ТЕХНИЧЕСКИЕ НАУКИ

Жукович Ярослав Петрович
инженер
Белорусский государственный университет
информатики и радиоэлектроники
г. Минск, Республика Беларусь

АНАЛИЗ МЕТОДОВ И АРХИТЕКТУР СИСТЕМ СПУТНИКОВОГО МОНИТОРИНГА ТРАНСПОРТА

***Аннотация**: статья посвящена архитектуре систем спутникового мониторинга транспорта. Раскрываются методы построения уровней системы спутникового мониторинга транспорта. Проводится анализ архитектуры по разработанным уровням. В результате анализа автор делает вывод о целесообразности проектирования новой архитектуры, в которой будут учитываться недостатки существующих подходов.*

***Ключевые слова**: спутниковый мониторинг транспорта, системы спутникового мониторинга транспорта, методы спутникового мониторинга транспорта, архитектура систем спутникового мониторинга транспорта.*

В архитектуре любой системы (здесь и далее система спутникового мониторинга транспорта будет упоминаться как система), независимо от ее типа, присутствует разделение на определенные слои или уровни. При детальном анализе, архитектуру системы можно разделить на 6 уровней, каждый из которых выполняет свою задачу и предоставляет интерфейсы соседним уровням. Наличие или отсутствие какого-либо из уровней может обуславливаться задачей либо недостаточной проработкой архитектуры.

Таким образом, в архитектуре системы можно выделить следующие уровни:

1. Уровень устройств.
2. Уровень взаимодействия с устройствами.
3. Уровень данных.
4. Уровень бизнес-логики.

5. Уровень взаимодействия с клиентами.

6. Уровень клиентов.

На *первом уровне* рассматриваются физические устройства (трекеры). Этот уровень по сути является внешним для системы, однако оказывает значительное влияние на проектирование вышестоящего уровня взаимодействия с устройствами. При анализе данного уровня рассматриваются функциональные возможности устройств.

Управляемые устройства поддерживают возможность управления их поведением в реальном времени. Например, на такой тип устройств можно отправить специальную команду, которая повлечет за собой блокирование дверей автомобиля. Такая функциональность дает неоспоримые преимущества по сравнению со вторым типом устройств – *неуправляемые*.

Неуправляемые устройства не имеют возможности интерактивного управления ими. Такие устройства настраиваются на определенный режим работы и затем поддерживают его. Например, отправка на сервер данных о местоположении с заданным интервалом.

Устройства без возможности управления можно разделить на два типа: *неконфигурируемые* и *конфигурируемые*. Первый тип устройств на сегодняшний день считается устаревшим и уже практически не производится. Вся логика поведения таких устройств реализована аппаратно, что значительно сужает спектр их использования несмотря на то, что в производительности они, в некоторых случаях, могут выигрывать у других типов устройств.

Конфигурируемые устройства можно разделить на три типа в зависимости от того, каким образом можно осуществлять конфигурацию.

Самый первый тип устройств позволял выполнять конфигурацию только посредством подсоединения устройства к персональному компьютеру через последовательный порт (COM-порт), что требовало также наличия специального программного обеспечения.

Следующий тип устройств уже использовал порт USB, а также имел возможность удаленного конфигурирования посредством отправки на устройство соответствующих команд посредством текстовых сообщений SMS.

Следующий тип устройств позволяет выполнять конфигурацию посредством интернет соединения через GPRS. Такой вариант является наиболее предпочтительным, так как позволяет избежать дополнительных издержек по времени в случае конфигурации посредством подсоединения устройства через кабель, и финансовых издержек при конфигурации устройства через SMS.

Самыми современными, и во многих случаях самыми нестабильными в работе, являются *управляемые* устройства с *разрабатываемой прошивкой*. В комплекте с такими устройствами поставляется среда для разработки программного обеспечения на специализированном (во многих случаях это собственная разработка компании) языке программирования.

Устройства такого типа комплектуются модулем мобильной связи, набором входных и выходных портов и входными аналоговыми портами. В данном случае под аналоговым портом понимается порт, аналоговое значение которого оцифровывается и доступно для использования в программном обеспечении устройства.

На таких устройствах изначально отсутствует какая-либо функциональность по умолчанию. То есть устройство можно запрограммировать на ту функциональность, которая необходима.

Данные тип устройств является многообещающим и перспективным. Однако, на сегодняшний день, все устройства такого типа, с которыми автор имел дело были нестабильны в работе и их использование в работающей системе было затруднено.

Уровень *взаимодействия с устройствами* может, как присутствовать как самостоятельный, либо быть интегрирован в уровень бизнес-логики. Для лучшего понимания необходимости наличия данного уровня рассмотрим структуру сообщений на примере двух устройств.

Общим для большинства форматов является следующая структура: <заголовок><разделитель><тело сообщения>. В заголовке обычно располагается различная служебная информация, например, идентификационный номер устройства (IMEI), версия программного обеспечения и так далее. Тело сообщения непосредственно содержит географические данные, скорость и направление движения, дату и время, различные флаги и так далее. Однако, несмотря на общую структуру сообщения в целом, далее форматы могут значительно различаться.

