Контроль транспортных маршрутов с помощью мобильных телефонов

И.В. Незнанов, Д.Е. Намиот

***Аннотация*— В данной работе рассматривается одна практическая задача сбора и анализа данных от сенсоров мобильных устройств. Для практической реализации была выбрана задача контроля маршрута транспортного средства, актуальная, например, для автопарков. Целью такой системы может быть осуществление контроля за соблюдением транспортным средством предписанного маршрута и графика, информирования пассажиров об ожидаемом времени прибытия транспортного средства. В работе представлены модель и практическая реализация системы контроля маршрута транспортного средства, состоящая из приложения, работающего на мобильном устройстве с операционной системе Android, а также веб-приложения для анализа и визуализации собранных данных.**

***Ключевые слова*—сенсор, Android, маршрут, GPS.**

I. Введение

Мобильные телефоны, планшеты и другие мобильные устройства стали широко распространены в обществе. Каждое такое устройство содержит в себе большое количество датчиков, данные которых (данные акселерометра, GPS-координаты, список доступных WiFi и Bluetooth сетей и т.д.) могут использоваться в прикладных приложениях. Например, мы можем говорить об анализе поведения владельца мобильного устройства, ведении журнала его активности, коллективном анализе поведения группы людей, сборе и анализе метеорологической информации и т.п. В целом, речь идет об использовании концепции, получившей название телефон как сенсор [1]. Идея абсолютно прозрачна – можно отказаться от специализированных устройств сбора информации, а использовать имеющиеся у потребителей (пользователей) мобильные устройства.

В работе рассматривается конкретный пример такого рода задачи – контроль маршрутов транспортного средства (ТС). Устройством сбора информации в такой модели должен служить мобильный телефон, имеющийся у водителя. Приложение, запущенное (работающее) на этом телефоне должно собирать информацию о маршруте движения. Это избавит от необходимости оснащать транспортные средства специализированными “черными ящиками”. И, естественно, нужен инструмент для анализа (отображения) собранных данных.

Очевидно, что сбор данных сенсоров мобильных телефонов, вообще говоря, не привязан к конкретной прикладной задаче. Следовательно, вполне корректно говорить о каких-то более или менее стандартных средствах сбора сенсорных данных с помощью мобильных устройств [2].

Для ускорения и облегчения создания подобных приложений для разных задач, можно разработать библиотеки, предоставляющие разработчику готовый инструментарий для доступа, конфигурирования, управления датчиками мобильного устройства, передачи накопленных данных и выполнения прочих типичных для подобных приложений операций.

В данной квалификационной работе рассматриваются подходы к созданию таких приложений. Приведен анализ областей и практических задач, где подобные приложения востребованы, далее приведен анализ существующих приложений и библиотек для работы с данными мобильных устройств.

Один из существующих фреймворков и был выбран в качестве инструмента для сбора маршрутной информации с помощью мобильного телефона водителя.

Специальное приложение для анализа собранной информации предоставляет следующие возможности:

- Отображение маршрута транспортного средства на карте, контроль времени прохождения контрольных точек;
- Составление и отображение карты скорости ТС;
- Составление и отображение карты покрытия Wi-Fi сетями маршрута движения ТС.

Последний пункт не относится непосредственно к анализу маршрутов и добавлен для будущих расширений (построения системы сбора и обработки контекстно-зависимых данных).

II. Примеры систем, основанных на сборе сенсорных данных с помощью мобильных телефонов

В данном разделе приводятся примеры прикладных систем, основанных на сборе и анализе данных сенсоров мобильных телефонов.

Статья получена 10 июля 2015.

И.В. Незнанов – выпускник Высшей Компьютерной Школы факультета ВМК МГУ имени М.В. Ломоносова (e-mail: ilja-n@yandex.ru).

Д.Е. Намиот – старший научный сотрудник лаборатории открытых информационных технологий факультета ВМК МГУ имени М.В. Ломоносова (e-mail: dnamiot@gmail.com).

A. LifeLogger – ведение лога деятености.

Приложение – лайфлоггер постоянно опрашивает данные с сенсоров мобильного устройства, которое пользователь носит с собой в течение дня, для составления сводки или отчета о его деятельности за день [3]. Данные обрабатываются и индексируются удаленным сервером, на котором пользователь может просмотреть свою деятельность через web-интерфейс. Там он, например, сможет увидеть, что он в 8.00 утра поехал на работу, что заняло 1 час, затем провел на работе 8 часов, после чего поехал в торговый центр, где провел 1,5 часа и вернулся домой за 45 минут. Также данные с мобильного устройства могут быть сегментированы и аннотированы пользователем. Например, если он регулярно совершает велопрогулки примерно в одном районе, то, вернувшись с одной из них, он помечает свою деятельность за последние 1,5 часа как “велопрогулка”, и в дальнейшем сервер распознает его перемещение с примерно такой скоростью примерно в этом районе как велопрогулку. Это дает возможность вести «высокоуровневый” лог деятельности, перечисляющий конкретные виды деятельности вместо банальных «нахождение на месте”, «бег” или «езда в транспорте”.

