств

Допущен к защите ГЭК _____

Зав. кафедрой	_____ (подпись)	Махортов С.Д., проф., д.ф.-м.н.
Обучающийся	_____ (подпись)	Зонов А. В.
Руководитель	_____ (подпись)	Болотова С.Ю., преп., к.ф.-м.н.

Воронеж – 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Задача создания мобильного приложения, позволяющего позиционировать девайс пользователя внутри помещений очень актуальна. Существует множество примеров очень крупных зданий со сложной внутренней структурой, таких как аэропорты, торговые центры, университеты. В постройках такого типа ориентироваться могут лишь те, кто постоянно посущает их, а для человека, попавшего туда впервые, ориентирование в таких местах превращается в пытку. Кроме того, традиционные системы точного геопозиционирования не работают в таких ситуациях, где плотность застройки не позволяет использовать GPS спутники, так как при высотной плотной застройке GPS сигнал доходит до девайсов отраженный, ослабленный или зашумленный.

Актуальность данной задачи в том, что в современном мире люди привыкли использовать навигаторы для построения маршрутов, но как только человек попадает из открытой местности в здание, например, в аэропорт, ведение по маршруту заканчивается. Данным вопросом занимается все крупные информационные корпорации, такие как Apple, Google, Yandex, 2Gis и другие. Практически все современные картографические сервисы предлагают поэтажные карты крупных торговых центров и аэропортов, но определить точное местополежение пользователя они не могут.

В результате проведенного исследования не было найдено никаких библиотек или общепринятых практик для позиционирования внутри помещений в открытом доступе, хотя востребованность таковых велика, так как это очень большая маркетинговая ниша. В процессе исследования мной этого вопроса начали появляться первые приложения для конкрутных выставок, которые предлагали пользователям скачать приложение со схематичным расположением павильонов, но реализация данных решений была закрытой и являлась строго ориентированной под определенное мероприятие.

ПОСТАНОВКА ЗАДАЧИ

Основная задача данной работы – поиск и сравнение различных технологий определения местоположения внутри помещений, знакомство с основами разработки приложений под управлением операционной системы iOS, изучение и сравнение различных алгоритмов по определению местоположения. Так же ставится задача разработки приложения, позволяющего с помощью iPhone определить этаж на котором находится утройство, а так же местоположение устройства на этаже с точностью большей чем позволяют существующие решения, использующие GPS и GPRS позиционирования внутри помщений. Приложение должно обладать следующими возможностями:

- Определение в каком здании, на каком этаже и в каком месте находится устройство.
- Получать подробную схему здания.
- Отображать карту этажа, на котором находится устройство.
- Отображать текущее местоположение устройства на карте.
- Обновлять изменение местоположения в реальном времени и обновлять карту.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ОПРЕДЕЛЕНИЯ МЕСТОПОЛОЖЕНИЯ ПОЛЬЗОВАТЕЛЯ.

1.1. Основы работы существующих сервисов на примере функционирования GLONASS.

GLONASS - Глобальная навигационная спутниковая система, одна из двух функционирующих сегодня систем глобальной спутниковой навигации. Она состоит из 24 спутников, которые движутся в трех орбитальных плоскостях. Происходит это примерно на высоте 40 000 км. Все 24 спутника транслируют на землю несколько видов радиосигналов, а также общаются между собой и с наземной службой, за счем чего всегда знают свое местоположение. Сигнал, отправляемый каждым спутником, имеет строготипизированный фирмат, который содержит информацию о координатах самого спутника, а так же время отправки сигнала. Как только мобильное устройство получает сигнал, оно его расшифровывает, получает координаты и время отправки, после чего, используя константную величину скорости сигнала вычисляется расстояние. Но для определения местоположения нужно больше спутников, так как в спутниковом позиционировании используется трилатерация. Несколько спутников отправляют сигнал, результирующее расстояние расчитывается до каждого спутника, если отложить отрезок от каждого спутника до точки совместного соприкосновения, то получим координаты устройства, принявшего сигнал. Этот принцип позиционирования считается достаточно точным, но тем не менее имеет погрешность, которая может достигать на открытой местности 13 метров. В помещениях же погрешность на столько велика, что использование глобальных систем позиционирования оказывается невозможным.

1.1.1. Рассмотренные технологии позиционирования внутри помещений

Как было рассмотрено выше, глобальная навигационная система не позволяет добиться достаточной точности в рамках решения задачи определения местоположения внутри помещений. Так как в постановке задачи было определено что приложение должно работать на смартфонах под управлением опрерационной системы iOS это не наложило ограниения, но так же предоставило мне широкий набор датчиков и сенсоров, которые можно использовать для решения нашей задачи.

1.1.1.1. Навигация, используя Wi-Fi

Wi-Fi (Wireless Fidelity) – Беспроводные сети на базе стандарта IEEE 802.11. На данный момент уже существуют систмы позиционирования с использованием Wi-Fi, обычно это вспомогательные системы для GPS, так как для точного позиционирования по Wi-Fi, необходимо достаточно много точек в зоне видимости без препятствий между трансмитером и приемником. Такие свойства позиционирования обусловлены способом определения местоположения, так как используется RSSI (received signal strenght indicator) и метод «Fingerprinting». Для построения необходимо иметь базу данных соответствий отпечатков кажной Wi-Fi точки и ее координаты, на основе этих данных, используя силу сигнала можно определить расстояние до устройства, получающего сигнал и по аналогии с GPS, используя трилатерацию и как минимум 3 Wi-Fi точки определить координаты устройства. При реализации данного типа навигации на базе существубщей инфраструктуры былы выявлены следующие проблемы:

1. Недостаточное пересечение областей покрытия Wi-Fi - очень малое количество мест общественного пользования оснащены достаточным количеством Wi-Fi роутеров, даже если покрытие аэропорта достигало 100%, в каждом конкретном месте было в основном 1-2 видимых точки, что не позволяет однозначно определить метоположение пользователя (пересочение 2 окружностей может дать 2 точки пересечения).
2. Недействительное соответствие значения RSSI и расстояния – из-за частоты работы в диапазоне между 2400-2500 MHz, сигнал не обладает достаточной

силой и перекрывается многими другими устройствами, так же работающими в этом диапазоне, из-за чего за частую даже при появлении 3 роутеров в зоне видимости RSSI был слошком мал и пересечения 3 окружностей с центрами в координатах роутера и и радиусом равным посчитанному расстоянию на основе RSSI не было. Погрешность вычисления в таком случае слишком велика.

Из-за этих особенностей была невозможна реализация проекта согласно поставленной задаче, так как погрешность вычисления слишком велика. Тем не менее, этот способ позиционирования имеет свои преимущества и недостатки.

Преимущества:

1. Возможность использовать существующую инфраструктуру. При постановке задачи определения комнаты в которой находится устройство или любой другой задачи позиционирования устройств внутри помещений, не требующих высокой точности определения местоположения этот способ является самым дешевым при наличии покрытия Wi-Fi сетью.
2. Низкое энергопотребление. При использовании Wi-Fi позиционирования вместо GPS значительно уменьшается энергопотребление.
3. Достаточно большой радиус действия. Wi-Fi сигнал способен распространяться вплоть до 150 метров.
4. Возможность определить этаж. За счет уникального отпечатка каждого роутера, есть возможность хранить в базе соответствие отпечатков и этажей и определяя самый сильный сигнал возможно определить, на каком этаже находится устройство.

Недостатки:

1. Низкая точность. Точность позиционирования даже при специально подготовленной инфраструктуре будет не менее 15 метров.
2. Высокая цена улучшения инфраструктуры. Цена установки новой точки будет складываться из стоимости роутера, витой пары нужной длинны, а так же подведения питания и работ по установке.

1.1.1.2. Геомагнитное позиционирование

Геомагнитное позиционирование основано на магнитном поле земли, а именно на аномалиях в магнитном поле, которые и используются в этом методе позиционирования. Погрешность при использовании данного подхода примерно равна 2 метрам. Перед осуществлением навигации, необходимо подготовить карту геомагнитных аномалий на карте, а также перед каждым сеансом необходимо произвести корректировку магнетометра, после чего позиционирование будет готово к работе.

При реализации текущего подхода, сразу были выявлены недостатки, магнитное поле в крупных зданиях не такое постоянно и современная архитектура, и обилие электронных устройств создает множество динамических аномалий, из-за которых построение постоянной карты здания становится невозможным. Таким образом, эмпирическим путем было выявлено несоответствие заявленой погрешности и реальной.

К тому же, из-за того, что крупные здания обвиты проводкой, поле в которой меняется в зависимости подключенной нагрузке, сильно меняя конфигурациюмагнитного поля вокруг себя, карта аномалий может меняться на протяжении дня, в зависимости от загруженности.

Все это явилось причиной отказа от этого подхода. Тем не менее, данный подход широко используется в местах без доступа GPS сигнала и статичным полем аномалий, например, на подземных стоянках и складах, где расстояние до стен достаточно велико, а количество носимых устройств недостаточно велико, чтобы динамически влиять на магнитное поле. Финский университет Уолу представил прототип описаной системы для использования в мобильных устройствах внутри помещений.

1.1.1.3. Ориентирование по базовым станциям операторов сотовой связи.