Задачи классификации активности требуют разработки адаптивных алгоритмов распознавания видов деятельности – это один из вызовов при построении приложений такого рода. Далее, активность пользователя может быть иерархически сегментирована – выделены ключевые активности, в каждой из которых выделены более мелкие активности. Например, для активности «велопрогулка” могут быть выделены под-активности «активная езда” и «отдых”, для каждой собрана своя статистика и выбраны важные данные, которые надо сохранить. Например, для активной езды интересно сохранить GPS-трек, среднюю скорость, распределение скорости по участкам трассы, а если устройство оснащено специальными сенсорами – например, пульсометром – то и их данные. Для активности «отдых” или “работа в офисе” сохранять такие данные нецелесообразно. В конце концов, пользователю предоставляется такой высокоуровневый структурированный иерархический лог его деятельности, в который он может углубиться и получить интересующую его статистику по каждому конкретному виду деятельности.

Наблюдение за людьми с ограниченными возможностями.

Приложения такого рода предназначены для наблюдения и контроля за людьми, которые не в полной мере способны нести за себя ответственность, например, за одинокими престарелыми людьми или людьми с отставанием в развитии [4]. На основе данных с мобильного устройства серверная часть способна распознать, например, факт случайного падения пожилого человека, или тот факт, что носитель устройства долго и хаотично перемещается в одном районе, что может означать что он заблудился и не в состоянии самостоятельно вернуться обратно, и ему требуется помощь. Также приложение такого рода дает опекунам и врачам информацию о деятельности и состоянии носителя устройства [5].

B. Мониторинг социального здоровья отдельных групп граждан.

Есть группы граждан, входящих в некую “группу риска”, которым предписаны периодические посещения врачей или сотрудников правоохранительных органов – например, военные, вернувшиеся из зоны боевых действий или освободившиеся заключенные. Заключение об их состоянии и настрое, психологической устойчивости или подверженности срывам выносится на основе их визитов к врачам, но неизвестно, что происходит с ними во время между визитов. Можно разработать приложение, которое даст наблюдателям больше информации о поведении конкретного человека в это время.

C. Социальная статистика

Еще одной областью применения подобных систем может быть сбор социальной статистики, например, в рамках конкретного города. Распространив среди его жителей мобильное приложение, можно собрать информацию о том, где проходят основные потоки людей, в какое время и какие городские маршруты наиболее перегружены, какие из городских парков пользуются популярностью, а какие – нет, в какое время и сколько людей регулярно уезжает из города на выходные, сколько людей приезжают в город на работу, каково среднее время ожидания автобуса и так далее. Это информация может быть использована для оптимизации управления городской жизнью – подбора подходящего расписания транспорта, первоочередного благоустройства наиболее популярных мест, планировки наиболее удобных дорожек в парках и большого количества других задач [6].

Массовое исследование в рамках географического района.

В такой системе собираются данные с мобильных устройств многих пользователей и классифицируются по месту нахождения пользователей – собирается информация от пользователей, находящихся в одном районе и делается заключение об это районе. После этого информация о соседних районах может быть сведена в единую карту.

Примерами таких систем могут быть

• Информатор о загруженности дорожной сети – система, собирающая данные акселерометров от автомобилистов и группирующая данные от водителей, находящихся на конкретном участке дороги, с целью оценить текущую загруженность этого конкретного участка. Далее эта информация может быть сведена в единую карту загруженности дорог, как это делает известный сервис Яндекс.Пробки.

• Система для оценки качества покрытия операторами сотовой связи или беспроводными сетями. Такая система собирает от пользователей информацию о наличии и уровне сигнала сотовых сетей конкретных операторов мобильной связи, а также об обнаруживаемых данным пользователем беспроводных сетях. Объединяя подобную информацию от различных пользователей, расположенных географически близко друг к другу, можно судить о качестве мобильной связи, о количестве и качестве покрытия Wi-Fi-сетями этого конкретного района, после чего данные по районам можно свести в единую карту покрытия сетями мобильных операторов всего региона [7].

D. Координация действий группы людей в чрезвычайных ситуациях.

Такие системы могут применяться для координации групп людей, выполняющих единую задачу в чрезвычайной ситуации. Примером могут быть боевое подразделение при выполнении боевой задачи или группа спасателей, работающих на месте ЧП, при отсутствии прямой видимости членов группы из-за сложного рельефа, плохой видимости (дым, туман, ночное время суток), ограниченных условий (невозможности выйти в открытый радиоэфир). Такая система позволяет командиру подразделения знать о местонахождении и активности каждого члена группы, а членам группы – координировать свои действия в зависимости от текущей ситуации. При этом участники операции не отвлекаются от выполнения своих непосредственных задач на то, чтобы отчитаться о своем статусе и действиях. Мобильное приложение делает это без их участия. Кроме того, сервер записывает действия каждого члена группы, что потом может быть использовано для воспроизведения и разбора проведенной операции.

III. Приложения и пакеты для работы с данными мобильных устройств

В этом разделе мы остановимся на обзоре пакетов для сбора сенсорной информации с мобильных устройств.

A. Mobisense

Одним из фреймворков является MobiSens, разработка университета Carnegie-Mellon [8]. Это система, состоящая из двух компонентов – мобильное приложение для ОС Android и серверная часть. Основные компоненты системы, а также их взаимодействие показаны на рис 1.

Мобильный клиент предназначен для сбора и передачи данных в серверную часть. Клиент универсален, один на все виды задач. Универсальность достигается благодаря широким возможностям конфигурирования клиента.