GSM ориентирование основано на триангуляции сигнала сотовых вышек. Каждая базовая станция регулярно транслирует сигнал в открытый эфир, чтобы телефоны могли понимать, находятся ли они в зоне покрытия. Обычно, в городах телефоны находятся в зоне покрытий сразу нескольких базовых станций и телефон

выбирает базовую станцию своего оператора с наилучшим сигналом. Сам же телефон с низкой периодичностью сообщает сети о том какую базовую станцию он «слышит» лучше всего, для упрощения доставки входящих звонков, обычно, когда сигнал с последней переданной станции становится слишком слабым. Это сделано для экономии заряда мобильных девайсов. Таким образом процесс определения местоположения в данном случае должен осуществляться на мобильном устройстве. Этот подход подходит для решения поставленой задачи. Для определения своего местоположения достаточно знать координаты видимых базовых станций, который можно взять из открытых источников и используя силу сигнала вычислить расстояние между абонентом и базовой станцией. В случае одной видимой базовой станции мы получаем окружность с центром в точке расположения станции и радиусом равным расстоянию между устройством и станцией, такая погрешность может доходить до 32 км, что является не приемлемым для нашей задачи. Рассматривая лучший случай, когда мы имеем 2 и более базовых станций, сила сигнала которых настроена на минимальные значения, мы получаем 2 и более окружностей, которые могут иметь или не иметь область пересечения. Такое поведение обусловлено физическими свойствами сигналов и большой погрешностью на закрытых площадях. В данном случае мы можем сократить погрешность до 100 метров. Таким образом можно сделать вывод, что данный подход нельзя использовать для решения поставленой задачи. Однако данный подход имеет свои преимущества и недостатки:

Преимущества:

1. Хорошее покрытие купных городов базовыми GSM станциями.
2. Возможность использовать базовые станции всех операторов связи.
3. Низкое энергопотребление.

Недостатки:

1. Большая погрешность вычислений.
2. Сложность улучшения инфраструктуры.
3. Невозможность определить высоту

1.1.1.4. Технология iBeacon

Определение местоположения при помощи bluetooth iBeacon маячков выгладит следующим образом: по всему периметру расставлены Bluetooth метки, которые придерживаясь протокола вещения для iBeacon производят широковещательную рассылку, которая, согласно типизации iBeacon протокола, содержит идентифицирующую их информацию. Мобильное устройство принимает сигналы от всех маячков, сигнал от которых до них доходит. Используя полученные данные, мобильное устройство может сопоставить уникальные идентификаторы устройств с координатами. Которые были сохранены заранее. Для определния расстояния от устройства до маячка используется параметр RSSI.

RSSI (Received Signal Strength Indicator) – параметр силы сигнала, определенный в спецификафии Bluetooth. Этот параметр вычисляется пользовательским Bluetooth приемником и обозначает силу принимаемого сигнала. Чем выше этот параметр, тем ближе мы находимся к маячку. Для того чтобы определить расстояние в эмперических величинах, например, в метрах, нам понадобится другой параметр – MP, который описан на уровне iBeacon.

MP (Measured Power) – уровень сигнала в 1 метре от передатчика. Данное значение передается вместе с остальной частью уникального идентификатора устройства. Данный параметр задается каждому маячку на заводе и у маячков одного производителя и одной модели этот показатель может различаться.

Физически каждый Beacon маячек выглядит как небольшая плата с Bluetooth 4.0 LE (Low Energy). Протокол iBeacon – стандарт, представленный Apple формата широковещательного сообщения. Таким образом, для тестоввозможно использовать любое устройство, оснащенное Bluetooth 4.0 и выше. Для практической реализации достаточно купить недорогие Beacon маячки, которые состоят из батарейки, микроконтроллера, отвечающего за широковещание и периферийного модуля Bluetooth 4.0 LE.

Данная технология полностью удовлетворяет критериям поставленой задачи. Прототип системы, описаный с использованием данного подхода показал достойные

результаты. iBeacon хорошо подходит для нашей задачи, но также, как и другие технологии имеет свои преимущества и недостатки.

Преимущества:

1. Низкая стоимость оборудования.
2. Не требует подключения к сетям питания или интернету.
3. Работоспособность до 3-х лет от одной батарейки.
4. Возможность прототипизирования системы с использованием мобильных телефонов, без покупки Beacon.
5. Небольшая пограшность вычислений, до 5 метров.
6. Легкое расширение инфраструктуры.
7. Определение этажей.
8. Низкое электропотребление пользовательского устройства.

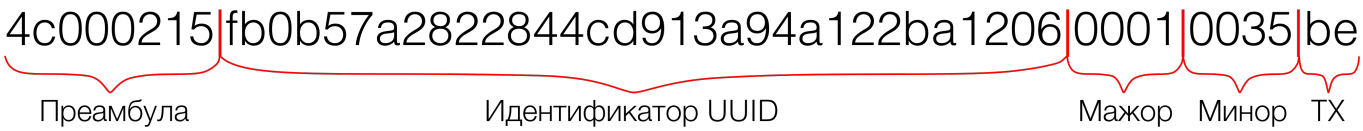
Минусы:

1. Необходимость контроля за состоянием элементов питания во всех маячках.
2. Слабый сигнал, который блокируется перекрытиями.
3. Необходимость вручную развешивать маячки и сохранять их координаты и уникальный идентификатор в базе.
4. Отражение радиосигнала от поверхностей.
5. Зависимость от направленности излучения маячка и пользовательского устройства.

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.



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

Идентификатор UUID – на него отведено 16 байт, этот идентификатор не уникален и определяет какую-то группу маячков. Представим, нам необходимо установить Beacon 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 так же могут быть установлены.

Мажор – на нее отведено 2 байта, это 2^16 = 65 536 различных значений. Мажор рекомендуется использорвать 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 (его значение RSSI) измеренное на заводе изготовителе на расстоянии 1 метра от маячка. Первый бит это знак, 1 соответствует отрицательный знак, а 0 - положительный. Именно благодаря этой константе мы можем использовать метрические координаты в пространстве. В схеме представленой на рисунке, значение TX равняется 0xBE, что соответствует

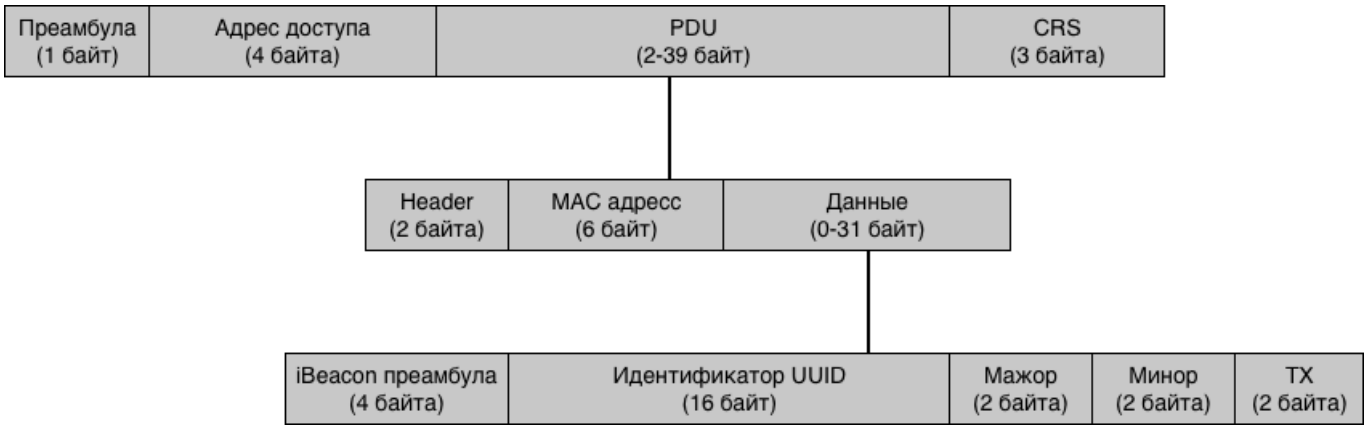
числу 190 в десятичной системе счисления. Таким образом, эталонный RSSI на расстоянии 1 метра можно вычислить как 256-190 = -66 dBm.

Таким образом получая Связку UUID+мажор+минор можно однозначно определить от какого маячка был получен сигнал, после чего сделать выборку из базы координаты маячка с таким значением.

Так же для решения задачи Можно использовать несколько UUID, например назначить уникальный UUID для каждого этажа. Используя данный подход можно будет однозначно определить какую часть карты стоит открыть не делая запрос в базу.

1.2.1. Формат данных Bluetoth

iBeacon формат является одним из стандартов данных, которые могут быть записаны в свободную область данных формата Bluetooth. Любое мобильное устройство оснащенное Bluetooth модулем может считать информацию о Beacon маячках. Формат Bluetooth пакета имеет следующую форму.



Как видно из рисунка, iBeacon одно из заполнений области данных, слоя PDU в формате сообщения Bluetooth. Благодаря такому свободному формату, существуют аналоги формата iBeacon, например формат Eddisone, разработанный в google. Преимущество формата iBeacon поддержка большим количеством платформ, чем другие форматы. iBeacon формат не занимает всю область данных, отведенную в PDU, то есть при необходимости можно расширить формат и использовать оставшиеся 5 байт в своих целях.

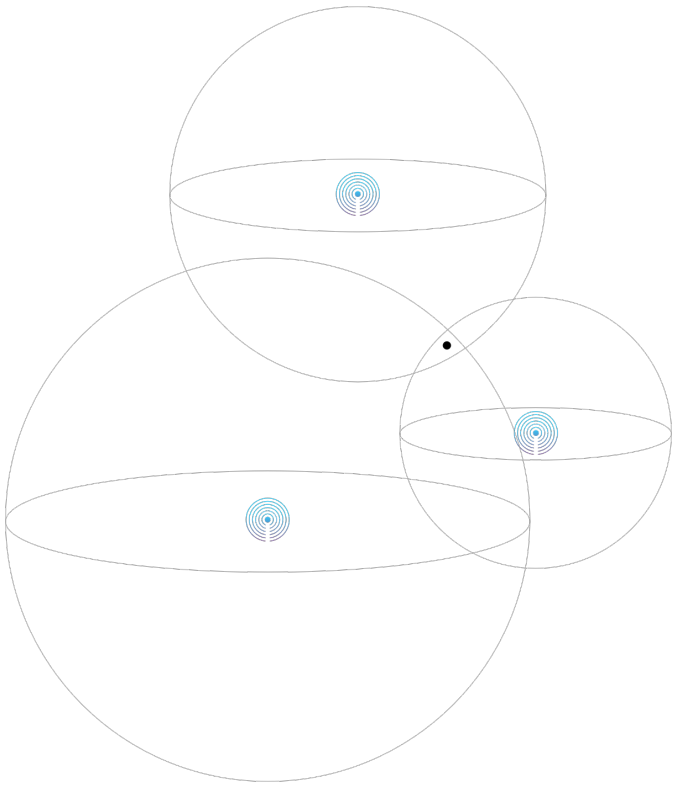
1.3. Алгоритмы определения местоположения.