Приложение-клиент имеет четыре компонента:

- Сенсоры данных
- Агрегаторы данных
- Контроллер устройства
- Модуль распознавания активности

Компоненты клиентского приложения одновременно являются и источниками, и потребителями сообщений. Они подписываются как слушатели на определенные события, обрабатывают данные, приходящие с этими событиями, и результат рассылают как новые события.

Сенсоры данных – это все источники данных. Они включают в себя аппаратные источники, такие как GPS и акселерометры, и программные источники, например, предоставляющие информацию о дате, времени, календаре, логе звонков и т.д.

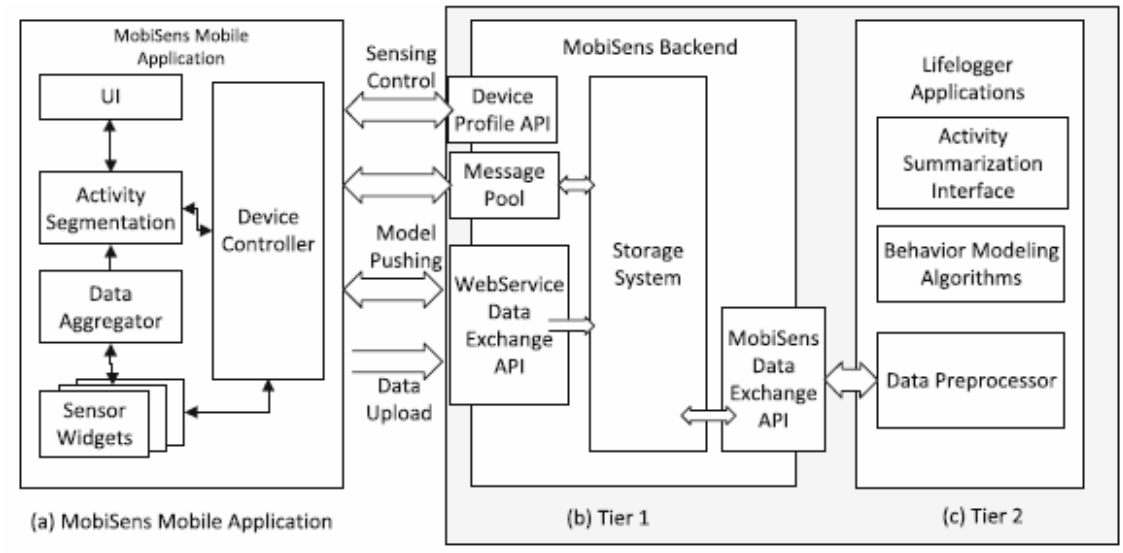

Рис. 1. Mobisense

Сенсоры данных собирают “сырые” данные с устройства и рассылают подписанным компонентам.

Агрегаторы данных – компоненты, которые накапливают данные от разных сенсоров, сохраняют их в многочисленных файлах. Также он управляет очередью загрузки файлов на сервер, помещая файлы с данными в очередь компонента-загрузчика данных, и информируя о том, какие файлы готовы к передаче.

Контроллер устройства – компонент, ответственный за конфигурирование клиента. Он периодически

получает информацию о том, какие сенсоры надо опрашивать, и с какой периодичностью, а также стратегию как именно опрашивать сенсоры и как передавать информацию, от серверной части. Этим осуществляется подстройка мобильного клиента именно под ту задачу и данные, которые интересны конкретному приложению на сервере.

Модуль распознавания активности – модуль, просматривающий очередь событий, подготовленную агрегатором данных и выполняющий простое распознавание активности на клиенте. Таким образом, во-первых, на сервер отправляется уже пред-обработанная информация об активности, во-вторых, мобильное приложение предлагает пользователю просмотреть и проаннотировать эту распознанную активность.

Серверная часть состоит из бэкенд-сервера и сервера приложений. Бэкенд-сервер напрямую обменивается информацией с мобильными устройствами и выполняет следующие функции:

- Управление устройством;
- Получение и хранение данных;
- Управление доступом к данным.

Данные, собранные бэкенд-сервером, через специальный MobiSens Data Exchange API передаются конкретным прикладным приложениям, работающим на сервере приложений, таким, как, например лайфлоггер. Если прикладному приложению требуется передать что-то, например конфигурацию, или отчет об активности для пользователя, на мобильное устройство, оно это делает через бэкенд-сервер.

Особенностью фреймворка являются реализованные сложные алгоритмы для сегментации, распознавания и аннотирования активностей. Этим алгоритмам разработчиками уделяется особое внимание, и программист прикладного приложения получает готовые и мощные инструменты для анализа данных.

Плюсы:

• Широкая функциональность мобильного клиента, много возможностей для его настройки;

• Готовые алгоритмы обработки и классификации деятельности, доступные для программиста прикладного приложения;

• Скрытый от прикладного программиста сбор данных и коммуникация с мобильными устройствами, все данные доступны в backend-сервере через специальный API.

Минусы:

• Не является библиотекой – не предоставляет инструменты для разработки собственных приложений;

• Мобильное приложение нельзя модифицировать;

• Исходный код мобильного приложения и серверной части закрыт.