1.2.1. Введение

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

1.2.2. Алгоритм трилатерации

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



Задача трехмерной трилатерации решается при помощи нахождения координат пересечения трех сфер, которые определяются при помощи решения системы уравнений. Для начала напишим систему уравнений для трех сфер, составленую следующим образом, так как любые 3 точки в пространстве образуют плоскость, поместим начало координат этой плоскости в центр одной из сфер, центр второй сферы поместим на ось координат Ox:

$$r_1^2 = x^2 + y^2 + z^2$$

$$r_2^2 = (x - d)^2 + y^2 + z^2$$

$$r_3^2 = (x - i)^2 + (y - j)^2 + z^2$$

В точке (x, y, z) будет располагаться приемник. Для решения системы, вычтем второе уравнение из первого, благодаря чему найдем x:

$$x = \frac{r_1^2 - r_2^2 + d^2}{2d}$$

Предположим, что две сферы имеют более одной точки пересечения, тогда имеет смысл следующее соответствие:

$$d - r_1 < r_2 < d + r_1.$$

В этом случае, после подстановки x в уравнение первой сферы, находязейся в начале координат, получим уравнение окружности, которое является фигурой, пулучившейся в результате пересечения двух сфер.

$$y^2 + z^2 = r_1^2 - \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x$$

Подставим в уравнение третьей серы равенство:

$$y^2 + z^2 = r_1^2 - x^2$$

И найдем значение y:

$$y = \frac{r_1^2 - r_3^2 - x^2 + (x - i)^2 + j^2}{2j} = \frac{r_1^2 - r_3^2 + i^2 + j^2}{2j} - \frac{i}{j}x$$

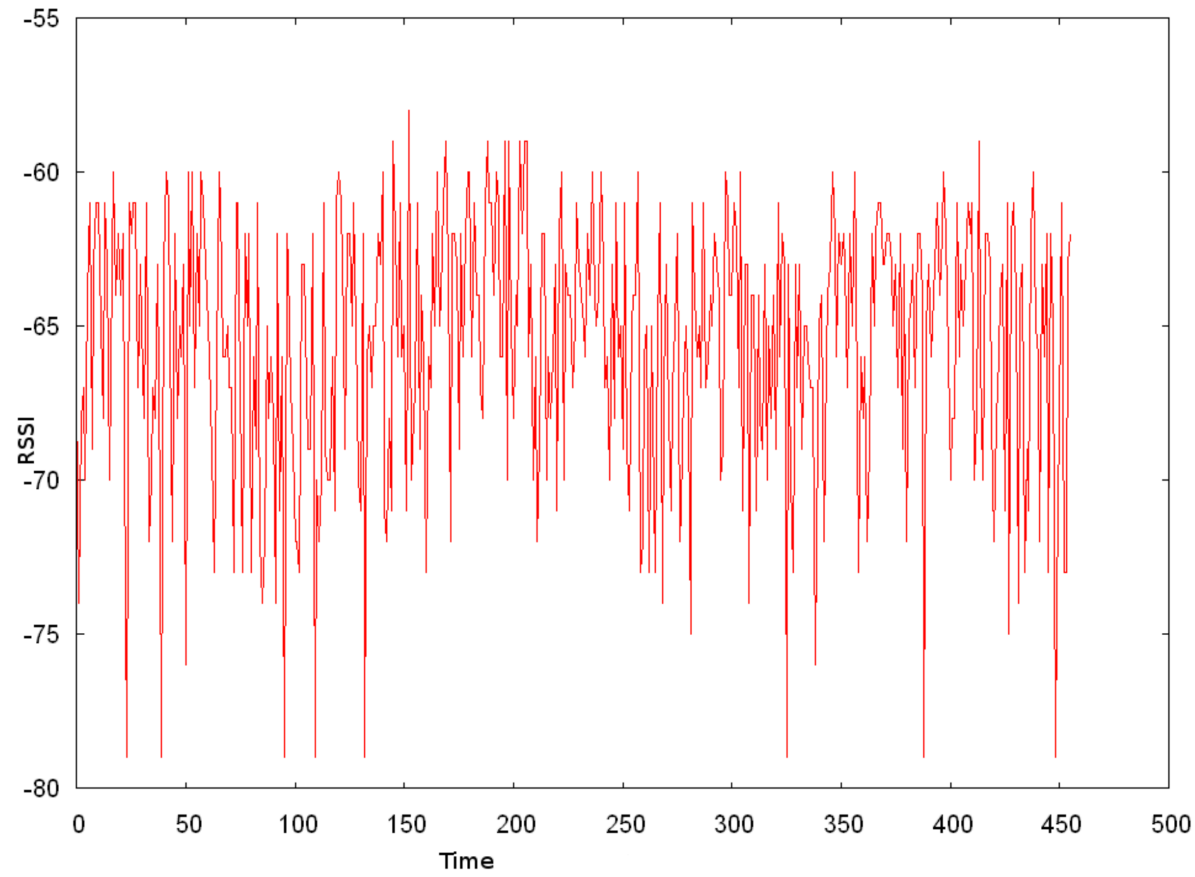
Зная координаты x и y, мы можем найти координату z:

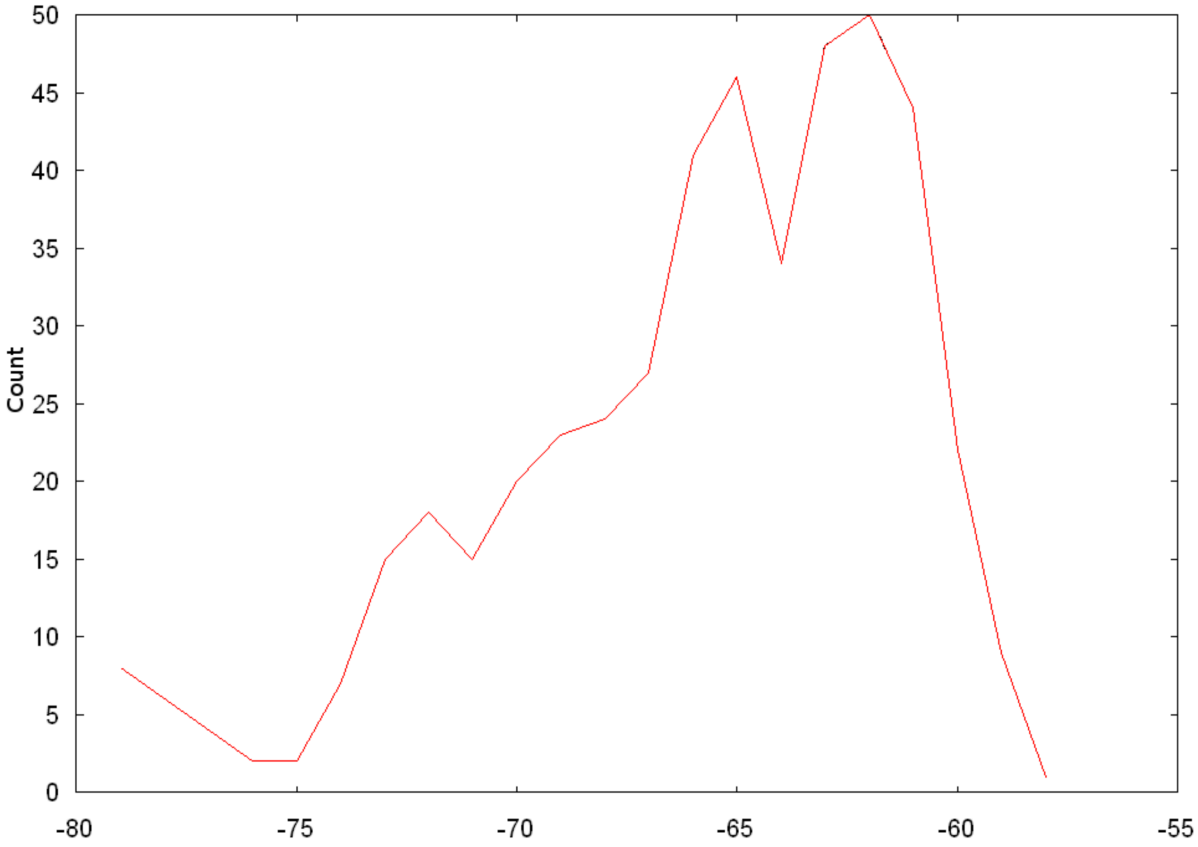
$$z = \pm\sqrt{r_1^2 - x^2 - y^2}$$

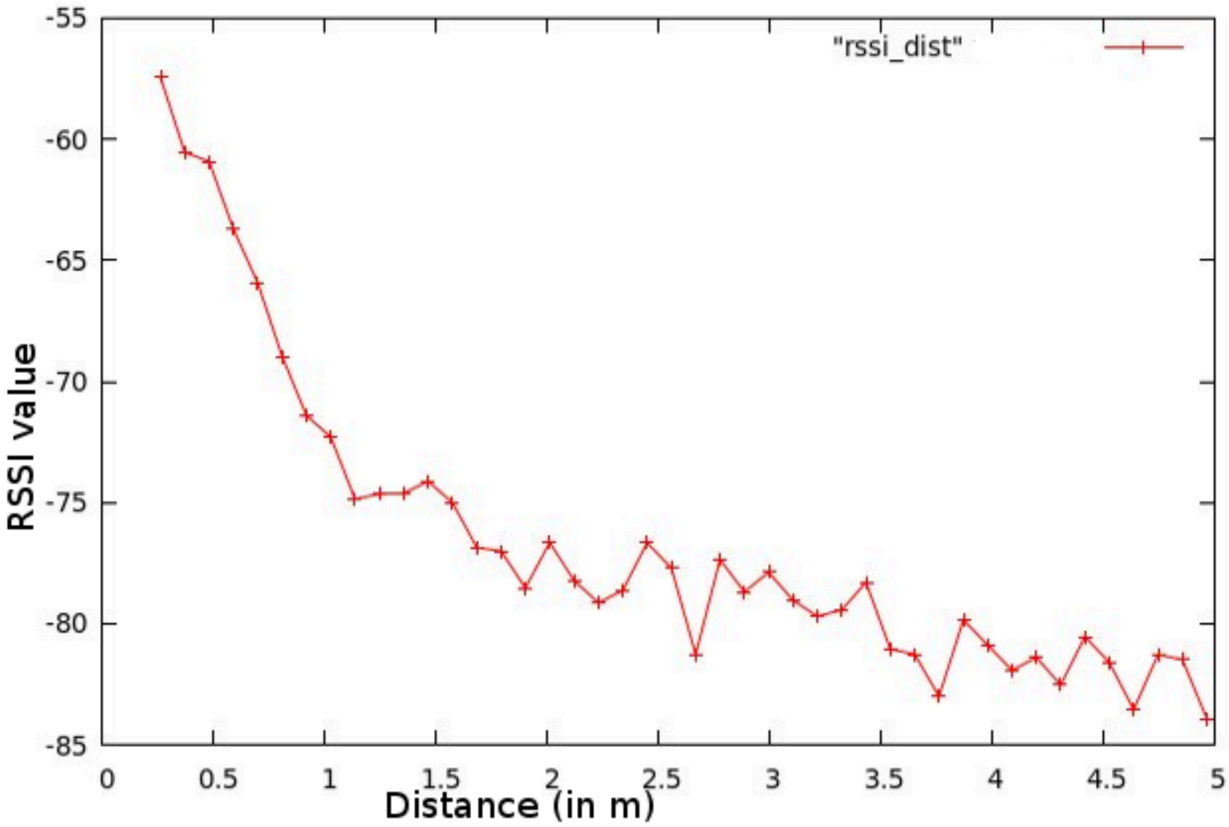
Таким образом, мы вывели уравнение для координаты приемника, используя информацию, полученную от Bluetooth передатчиков. Как видно, координата высоты может принимать 2 значения. Это убусловлено тем что передатчики и приемник могут находиться не в одной плоскости, но для нашей задачи мы можем рассматривать только нижнее значение, так как маячки расположены выше пользовательских устройств.

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 следующим образом: маячек 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.


-55
-60
-65
-70
-75
-80
RSSI
0
50
100
150
200
250
300
350
400
450
500
Time


50
45
40
35
30
25
20
15
10
5
0
Count
-80
-75
-70
-65
-60
-55



Как видно из представленных графиков, при использовании одного маячка можно получать разную удаленность при статичном расположении приемника и передатчика. Таким образом задача трилатерации в чистом виде не позволяет определить местоположение устройства. В данных условиях мы можем получить в пересечении трех сфер область, либо не получить точек пересечения сфер вообще. Для решения данной задачи необходимо было модифицировать алгоритм трилатерации, используя итеративный подход.

1.2.4. Алгоритм итеративной трилатерации.

Обозначим начальные точки и полученное расстояние как (*xi, yi)* и *di,* соответственно. Полученные при помощи тривиального алгоритма начальные координаты обозначим (*xe*, *ye)*.

Разница или ошибка рассчитывается по формуле:

$$|f_i| = |d_i - \sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}|$$

Теперь, применяя аппроксимацию при помощи ряда Тейлора первой степени, можно

найти $(\Delta x,\Delta y)$, используя следующие равенства:

$$\Delta= (B^T B)^{-1} B^T f$$

$$\Delta= \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix}$$

Где B вычисляется следующим образом

$$B = \begin{pmatrix} \frac{\partial f_1}{\partial x_e} & \frac{\partial f_1}{\partial y_e} \\ \frac{\partial f_2}{\partial x_e} & \frac{\partial f_i}{\partial y_e} \\ \vdots & \vdots \\ \frac{\partial f_i}{\partial x_e} & \frac{\partial f_i}{\partial y_e} \end{pmatrix} == \begin{pmatrix} \frac{(x_1 - x_e)}{\sqrt{(x_1 - x_e)^2 + (y_1 - y_e)^2}} & \frac{(y_1 - y_e)}{\sqrt{(x_1 - x_e)^2 + (y_1 - y_e)^2}} \\ \vdots & \vdots \\ \frac{(x_i - x_e)}{\sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}} & \frac{(y_i - y_e)}{\sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}} \end{pmatrix}$$

В результате получим:

$x_e = x_e + 0.05\Delta x$

$y_e = y_e + 0.05\Delta x$

Учитывая это, fi может быть рассчитан как:

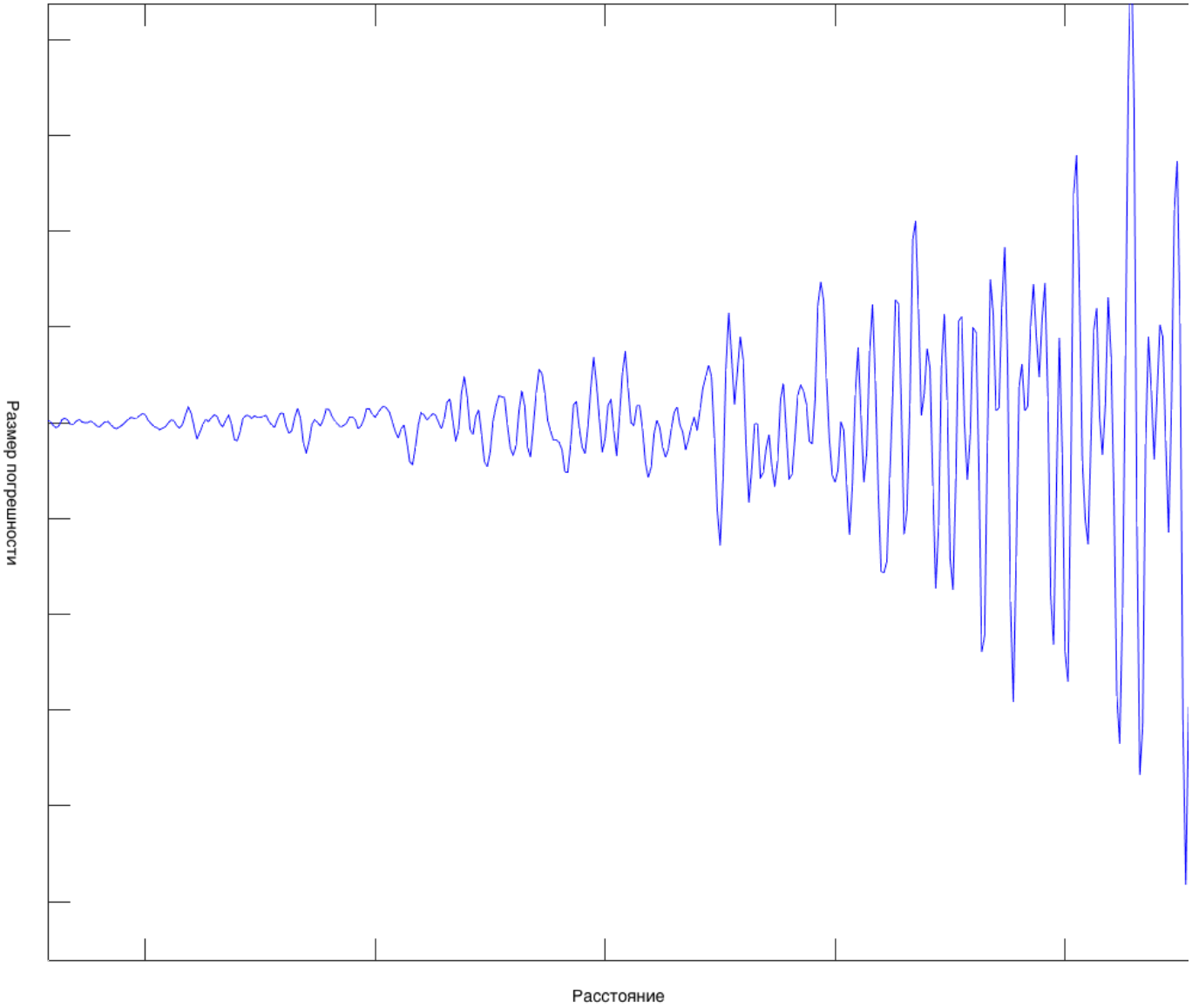
$$|f_i| = \left| \frac{d_i - \sqrt{(x_i - x_e)^2 + (y_i - y_e)^2}}{d_i} \right|$$

Таким образом получаем дробное знаение, характеризующее ошибку. Итеративно необходимо повторять данные расчеты до тех пор, пора ошибка не уменьшится до заданной ∂.

1.2.5. Практическая реализация алгоритма итеративной трилатерации.

В результате практичееской реализации алгоритма итеративной трилатерации, удалось решить проблемы отсутствия решений системы уравнений или же получения области решений уравнения. Для достижения необходимой точности,

необходимо четко определить минимальное значение RSSI, значению которого можно доверять. Это необходимо сделать исходя из технических характеристик маячков, более дешевые решения, применяемые в дешевых маячках, имеют не достаточную силу сигнала, и точность определения расстояния обратно пропорциональна растоянию, то есть чем дальне мы находимся, тем меньше точность определения. Стоит заметить, что отношение значения погрешности к расстоянию выглядит очень похожим на экспоненциальную функцию.



То есть зная допустимую погрешность, можно определить минимальный уровень доверия маячку. Так же стоит учитывать маячки должны быть расположены таким образом, чтобы в любой точке помещения находилось какк минимум 3 маячка с уровнем RSSI выше порогового допустимого значения для точности. При наличии

же физических препятствий или отражателей сигналов, таких как столбы, шкафы, зеркала необходимо размещать больше 3-х маячков с достоточной силой сигнала. Этот подход позволяет получить несколько значений и расчитать их среднее, для получения всех возможных сочетаний легко посчитать по формуле:

$$\frac{n}{k} = C_n^k = \frac{n!}{k! * (n-k)!}$$

Где k=3 - элементы, необходимые для трилатерации местоположения, а n – различные маячки, где n ≤k. Таким образом, увеличив количество видимых маячков с достаточной силой сигнала до 4, мы получим 4 координаты, усреднив значения которых, мы получим более точное значение. При 5 маячках, количество координат увеличивается до 10. Такое количество маячков позволяет определить местоположение с достаточной точностью даже в самых сложных условиях.