B.OpenSignal

Еще один пример подобного фреймворка – OpenSignal [9]. Как обычно, это система из мобильного приложения и серверной части. Изначально этот проект предназначался для анализа качества покрытия территории сетями мобильной связи. Он позволяет для конкретной точки определить, сети каких мобильных операторов там присутствуют, сравнить их качество связи. Этот анализ не зависит от данных, предоставляемых мобильными операторами. Также система сканирует Wi-Fi-сети и может показать карту бесплатных Wi-Fi-спотов поблизости, направление, в котором надо двигаться, чтобы улучшить качество сигнала Wi-Fi-сети, имеет тест скорости передачи данных для 3G/4G сетей и Wi-Fi-сетей.

Однако возможности мобильного клиента шире задач мониторинга сетей. Клиент может считывать данные с нескольких десятков сенсоров мобильного устройства. Так, с помощью OpenSignal оценивали число различных типов устройств на ОС Android, их оказалось около 20000. Другой пример – та же команда, которая запускала OpenSignal, запустила проект WeatherSignal [10], где, основываясь на данных от множества мобильных телефонов, составляет сводную карту текущей погоды на некоторой территории. Для этого мобильный клиент анализирует датчики температуры, освещенности, давления, магнитного потока, влажности и прочие. Также, OpenSignal использует накопленные объемы данных для выпуска различных аналитических отчетов, например, о состоянии рынка мобильной связи в Англии, крупнейших бесплатных Wi-Fi-сетях в США.

Сторонним разработчикам OpenSignal предоставляет доступ к тем данным, которые он отображает на своем собственном сайте. Для получения информации о мобильных сетях предназначен NetworkStats API, для получения информации о вышках сотовой связи – TowerInfo API. Например, чтобы получить с помощью NetworkStats API данные о сетях на некотором квадратном участке местности надо отправить HTTP GET-запрос на адрес *http://api.opensignal.com/v2/networkstats.json*, в параметрах указав широту и долготу центра участка, длину стороны квадрата участка, тип интересующих сетей (2G, 3G или 4G). Результат возвращается в ответном сообщении HTTP в виде JSON-строки. Для доступа к этому сервису требуется регистрация на сайте OpenSignal.com, которая предоставляет уникальный API Key, который необходимо указывать в запросе. Использование API ограничено 5 запросами в минуту и 2000 запросами в месяц.

Плюсы:

• Большая база установленных мобильных клиентов – установлен на миллионы устройств по всему миру;

• Нацеленность на конкретную задачу;

• Реальный, не университетский проект, имеющий определенную известность.

Минусы:

• Отсутствие сложных алгоритмов анализа данных на серверной стороне;

• Ограниченный API для сторонних разработчиков – позволяет получить только те данные, которые OpenSignal готовит для своего сайта;

• Не предоставляет инструментов для разработки собственных приложений;

• Исходный код мобильного клиента и серверной части закрыт.

C. SensingKit

SensingKit – фреймворк, разработанный в университете Queen Mary University of London [11]. В отличие от предыдущих фреймворков, он предоставляет библиотеку для разработки мобильных приложений. Также он предоставляет серверную часть.

Особенностью SensingKit является мобильная мульти-платформенность – фреймворк предоставляет библиотеки со схожим интерфейсом для ОС Android и Apple iOS. Также библиотека предоставляет простые возможности классификации активности по аннотированным пользователем образцам. Различаются такие активности, как “покой”, “прогулка”, “бег”, “езда на автомобиле”. На ОС Android также есть активность “езда на велосипеде”.

Основные компоненты фреймворка показаны на рис. 2.

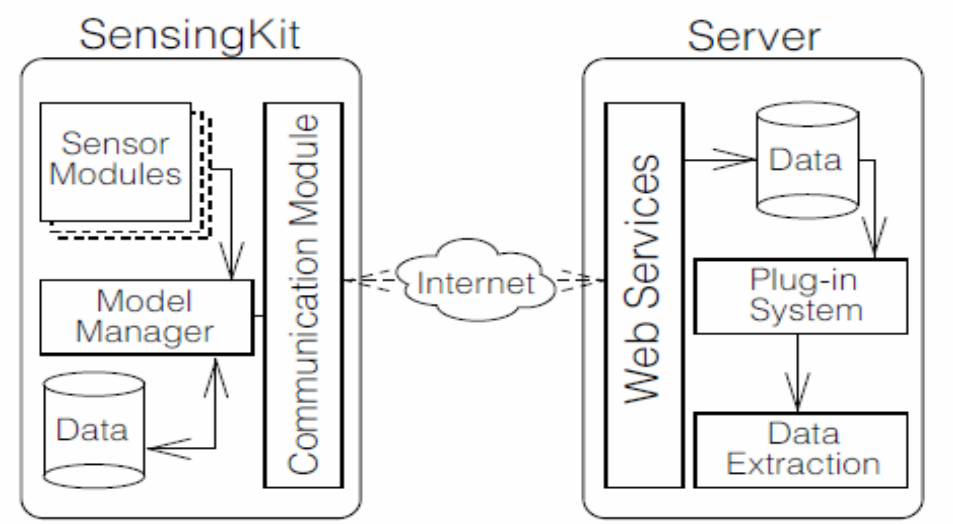

Рис. 2 Архитектура фреймворка SensingKit

Клиентские библиотеки разработаны на целевом для каждой ОС языке – на Java для Android и на Objective-C для iOS. В библиотеке присутствует следующий набор модулей сенсоров, являющихся источниками данных:

• Движение – предоставляет информацию о линейном ускорении, вращении устройства;

• Позиционирование – предоставляет данные от GPS-сенсора;

• Близость - предоставляет данные о близости к другим устройствам используя технологию Bluetooth Smart 4.0

• Питание – предоставляет информацию об уровне заряда батарей.

Также предоставляются данные с гироскопа и датчика магнитного потока.

Как следствие модульности системы, возможна разработка и добавление собственных модулей сенсоров.

Собранные с модулей сенсоров данные сохраняются в памяти мобильного устройства с использованием компонента *ModelManager*. При доступном Wi-Fi-соединении данные отправляются на сервер с помощью компонента *CommunicationManager*.

Серверная часть реализована на языке Python. Она состоит из трех основных компонентов: веб-сервисов, системы плагинов и модуля извлечения данных. Веб-сервисы отвечают за взаимодействие с мобильными устройствами. В их задачи входит назначение уникального идентификатора клиентскому устройству, отправка на клиента времени сервера для последующей синхронизации данных на сервере, получение и сохранение данных от клиента.

Система плагинов представляет интерфейс для подключения пользовательских функций, написанных на языке Python, для предобработки данных перед извлечением. Обязательной частью является плагин синхронизации данных – он обрабатывает данные таким образом чтоб данные с различных устройств были синхронизированы между собой.

Модуль извлечения данных используется для получения данных сторонними приложениями в различных форматах, таких как JSON, CSV и других.

Плюсы:

• Является библиотекой – предоставляет инструменты для создания собственных мобильных приложений;

• Мульти-платформенность – поддерживает ОС Android и Apple iOS;

• Исходный код библиотек для создания мобильного приложения и исходный код серверной части открыт, это open-source проект.

Минусы:

• Небольшое количество поддерживаемых сенсоров;

• Практическое отсутствие готовых алгоритмов анализа данных в серверной части.

D.Funf

Этот проект для разработки мобильных приложений был создан в Массачусетском Технологическом Институте [12]. Сейчас это open-source проект. Его основная часть - это библиотека, предназначенная для создания собственных мобильных приложений для ОС Android. Программисту даются широкие возможности по конфигурированию списка интересующих его датчиков и составлению расписания их опроса. Сохранение данных возможно в файловой системе устройства, либо автоматически на сервер при наличии интернет соединения, в соответствии с указанным программистом расписанием. Серверная часть фактически отсутствует, приводятся примеры скриптов на языке Python для сервера, осуществляющие взаимодействие с мобильным устройством. Также существует технология funf-in-a-box, позволяющая создавать и использовать мобильные приложения вообще без программирования. Пользователь указывает необходимые ему датчики, после чего автоматически создается мобильное приложение и помещается в аккаунт пользователя на сервисе *dropbox.com*. После установки на мобильное устройство приложение автоматически помещает файлы с накопленными данными в аккаунт пользователя на сервисе *dropbox.com*.

Плюсы:

- Мощная библиотека для создания собственных мобильных приложений;
- Поддерживает большое количество сенсоров мобильного устройства;
- Имеет гибкие механизмы создания расписания опроса сенсоров.

Минусы:

- Практически не имеет серверной части.
- Слабая документация.

IV Практическая реализация

Практическая реализация системы контроля маршрутов состоит из двух приложений: мобильного приложения для сбора информации с устройства (мобильного приложения) и приложения для обработки и визуализации полученной информации (приложения-обработчика).

Задачами мобильного приложения является сбор и архивация данных со следующих датчиков мобильного устройства:

- Сенсор информации об устройстве
- GPS – сенсор
- Сенсор Wi-Fi сетей.

Приложение-обработчик извлекает и расшифровывает эти данные, выполняет их анализ и визуализацию результатов. Анализ результатов может быть самым разным, в текущей реализации доступно следующее:

- Отслеживание маршрута движения транспортного средства, результат – отображение пройденного маршрута на карте и фиксация времени прохождения контрольных точек, расположенных вдоль маршрута.
- Составление карты скорости движения транспортного средства – разбиение пройденного маршрута на участки, подсчет средней скорости на каждом участке и ее отображение на карте с помощью палитры цветов.
- Определение покрытия маршрута движения транспортного средства Wi-Fi-сетями, примерный анализ размера Wi-Fi-сетей вдоль маршрута движения, отображение на карте наиболее крупных Wi-Fi-сетей.

Мобильное приложение было создано с использованием библиотеки Funf. Менеджер Funf конфигурируется следующим образом: создается одна очередь сообщений, с сенсорами типа *LocationProbe*, *WifiProbe* с интервалом опроса 5 секунд и *HardwareInfoProbe* интервалом опроса 60 секунд.

Накопленные данные сохраняются в файле в файловой системе мобильного устройства в формате базы данных *Sqlite*. Всю работу по формированию этого файла выполняет библиотека Funf.

Результатом работы мобильного приложения является файл базы данных *Sqlite*, содержащий записи со значениями данных от трех сенсоров. Для дальнейшей обработки этот файл вручную переносится из файловой системы мобильного устройства на компьютер, где работает приложение-обработчик. Одним из наиболее очевидных развитий данной квалификационной работы является реализация полноценной серверной части и использование функциональности библиотеки Funf, связанной с автоматической загрузкой данных с мобильного устройства на сервер по доступному интернет-соединению.