Ипользуя описанные решения, на прототипе вычисляется местоположение пользователя, но из-за оставшегося «шума», который проявляется в виде изменения координаты пользователя на дистанцию до 5 метров, при отсутствии физического движения. Для фильтрации «шума» недосточно только информации от маячков. Понять находиться ли устройство в покое или в движении можно при помощи встроенных в устройство сенсоров, магнетометра, гироскопа и акселерометра. Для того чтобы учитывать внешние атрибуты, подходят методы на основе рекурсивной оценки Байеса.

1.2.5. Методы на основе оценки Байеса.

Для интеграции данных о передвижении устройства, получаемых от сторонних сенсоров, были рассмотрены и реализованы многочастотный фильтр и фильтр Калмана. Фильтр Калмана предназначен для рекурсивного дооценивания вектора состояния априорно известной динамической системы, другими словами, для расчета текущего состояния системы, необходимо знать текущее состояние системы, а также предыдущее состояние самого фильтра. Алгоритм ориентирован на двуэтапную работу:

1 Этап прогнозирования. На первом этапе, фильтр Калмана экстаполирует значения переменных состояния, а также их неопределенности.
2 Этап уточнения. На втором этапе, по данным измерения, полученного с некоторой погрешностью, результат экстраполяции уточняется.

Благодаря пошаговой природе алгоритма, он может в реальном времени отслуживать состояние объекта.

Для интегции фильтра Калмана в нашу систему необходимо определить динамическую составляющую системы последовательных прилатераций. Управляющим воздействием будет информация о векторе движения, получаемая от надора датчиков, и множество последовательных измерений местоположения методм прилатерации маячков для формирования реальной оценки состояния.

Координату мобильного устройства можно описать законом:

$$x_{k+1} = x_k + u_k dt \qquad (1)$$

Из-за невозможности опираться только на сенсоры, необходимо добавить случайную величуну ξ_k.

$$x_{k+1} = x_k + u_k dt + \xi_k \qquad (2)$$

У нас есть алгоритм получения координат от внешних маячков, для использования его с фильтром Калмана добавим к этим координатам ошибку распространения навигационного сигнала, обозначенную за η_k, которая будет являться случайной величиной. В результате получим ошибочные данные вычисления координаты:

$$z_k = x_k + \eta_k \qquad (3)$$

Итоговая задача фильтрации будет состоять в том, чтобы используя неверные показания сенсора z_k, добиться достаточно точного приближения к x_{opt}. В результате, получаем уравнение системы для мобильного девайса с использованием набора сенсоров и вычислением позиции по маячкам будет иметь следующий вид:

$$\begin{cases} x_{k+1} = x_k + u_k dt + \xi_k \\ z_k = x_k + \eta_k \end{cases} \quad (4)$$

Где u_k – известная временная, контролирующая изменение системы, она определяется из вектора движения, полученного от датчиков.

ξ_k и η_k – ошибки изменения вектора системы и датчиков, соответственно.

Если для k-го шага известно значение сенсора x_k^{opt}, отфильтрованое таким образом, что приближает истинную координату x_{k+1} достаточно, то на следующем шаге $k+1$ становится известно неточное показание сенсора z_{k+1}.

Задача интеграции фильтра Калмана для использования с определением местоположения при помощи iBeacon внешних маячком и набора внутринних сенсоров состоит в получении наилучшего приближения к истинной кооординате x_{k+1}, из ошибочных показаний сенсора z_{k+1} и предсказаний истинного значения - $x_k^{opt} + u_k$. Для этого необходимо добавить коэффициент, определяющий вес сенсора K, а предсказанному значению (1-K). В результате чего будет получена следующая формула:

$$x_{k+1}^{opt} = K * z_{k+1} + (1 - K) * (x_k^{opt} + u_k) \quad (5)$$

Где K – коэффициент Калмана шага.

Коэффициент Калмана должен быть выбран так, чтобы получившееся оптимальное значение координаты x_k^{opt} было наиболее близко к истинной координате x_{k+1}. Представим что нам известно, что в данном здании маячки установлены свысокой плотностью и количество препятствий, ослабляющих сигнал миниматьно, в таком случае определние координат при помощи iBeacon маячков будет обладать высокой точностью. Для того чтобы доверять координатам результата трилатерации маячков, значению коэффициента K устанавливается больший вес. Если же мы понимаем что в области определения местоположение сигнал маячков слаб или количество маячков не большое, либо же вообще не достаточно (меньше трех), то ориентацию в этой части помещения лучше производить используя вектор движения, полученный от встроенных датчиков, то есть увеличить коэффициент Калмана для вектора сенсоров.

Для нахождения точного значения коэффициента Калмана, необходимо произвести минимиззацию ошибки:

$$e_{k+1} = x_{k+1} - x_{k+1}^{opt} \quad (6)$$

Используя уравнение (4), получим:

$$e_{k+1} = (1 - K)(e_k + \xi_k) - K\eta_{k+1} \quad (7)$$

Для рассматриваемой системы критерием минимизации можно считать среднее значение от квадрата ошибки:

$$E(e_{k+1}^2) \to min \quad (8)$$

Таким образом, применяя уравнение (7) и критерий минимизации (8), получаем:

$$E(e_{k+1}^2) = (1 - K)^2\left(Ee_k^2 + \sigma_\xi^2\right) + K^2\sigma_\eta^2 \quad (9)$$

Минимальное значение уравнение (9) принимает при:

$$K_{k+1} = \frac{Ee_k^2 + \sigma_\xi^2}{Ee_k^2 + \sigma_\xi^2 + \sigma_\eta^2} \quad (10)$$

Подставив в среднеквадратичкую ошибку $E(e_{k+1}^2)$, миниммизирующее значение коэффициента Калмана K_{k+1}, получим следующее:

$$E(e_{k+1}^2) = \frac{\sigma_\eta^2(Ee_k^2 + \sigma_\xi^2)}{Ee_k^2 + \sigma_\xi^2 + \sigma_\eta^2} \quad (11)$$

Таким образом мы полностью интегрировали фильтрацию Калмана по данным датчиков к существующему алгоритму трилатерации.

Формула (11) является итерационной и позволяет находить коэффициент Калмана. Формулы (5) и (11) позволяют найти приближения расчитанной координаты к вектору скорости, путем подсчета коэффициента Калмана.

Многочастотный фильтр – последовательный метод Монте-Карло – рекурсивный алгоритм для численного решения проблем оценивания – фильтрации и сглаживания. Особенно для нелинейных и негауссовских случаев. В отличии от фильтра Калмана многочастотные фильтры не зависят от методов линеализации и апроксимации. Детально рассматривать данный подход не имеет смысла, так как эмпирическим путем было выявлено, что блягодаря своей пошаговой природе,

фильтр Калмана лучше справляется с устранением шума. К тому же, при движении он обеспечивает оптимальное сглаживание, за счет вектора состояния.

1.4. Создание мобильного приложения

1.4.1. Выбор платформы и языка программирования

..Для выбора платформы мобильного приложения необходимо определить, что и для кого нужно написать. Требуется разработать алгоритм для определения местоположения мобильного устройства с использованием маячков iBeacon и встроенных сенсоров. Так как для работы данных алгоритмов нам нужен минимальный набор сенсоров в телефона, состоящий из гироскома, магнетометра и акселерометра, можно сделать вывод что нам необходимо разработать натичное приложение для смартфонов под управлением наиболее популярных мобильных операционных систем, а именно iOS и Android.

Операционные системы iOS и Android имеют различные инструменты для нативной разработки. Подходы при разработке приложения для мобильных устройств можно разделить на три типа:

1. Веб приложения. Самый быстрый способ разработать мобильное приложение. Благодаря принципам гибкой разметки и современным стандартам современных веб технологий, становится возможным реализация веб приложений с элементами, характерными нативной разработке, такими как кеширование, оффлайн работоспособность, возможность получить доступ к большому числу API операционной системы, такими как доступ к галерее, воспроизведение музыки, использование внутренных сенсоров и многое другое. Основными преимуществами данного подходя является отсутствие необходимости установки приложения и кроссплатформенность. Недостатками является ограниченость оффлайн функциональности, сложность поддержки всех браузеров, ограничения в использовании API, например, в результате исследования, данный подход оказался не подходящим для решения поставленной задачи, так как современные Web фреймворки не дают доступ к необходимому API для работы с Bluetooth, и в частности с iBeacon.

2. Нативные приложения. Такими приложениями обычно называют приложения, требующие установки и разрабатываемые отдельно для каждой платформы. Для нужных нам платформ, iOS и Android, необходимо использовать разные принципы и подходы разработки, начиная от дизайна и заканчивая принципами проектирования. Так же для реализации нативной разработки, нужно использовать различные языки, инструменты и среды разработки. Для разработки приложений для операционной системы iOS, можно использовать такие языки как Objective-C, Swift, среду разработки Xcode и менеджер зависимостей Cocoapods. Для Для разработки приложения под операционную систему Android необходимо использовать такие языки как Java или Kotlin, среду разработки Android Studio и менеджер зависимостей Gradle. К тому же для разработки интерфейсов приложений, необходимо придерживаться принципам построения интерфейсов, Human Interface Guidelines для iOS и Material Design Principles для Android. Несмотря на большие временные затраты, в сравнении с веб приложениями, нативная разработка обладает такими преимуществами, как безопасность, скорость работы, поддержка всех современных механизмов, доступных при обновлении системы, работа в фоне, работа без доступа к сети.
3. Гибридные. Создание гибридных приложений находится всегда между веб и нативной разработками. Существует множество инструментов, одни из которых делают упор на веб сторону, другие на нативность. Основной принцип создания гибридного приложения – создать одно приложение, унивесальное для всех платформ с доступом к нативному API. До недавнего времени гибридная разработка являлась очень востребованной, но применялась в основном в простых приложениях, без повышеных требований к безопасности, скорости работы, пользовательскому опыту использования и работе в оффлайн режиме. Но в середине 2015 года мир гибридных приложений переродился благодаря разработанной на хакатоне Facebook технологии, получившей в дальнейшей название React native. Этот подход к разработке не пытается предлагать создавать одно приложение, для всех

операционных систем, а предполагает вынесение основной бизнесс логики и модели в отдельный переиспользуемый модуль, написанный на javascript. В отличии от других подходов к гибридной разработке мобильных приложений React native предлагает описывать интерфейс пользователя, отрисовку анимаций с использованием нативных языков и средств разработки, что качественно влияет на пользовательский опыт, безопасность или работу в оффлайн режиме.

Очевидно, что для написания кроссплатформенного продукта, использование React Native является оптимальным, и первый прототип приложения был написан именно на этой технологии, но у этого подхода есть минус, о котором все стараются молчать, а именно синхронный мост между нативным кодом и кроссплатформенным, то есть все сообщения по обработке пользовательских касаний, скрола карты и передача данных о текущем местоположении пользователя идут синхронно в главном потоке, что приводит к замораживанию пользовательского инферфейса при большом передаваемых данных между блоками приложения, а это происходит при появлении большого количества маячков. Причина этого заключается в том, что модуль получения информации о маячках возможно описать только на уровне нативного приложения, а алгоритм расчета местоположения должен быть вынесен в кроссплатформенный модуль.
В итоге, для решения данной задачи был интегрирован подход ручной синхронизации уровня подсчета координаты для двух платформ. Эта задача стала намного проще с появлением двух похожих языков разработки для iOS и Andoid, а именно Swift и Kotlin соответственно разница в этих 2 языках на столько незначительна, что путем произведения набора операций по поиску и замене, можно привести логику, написанную с одного языка на другой. На данный момент есть открытые библиотеки, которые могут делать это автоматизировано.

1.4.2. Организация взаимодействия с iBeacon маячками

API для работы с iBeacon маячками было представлено на Apple World Wide Developers Conference в 2013 году вместе с форматом iBeacon. Этот API позволяет устойствам под управлением оперционной системой iOS определять в зоне действия каких маячков они находятся.

Для того чтобы добавить отслеживание iBeacon маячков, нужно подключить к приложению библиотеку CoreLocation. Api встроенный в Core Location способен извещать устройство о том что оно вошло в регион вещания конкретного маячка или вышло из региона его вещания. Вся эта функциональность доступна начиная с iOS 7 и iPhone 4.

Так как iBeacon является частью Core Location и позволяет приложению отслеживать местоположение пользователя, для использования API пользователю нужно будет разрешить приложению доступ к его геоданным. Существуют различные пользовательские истории для реализации взаимодействия с iBeacon маячками. В начале необходимо зарегистрироваться на отслеживание маячков с определенными уникальными идентификаторами (UUID), максимальное количество UUID на отслеживание которых можент подписаться одно приложение 20. После чего возможно зарегистрировать нотификацию о появлении в зоне видимости зарегистрированных маячков, это уведомление появится дае в случае, если приложение было выгружено из памяти. Apple рекомендует использовать эти уведомления для сообщения пользователю что приложение обладает возможностью навигации в здании в которое он зашел. Для регистрации в центре уведомлений приложения, необходимо так же получить от пользователя разрешение на совершение данного типа действий.

Пример 1-1 Создание и регистрация iBeacon региона

```
- (void)registerBeaconRegionWithUUID:(NSUUID *)proximityUUID
andIdentifier:(NSString*)identifier {
     // Создание iBeacon региона для мониторинга
     CLBeaconRegion *beaconRegion = [[CLBeaconRegion alloc]
     initWithProximityUUID:proximityUUID
     identifier:identifier];
     // Регистрация Маячка в менеджере локации
     [self.locManager startMonitoringForRegion:beaconRegion];}
```

Пока пользовательское устройство находится в области вещания маячка, приложение может начать мониторинг информации о маячках в регионе, используя метод startRangingBeaconsInRegion: описанный в классе CLLocationManager.

Пример 1-2 Определение расстояния между iBeacon маячками и устройством

```
     // Метод делегата из протокола, описанного в
CLLocationManagerDelegate.
     -(void)locationManager:(CLLocationManager *)manager
     didRangeBeacons:(NSArray *)beacons
     inRegion:(CLBeaconRegion *)region {
        if ([beacons count] > 0) {
          //самый близкий
           CLBeacon *nearestExhibit = [beacons firstObject];
           if (CLProximityNear == nearestExhibit.proximity) {
               //Отображение информации о маячке к которому подошел
пользователь
             [self
presentExhibitInfoWithMajorValue:nearestExhibit.major.integerValue];
           } else {
               //Скрывание информации о маячке
             [self dismissExhibitInfo];
             }
     }
```

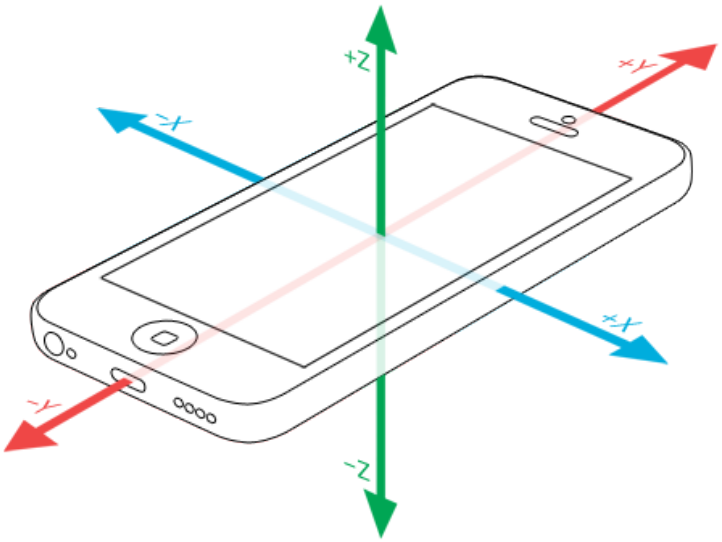
Так же для поиска других устройств с приложением рядом, можно использовать встроенный в мобильное устройство Bluetooth, превратив его в iBeacon маячек:

Пример 1-3 Перевод мобильного устройства в режим iBeacon маячка

```
NSUUID *proximityUUID = [[NSUUID alloc] initWithUUIDString:@"39ED98FF-
2900-441A-802F-9C398FC199D2"];
CLBeaconRegion *beaconRegion = [[CLBeaconRegion alloc]
initWithProximityUUID:proximityUUID identifier:@"ru.vsu.myregion"]
NSDictionary *beaconPeripheralData = [beaconRegion
peripheralDataWithMeasuredPower:nil];
CBPeripheralManager *peripheralManager = [[CBPeripheralManager alloc]
initWithDelegate:selfqueue:nil options:nil];
[peripheralManager startAdvertising:beaconPeripheralData];
```

1.4.3. 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 о 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 ость Ox проходит через устройство слева направо, Oy снизу-вверх, а Oz проходит перпендикулярно сзади вперед:



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 случае если пользователь при первом запуске приложение даст права на это. Класс CMMotionManager предоставляет подобные интерфейсы для каждого типа датчиков, акселерометра, гироскома и магнетометра.

```
let manager = CMMotionManager()
if manager.isGyroAvailable {
     manager.gyroUpdateInterval = 0.1
     manager.startGyroUpdates()
}
```

Для добавления вектора движения и фильтрации состояния покоя, используя фильтр Калмана, нам необхоимо знать, когда устройство находится в состоянии покоя, а когда движется, а так же поворачивать карту туда, куда направлено устройство. Для этого нам нужно подписаться на информацио о обновлениях данных акселеромента:

```
if manager.isAccelerometerAvailable {
    manager.accelerometerUpdateInterval = 0.01
    manager.startAccelerometerUpdates(to: .main) {
        [weak self] (data: CMAccelerometerData?, error: Error?) in
        if let acceleration = data?.acceleration {
            let rotation = atan2(acceleration.x, acceleration.y) -
M_PI
            self?.mapView.transform = CGAffineTransform(rotationAngle:
rotation)
        }
    }
}
```

Каждый пакет CMAccelerometerData включает x, y, z занчения, отражающие количество ускорения в g (единица измерения гравитации) для каждой из осей. Например, если пользователь будет держать телефон в портретном режиме, векток (x, y, z) будет выглядеть (0,-1,0), лежащий на столе телефон дает вектор (0,0,-1), а повернутый на 45 градусов (0.707, -0.707, 0). Для сглаживания движения нужно использовать данные гироскопа, для этого нужно аналогично у инстанса класса CMMotionManager вызвать метод startGyroUpdates.

```
if manager.isDeviceMotionAvailable {
    manager.deviceMotionUpdateInterval = 0.01
    manager.startDeviceMotionUpdates(to: queue) {
        [weak self] (data: CMDeviceMotion?, error: Error?) in
        if let gravity = data?.gravity {
            let rotation = atan2(gravity.x, gravity.y) - M_PI
            self?.mapView.transform = CGAffineTransform(rotationAngle:
rotation)
```

```
            self?.imageView.transform =
CGAffineTransform(rotationAngle: rotation)
        }
    }
}
```

Остается проблема определения вектора движения, так как при равномерном движении ускорение может не изменяться, в этом случае, если пользователь запустил приложение уже находясь в равномерном движении, можно использовать класс CMPedometerData, задав промежуток времени в конфигурации которого можно получить доступ к количеству шагов пользователя, тип его текущего движения, узнать расстояние которое он прошел за прошедший промежуток времени и много другое. Таким образом беря промежуток времени 20 секунд до запуска приложения, зная тип движения, количество шагов и пройденное расстояние мы можем расчитать среднюю длинну шага. На основе данных гироскопа мы знаем в каком направлении движется пользователь и величину шага на основе расчетов сопроцессора серии M, таким образом мы можем построить вектор движения, который необходим для фильтрации Калмана.