После возвращения ТС в парк файл с накопленными данными переносится на стационарный компьютер для анализа и визуализации, этим занимается приложение – обработчик. Оно реализовано как JavaSE – приложение.

Основными компонентами приложения являются

- Парсер JSON-строк из файла с данными;
- Хранилище данных об устройстве (HardwareInfoStorage);
- Хранилище данных о GPS-локациях (LocationStorage);
- Хранилище данных об обнаруженных Wifi-сетях (WifiStorage);
- Формирователь HTML-страниц (HtmlPrinter);

Обработку данных производят классы – хранилища. Через общий экземпляр JDBC-соединения с Sqlite-базой данных в файле, полученном с устройства, каждый из них запрашивает записи соответственно сенсора информации об устройстве, GPS-сенсора и Wifi-сенсора. Это производится стандартными средствами JDBC с использованием классов *Statement* и *ResultSet*. Запись сенсора представляет собой строку в формате JSON, формат записи свой для каждого сенсора. Например, запись GSP-сенсора имеет следующий вид:

```
{"mAccuracy":3.0,"mAltitude":213.30000
30517578,"mBearing":141.1,"mElapsedRealt
```

imeNanos":5655641539595,"mExtras":{"satellites":12},"mHasAccuracy":true,"mHasAltitude":true,"mHasBearing":true,"mHasSpeed":true,"mIsFromMockProvider":false,"mLatitude":55.91939068,"mLongitude":37.39432066,"mProvider":"gps","mSpeed":15.75,"mTime":1432552282000,"timestamp":1432509020.776}

Основная информация, запоминаемая хранилищем локаций – это список записей, хранящий время получения данных от GPS-сенсора, широту, долготу и скорость ТС в этот момент времени. Под локацией здесь и далее понимается одна такая запись хранилища локаций. Список упорядочен по времени получения записей.

Основная информация, запоминаемая хранилищем данных об устройстве – это Android ID, модель и производитель мобильного устройства.

Основная информация, запоминаемая хранилищем данных о Wi-Fi-сетях – это упорядоченный по времени их получения список записей, хранящих SSID и BSSID обнаруженной сети.

Использование информации об устройстве (hardware info):

- Android ID устройства;
- Производитель мобильного устройства;
- Модель мобильного устройства.

Android ID используется для установления соответствия между водителем ТС и мобильным устройством. Производитель и модель устройства отображаются просто для полноты информации. Обработка этой информации весьма проста, из базы данных берется первая по времени запись от сенсора информации об устройстве и запоминается в соответствующем хранилище. Это допустимо, поскольку эти параметры не меняются в течение всего времени жизни устройства.

Полученная из файла базы данных и разобранная информация о локациях сохраняется в виде списка локаций, упорядоченного по времени их получения. В зависимости от длительности маршрута число таких записей может составлять тысячи и десятки тысяч.

Для построения трека маршрута требуется сформировать список локаций и поместить его в шаблон HTML страницы. Весь список локаций избыточен, для построения трека требуется его регулярно проредить, оставив порядка 50-100 точек, равномерно распределенных по маршруту. Сама отрисовка трека осуществляется кодом на языке Javascript с использованием Yandex.Maps API; сам скрипт встроен в шаблон генерируемой HTML-страницы. После добавления списка локаций шаблон HTML-страницы превращается в полноценную HTML-страницу, отображаемую в браузере.

Расписание задается как преопределенный список локаций – контрольных точек (КТ). Для каждой из них требуется определить время прохождения точки – временную метку той локации на маршруте, которая наиболее близка к данной контрольной точке. Это выполняет хранилище локаций. Список времен прохождения всех контрольных точек формирует расписание маршрута, которое может быть использовано для контроля, сбора статистики и других задач. Пример окончательной html-страницы с треком маршрута ТС приведен на рис. 3