1.2.5.2. Способы организации работы с CoreData

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

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

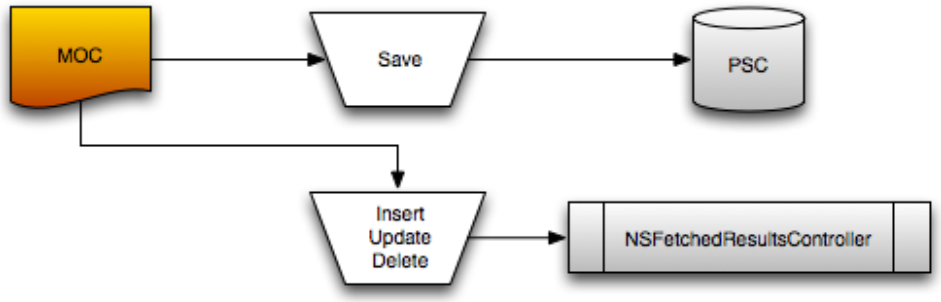


Рис 13. Схема работы в главном потоке

2) 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.

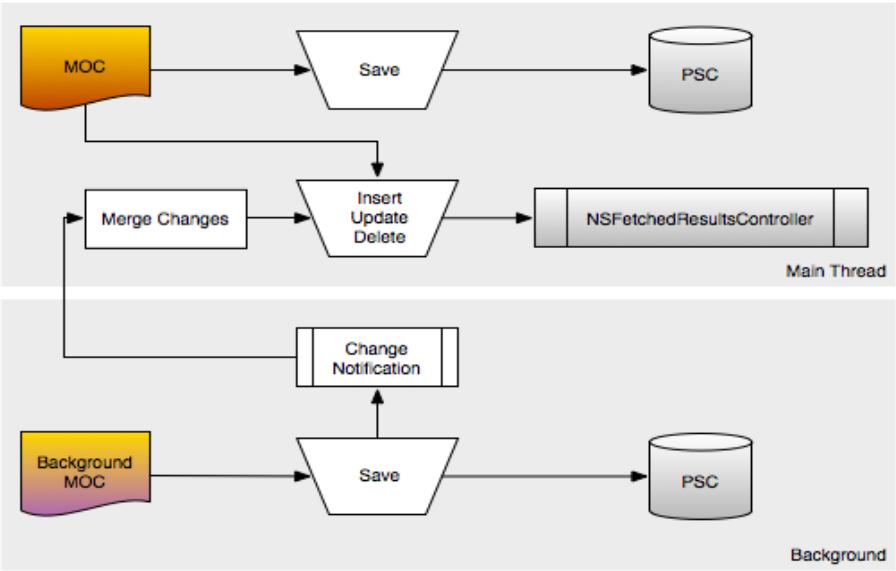


Рис 14. Схема работы в многопоточном режиме

3) 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).

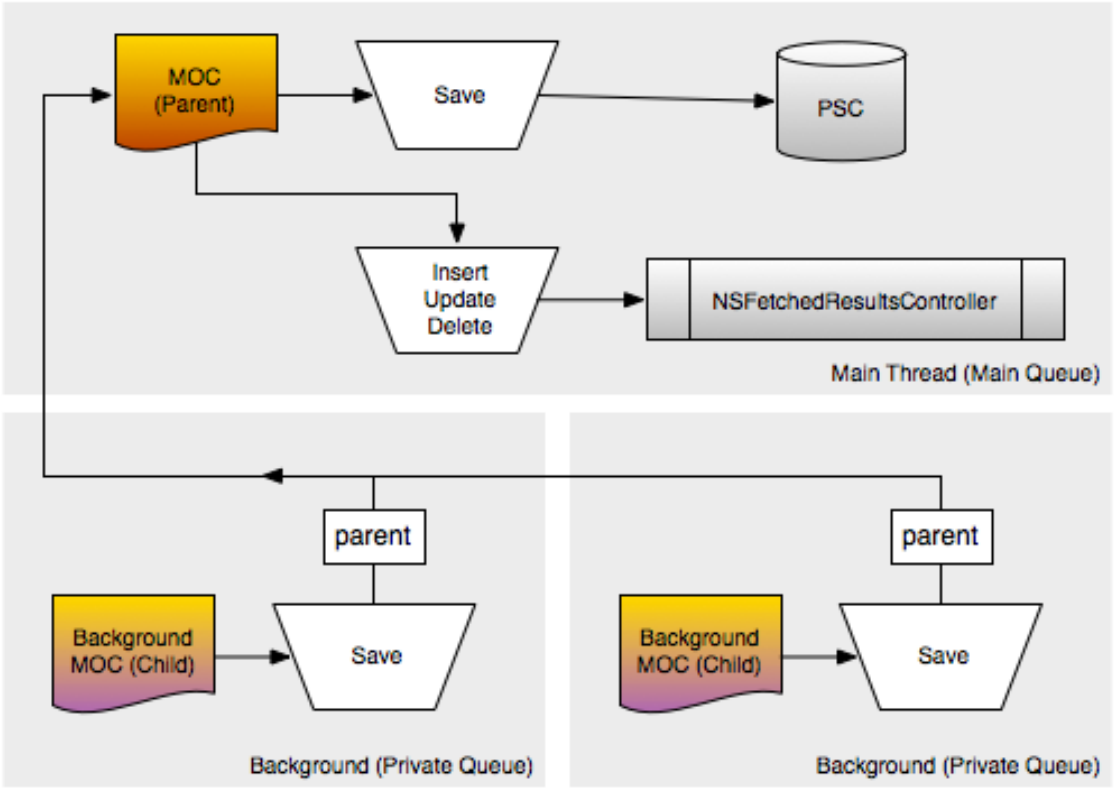


Рис 15. Схема работы с родительскими контекстами

Но любой fetch request приведет к операциям чтония 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".

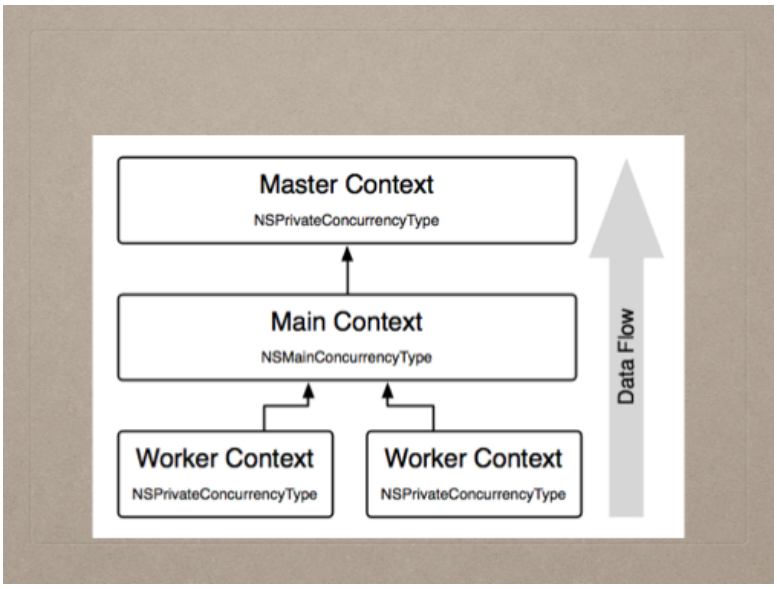


Рис 16. Финальная структура организации работы с CoreData

Идея заключается в том, что master managed object context присоединяется к persistent store с приватным распараллеливанием (операции в фоновом потоке) после чего создается managed object context в главном потоке as a child of this master context. Серьезные задания, такие как добавление крупных данных происходит в worker contexts которые устанавливаются как child contexts для main context с private concurrency type.

1.2.5.3. NSFetchedResultsController

NSFetchedResultsController представляет собой контроллер, предоставляемый фрэймворком Core Data для управления запросами к хранилищу. Этот класс идеально подходит для отображения выборок в таблицу, через него осуществляется запрос к БД, объекты загружаются в оперативную память по необходимости. Так же большим преимуществом является возможность отслеживать изменения в результатах запроса, то есть достаточно добавить/изменить/удалить запись из базы, а FRC сам отследит эти изменения и в случае изменения перезагрузит нужные ячейки с обновленными данными.

1.2.5.4. Схема базы данных

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

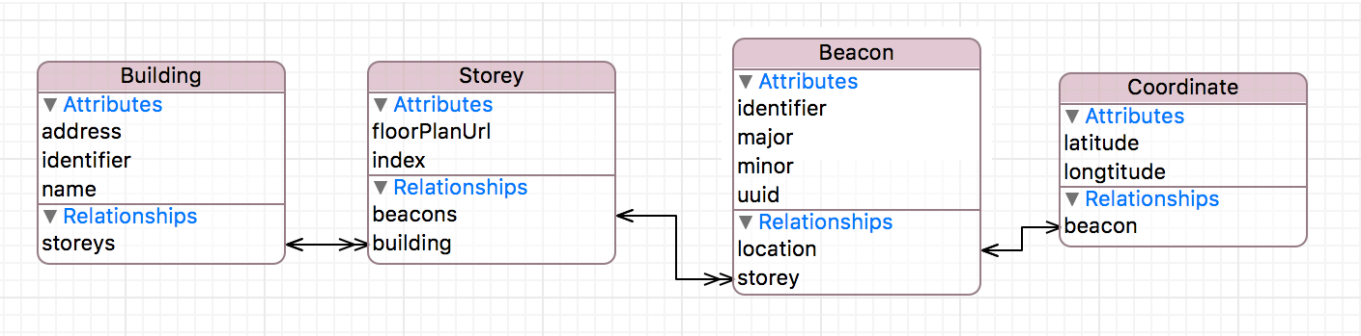


Рис 17. Схема базы данных

На рисунке 17 представлена схема локальной базы данных, основная сущность Beacon. Все атрибуты сущности являются обязательнымы, атрибут identifier является конкатенацией трех сторк uuid+major+minor, обеспечивающее уникальный идентификатор каждого маячка, этот атрибут используется для построения индекса в таблице. Маячек связан с вспомогательной сущностью Coordinate, это сделано для удобства работы с объектным представлением схемы. Сущность Storey – представляет информацию о этаже, а именно ссылку на схему этаже и индекс этажа. Связь этажа с маячками каскадная, что означает что при удалении маячка 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 так же каскадная, при удалении здания, будут удалены все этажи и все маячки каскадом.

1.2.6. Обеспечение безопасности пользовательских данных

Задача обеспечения безопасности очень актуально в современном мире. Схема работы с маячками организована таким образом, что маячек транслирует сигнал, а приложение его принимает, сам же маячек не знает о нахождении пользователя рядом с ним.

Для хранения необходимой программе информации о посещенных местах в программе используется Keychain Services. Согласно документации Apple, Keychain - безопасный контейнер, предоставляющий возможность безопасно хранить пароли, ключи, сертификаты, заметки и подобное. Встроенные средства операционной системы Apple хранят в Keychain сетевые пароли для Wifi, учетные записи Vpn и т. д. Эти данные зашифрованы и хранятся в базе данных sqlite в файле /private/var/Keychains/keychain-2.db. Основные возможности, предоставляемые Keychain:

1. Поиск.
2. Изменение.
3. Добавление
4. Удаление

Эти действия можно производить над объектами keychain.

Для сохранения информации в keychain - используется класс KeychainItemWrapper. При записи объекта необходимо указать идентификатор. Так же при записи объекта в Keychain необходимо указать значение kSecAttrAccessible, которое регулирует уровень доступа к данным keychain. Существует 6 возможных вариантов ограничения уровня доступа:

1. kSecAttrAccessibleAfterFirstUnlockK - к информации нет доступа после перезагрузки устройства, пока устройство не разблокировано пользователем. Записи с этим атрибутом мигрируют на новое устройство при использовании зашифрованных резервных копий
2. kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly - к информации нет доступа после перезагрузки устройства. Записи с этим атрибутом не мигрируют на новое устройство.

3. kSecAttrAccessibleAlways - данные будут доступны всегда, даже если устройство заблокировано. Записи с этим атрибутом мигрируют на новое устройство.
4. kSecAttrAccessibleAlwaysThisDeviceOnly - данные будут доступны всегда, даже если устройство заблокировано.Записи с этим атрибутом не мигрируют на новое устройство.
5. kSecAttrAccessibleWhenUnlocked - данные будут доступны только тогда, когда устройство разблокировано. Записи с этим атрибутом мигрируют на новое устройство.
6. kSecAttrAccessibleWhenUnlockedThisDeviceOnly - данные будут доступны только тогда, когда устройство разблокировано пользователем. Записи с этим атрибутом не мигрируют на новое устройство.

Для целей данного проекта выбран kSecAttrAccessibleWhenUnlocked, так как доступ нужен только активному приложению и миграция на другие устройства всегда удобна.

2.4. Дополнительные инструменты

2.4.1. Система управления версиями файлов

В качестве системы управления версиями файлов использовалась git. При разработке програмного продукта важно хранить все изменения, желательно на нескольких машинах. При разработке проекта над ним, как правило, работают несколько людей, возникает необходимость иметь последнюю версию продукта со всеми изменениями. Эти функции выполняют системы контроля версиями. Самыми популярными версиями таких систем являются SVN и GIT. При работе над данным проектом был выбран Git из-за ряда факторов:

- Наличие удобного механизма для работы с ветками.
- Распределенность рабочих копий.
- Хранение метаданных а не файлов.

2.4.2. Система отслеживания ошибок

В качестве системы отслеживания ошибок использовалась Jira. Системы отслеживания ошибок существуют для того чтобы документировать процесс разработки. Если рассматривать Jira то при ее использовании заводятся задачи которые переводятся в список тех которые нужно сделать. Задачи могут быть разного типа:

- Задания
- Эпики
- Стори
- Ошибки

Каждая задача имеет ряд состояний, например «нужно сделать» - «в процессе» - «проверяется» - «готова». Данный подход позволяет систематизировать процесс разработки. Рассматриваемая система отслеживания ошибок, помогает с планированием работ. Новые задачи могут добавлять разные пользователи и назначать исполнителя, который в свою очередь может оценить затраты времени на задачу, оставить свой комментарий или переводить на другого исполнителя; каждое действие фиксируется на общем экране, таким образом, руководителю проекта не

нужно следить за каждым участником и узнавать его статус, а достаточно лишь смотреть на общий статус работ.

2.4.3. Система сбора статистики, аналитики и обратной связи

В процессе разработки, тестирования, а особенно после выпуска продуктов очень важно собирать данные пользователей. Их вопросы, отзывы, ошибки и проблемы. Системы аналитики мобильных приложений активно развиваются и конкурируют, большенство из них предоставляются бесплатно пока приложением пользуется небольшое количество людей. В текущем проекте интегрированны следующие системы:

- Mixpanel
- Flurry
- Crashalytycs
- Testflight

Каждая система отвечает за свою часть действий:

Mixpanel собирает информацию об активности пользователей, о том как часто они пользуются приложением, куда они заходят в приложении, что не используют. Этот инструмент позволяет анализировать проблемные места приложения, выявлять недостатки и добавлять/удалять возможности приложения которые требуются/не используются. Так же данный инструмент используется для A/B тестирования. Общий прицип данного тестирования состоит в том, что чать пользователей пользуются одной версией программы, а часть другой и исходя из данных можно понять какие изменения и как повлияли на удобство использования приложением.

Flurry данный инструмент используется для агрегирования обратной связи пользователей, если у человека возникают проблемы, вопросы или приложения, он может написать в обратную связь и вместе с письмом отображается его активность, ошибки с которыми он сталкивался и вся информацию, которая в дальнейшем может понадобиться разработчикам.

Crashalytycs - система обработки ошибок приложения, позволяет отлавливать все ошибки на устройствах пользователей. Этот инструмент по каждой неисправности формирует отчет, в котором содержится информация о версии,

конфигурации программы, а так же при каких манипуляциях и даже при вызове какого метода пошел сбой. Так же этот инструмент позволяет распространять тестовые версии продукта между тестировщиками.

Testflight был выбран для тех же целей, что и Crashalytycs но гораздо раньше. В процессе создания проекта Apple купила Testflight и интегрировала его в свой сервис публикации приложений itunesConnect.com. К сожалению, этот сервис не поддерживает iOS 7, и заменой сервису Testflight выступил Crashalytycs.

ЗАКЛЮЧЕНИЕ

В результате проделаной работы были изучены изучены подходы для решения задачи определения местоположения внутри помещений. Был разработан прототип системы, позволяющей успешно выполнять навигацию внутри помещений| Прототип удовлетворяет требованиям по точности определения местоположения.

В ходе решения поставленых целей были решены следующие задачи:

- Рассмотрены существующие технологии, применяемые в навигации внутри помещений.
- Реализованы подходы, наиболее отвечающие решению поставленой задачи.
- Проведена оценка работоспособности технологий, выделены плюсы и минусы каждого
- Реализован алгоритм итеративной трилатерации
- Интегрирован фильтр Калмана с использованием встроенных сенсоров для оценки состояния системы

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация Apple: [сайт] - (URL: http://developer.apple.com/) (дата обращения 15.09.2014).
2. Bing-Fei Wu, Cheng-Lung Jen, and Kuei-Chung Chang. Neural fuzzy based indoor localization by Kalman filtering with propagation channel modeling. In IEEE International Conference on Systems, Man and Cybernetics, 2007 , pages 812-817, Oct. 2007.
3. Erin-Ee-Lin Lau and Wan-Young Chung. Enhanced RSSI-based real- time user location tracking system for indoor and outdoor environ- ments.In ICCIT 07: Proceedings of the 2007 International Conference on Convergence Information Technology , pages 1213-1218, Washing- ton, DC, USA, 2007. IEEE Computer Society.
4. Sebastian Thrun, Wolfram Burgard and Dieter Fox, Probabilistic Robotics, September 2005
5. Zhe Chen, Bayesian filtering: From Kalman filters to particle filters, and beyond, Technical report, McMaster University, 2003.
6. Ioannis M. Rekleitis, A particle filter tutorial for mobile robot localiza- tion, Technical Report TR- CIM-04-02, Centre for Intelligent Machines, McGill University, 3480 University St., Montreal, Quebec, Canada H3A 2A7, 2004.
7. J. Carpenter, P. Clifford, and P. Fernhead, An improved particle filter for non-linear problems,tech. rep., Department of Statistics, University of Oxford, 1997.
8. Згуровский, М. З. Аналитические методы калмановской фильтрации для систем с априорной неопреде- ленностью / М. З. Згуровский, В. Н. Подладчиков. – Киев : Наукова думка, 1995. – 298 с.
9. Оценивание состояния динамической системы в условиях неопределенности / В. И. Ширяев, В. И. Дол- бенков, Е. Д. Ильин, Е. О. Подивилова // Экстремальная робототехника : сб. докл. Междунар. науч.-техн. конф. – СПб., 2011. – С. 234–243.

10. Кац, И. Я. Минимаксная многошаговая фильтрация в статистически неопределенных ситуациях / И. Я. Кац, А. Б. Куржанский // Автоматика и телематика. – 1978. – No 11. – С. 79–87.
11. Климченко, В. В. Планирование измерений параметров контролируемых технических объектов / В. В. Климченко // Труды Междунар. симп. Надежность и качество. – 2011. – Т. 1. – С. 46–48.
12. Еременко, В. В. Динамика информационных потоков в системе управления сложным техническим ком- плексом / В. В. Еременко // Труды Междунар. симп. Надежность и качество. – 2012. – Т. 1. – С. 293–294.
13. Петрунин, В. В. Система управления роботом / В. В. Петрунин, Ю. В. Анохина // Труды Междунар. симп. Надежность и качество. – 2013. – Т. 2. – С. 219–220.