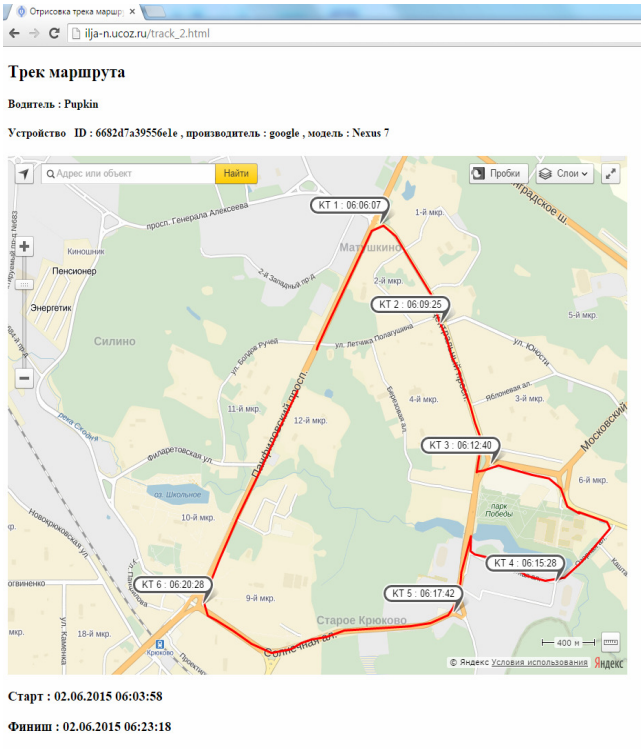

Рис.3 Отображение трека маршута

Для построения карты скорости, помимо отрисовки трека ТС, требуется разделить весь трек на несколько десятков коротких интервалов, координаты концов которых надо передать в формирователь HTML-страниц. Также для каждого участка надо посчитать среднюю скорость – среднее арифметическое значений скорости для всех отметок на этом участке. Это делает хранилище локаций. Подготовленные данные оно передает формирователю HTML-страниц. Далее исходя из значения средней скорости для каждого участка требуется выбрать цвет для его отрисовки. Пример сформированной html-страницы с картой скоростей приведен на рис. 4

Основной информацией, запоминаемой в хранилище данных о сетях, является упорядоченный по времени их получения список Wi-Fi-сетей, обнаруженных устройством. Каждый инициируемый мобильным устройством опрос Wi-Fi-сенсора приводит к получению последовательности записей (одна запись для каждой определяемой в данном месте Wi-Fi-сети), получаемой в течение нескольких секунд. Эта информация (временная метка получения записи, SSID и BSSID мобильной сети) и помещается в хранилище данных о Wi-Fi-сетях. Для построения карты покрытия, для каждой сети требуется определить

- Диаметр сети – максимальное расстояние между точками, в которых обнаруживается данная сеть;
- Центр сети – примерное место расположения точки

доступа.

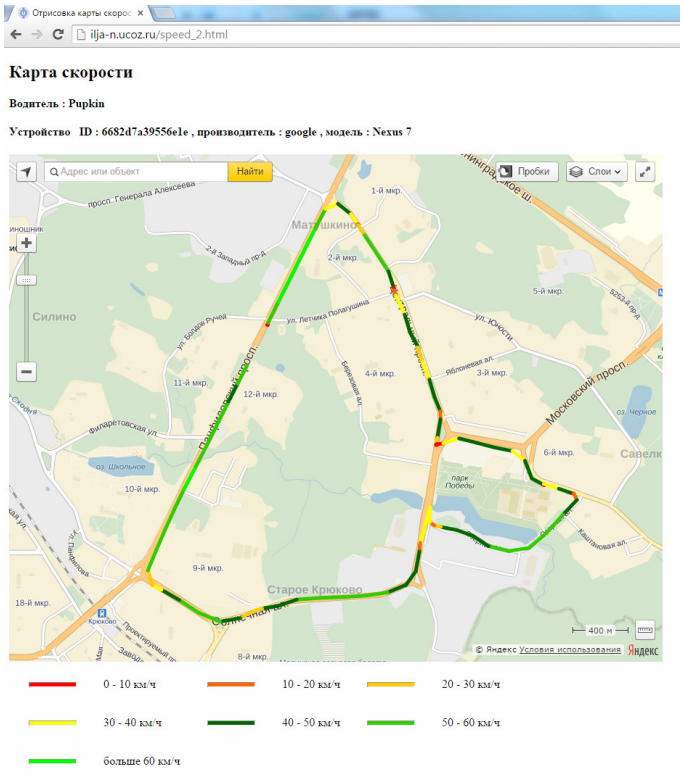

Рис.4 Отображение карты скорости

Для решения данных задач, надо каждую запись об обнаруженной Wi-Fi-сети привязать к координатам. Эту информацию хранилище данных о Wi-Fi-сетях получает от хранилища локаций: для каждой записи о Wi-Fi-сети оно получает ближайшую локацию, где ТС находилось в этот момент. То есть сопоставление записей об обнаруженных Wi-Fi-сетях и локациях ТС происходит по временным меткам: для каждой записи о Wi-Fi-сети хранилище локаций возвращает локацию ТС, временная метка которой наиболее близка к временной метке данной записи.

На основе этой привязки хранилище данных о WiFi-сетях для каждой сети вычисляет ее диаметр как максимальное расстояние между двумя локациями, в которых определялась эта сеть. Также определяется примерный центр сети как среднее положение среди всех точек, где определялась эта сеть. После этого информацию о 10 наиболее крупных сетях хранилище данных передает в формирователь HTML-страниц, который генерирует HTML-страницу, на которой эти сети схематично изображены. Пример сформированной HTML-страницы с картой покрытия приведен на рис. 5

Генерацией HTML-страниц, которые визуализируют результат анализа данных от мобильного устройства, занимается формирователь HTML-страниц. Отображение осуществляется скриптом на JavaScript с использованием Yandex.Maps API. В приложении-обработчике подготовлено несколько шаблонов страниц, например для отображения трека, для отображения Wifi-сетей. Скрипты, осуществляющие отображение, встроены в шаблоны, но вместо конкретных данных в них вставлены специальные заполнители-плейсхолдеры. Формирователь HTML-страниц преобразует шаблон страницы в страницу, вставляя полученные из объекта-хранилища данные на место подходящих заполнителей. Таким образом формирователь HTML-страниц генерирует корректный Javascript-код в результирующей странице.

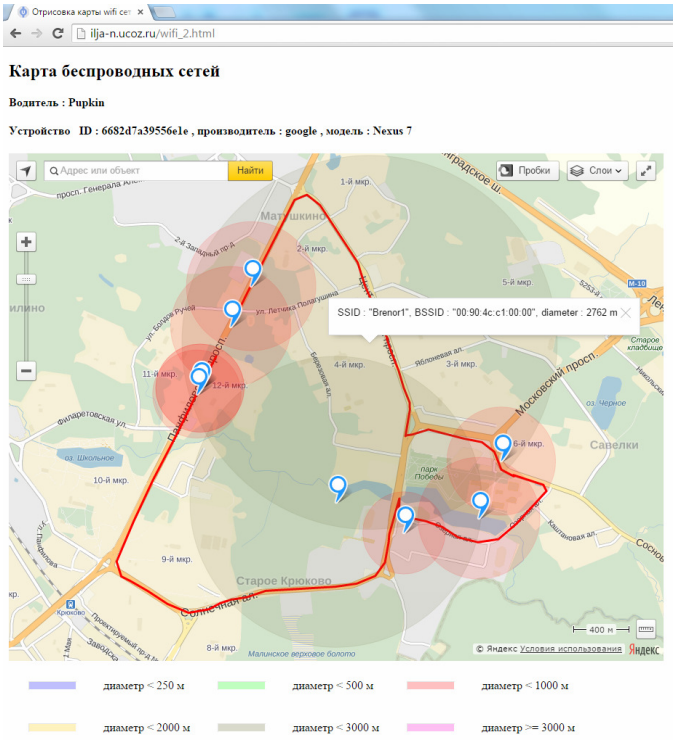

Рис.5 Отображение карты покрытия Wi-Fi-сетей

После этого готовая HTML-страница помещается на web-сервер, JavaScript-код отображает результаты анализа данных.

V Заключение

В данной работе рассмотрена реализация системы контроля движения транспортного средства на базе мобильного телефона с ОС Android. Для сбора информации с сенсоров мобильного телефона используется библиотека Funf. Реализована модель серверного приложения-обработчика, извлекающая и анализирующая информацию от мобильного устройства с целью решения трех задач:

- Построение расписания движения ТС по маршруту;
- Построение карты скорости движения ТС;
- Построение карты покрытия Wi-Fi-сетями маршрута движения ТС.

Разработанная модель фреймворка иллюстрирует методику построения подобной системы, а именно сбора данных с сенсоров мобильного устройства и анализа результатов в серверной части.

Библиография

[1] Lane, N. D., Miluzzo, E., Lu, H., Peebles, D., Choudhury, T., & Campbell, A. T. (2010). A survey of mobile phone sensing. Communications Magazine, IEEE, 48(9), 140-150.

[2] Namiot, Dmitry, and Manfred Sneps-Sneppe. "On Open Source Mobile Sensing." Internet of Things, Smart Spaces, and Next Generation Networks and Systems. Springer International Publishing, 2014. 82-94.

[3] Chennuru, S., Chen, P. W., Zhu, J., & Zhang, J. Y. (2012). Mobile Lifelogger–Recording, Indexing, and Understanding a Mobile User's Life. In Mobile Computing, Applications, and Services (pp. 263-281). Springer Berlin Heidelberg.

[4] Sneps-Sneppe, M., & Namiot, D. (2015). Smart Socket for Activity Monitoring. arXiv preprint arXiv:1502.06904.

[5] Намиот, Д. Е., & Шнепс-Шнеппе, М. А. (2015). Устройство для мониторинга активности в умном доме. International Journal of Open Information Technologies, 3(2), 23-26.

[6] Волков А. А., Намиот Д. Е., Шнепс-Шнеппе М. А. О задачах создания эффективной инфраструктуры среды обитания //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 7. – С. 1-10.

[7] Namiot, D., & Schneps-Schneppe, M. (2013). Smart Cities Software from the developer's point of view. arXiv preprint arXiv:1303.7115.

[8] Reddy, Sasank, et al. "MobiSense—mobile network services for coordinated Participatory Sensing." Autonomous Decentralized Systems, 2009. ISADS'09. International Symposium on. IEEE, 2009.

[9] Opensignal http://opensignal.com Retrieved: Jul, 2015

[10] WeatherSignal http://weathersignal.com Retrieved: Jul, 2015

[11] Katevas, K., Haddadi, H., & Tokarchuk, L. (2014, September). Poster: Sensingkit: A multi-platform mobile sensing framework for large-scale experiments. In Proceedings of the 20th annual international conference on Mobile computing and networking (pp. 375-378). ACM.

[12] Funf Open Sensing Framework http://funf.org Retrieved: Jul, 2015

Control of transport routes via mobile phones

Ilya Neznanov, Dmitry Namiot

Abstract— This paper considers a practical task of collecting and analyzing data from sensors of mobile devices. For practical implementation was chosen the task of monitoring the route of the vehicle. The purpose of such a system may be monitoring compliance with the vehicle prescribed route and schedule, informing passengers about the expected arrival time of the vehicle. The paper presents the model and the practical implementation of the monitoring system of the route of the vehicle, consisting of an application running on a mobile device with the operating system, Android, as well as web application for analysis and visualization of data collected.

***Keywords*—sensor, Android, route, GPS.**