Рассмотрим устройство компании *Arknav*, модель X8. Сообщения данной модели имеют следующую структуру: <IMEI>,<код продукта>;<тип сообщения>,<тело сообщения>;<тип сообщения >,<тело сообщения>;<тип сообщения>,<тело сообщения>. Далее, для примера, рассмотрим сообщение, взятое из базы данных работающей системы:
«351856040005407,240101;1G,110509053245,A,2457.9141N,12126.3192E,3.1,288,1.7,00000001».

Далее рассмотрим устройство компании ATrack, модель AT1. Структура сообщений упомянутой модели устройств приведа в таблице 1. Для примера рассмотрим одно из сообщение данной модели, взятое из базы данных работающей системы.

«20131213143246,
20131213221539,20131213221540,103797676,1445261,81,2,
195719,990,1,0,0,0,,-11,2000».

Таблица 1

Структура сообщения устройства ATrack AT1

Заголовок					Тело сообщения
Префикс	Контрольная сумма	Длина	Номер сообщения	ID устройства	Данные
			Поля, которые входят в длину сообщения		
		Поля, по которым рассчитывается контрольная сумма			

Рассмотрев структуру сообщений на примере двух популярных устройств можно сделать вывод о том, что не смотря на то, что сообщения в целом несут

одинаковую смысловую нагрузку – существенно различаются в структуре. Приведенные различия затрудняют использование в системе устройств различных моделей и производителей, из чего можно сделать вывод о необходимости выделения отдельного уровня архитектуры, который будет инкапсулировать в себе логику по абстрагированию для вышестоящих уровней различий в устройствах.

Далее рассмотрим варианты архитектуры данного уровня, начиная с более примитивных и менее функциональных к более сложным и функциональным.

Первыми рассматриваются так называемые «пассивные» системы, в которых осуществляется одностороннее взаимодействие с устройствами. То есть устройства с некоторой периодичностью отправляют данные системе, система же в свою очередь никак устройствами не управляет.

В некоторых системах предусмотрено использование устройств только одной модели устройств. В таком случае нет необходимости в поддержке различных форматов сообщений. Подсистема сбора данных при таком варианте представляет собой небольшой программный модуль (рис. 1), который поддерживает соединение с устройствами, обеспечивает проверку данных на корректность и их сохранение в базу данных.

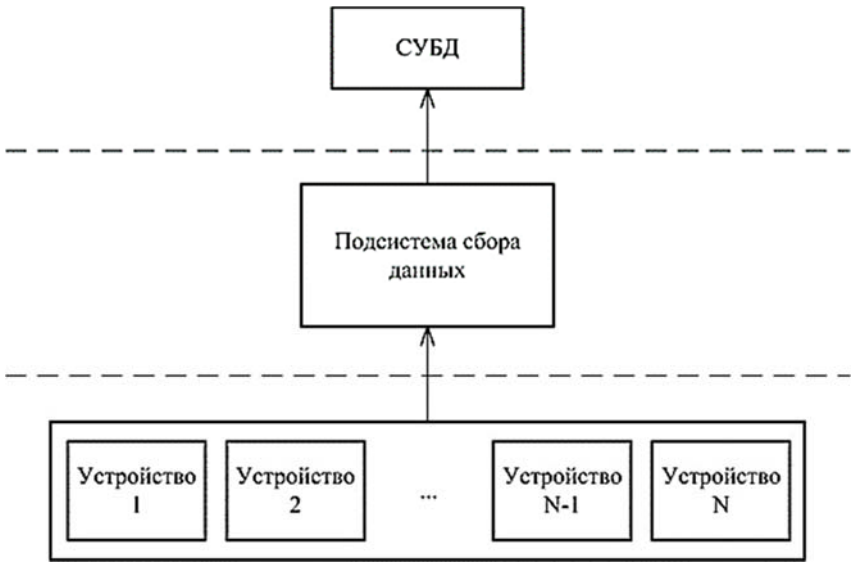

Рис. 1. Архитектура пассивной системы с одной моделью устройств

Данный вариант является самым примитивным и впоследствии, в большинстве случаев, требует значительной доработки либо полной его переработки.

Следующий вариант подразумевает поддержку системой устройств различных моделей устройств. При этом подсистема сбора данных устройств значительно усложняется. Появляется необходимость в поддержке различных форматов сообщений.

Имеются различные варианты внутренней структуры данного уровня. Возможен вариант, в котором подсистема сбора данных будет представлять собой монолитный компонент, для внесения дополнений в которой будет необходимо его перекомпилировать (либо перекомпановать, в зависимости от используемых средств разработки). В ином же случае данная подсистема представляет собой платформу в рамках которой работают независимые компоненты. Каждый такой компонент отвечает за поддержку отдельной модели устройств.

Такие независимо разрабатываемые компоненты можно назвать плагинами (plug-in). Благодаря им отпадает необходимость в перекомпиляции уровня при внесении дополнений. Каждый плагин может загружаться динамически, то есть «на лету».

Далее рассматриваются «активные» системы, в которых имеется, кроме простого получения данных от устройств, возможность интерактивного управления ими. Архитектура данного уровня значительно усложняется за счет того, что, кроме подсистемы сбора данных устройств, добавляется подсистема управления устройствами. То есть, при необходимости, в системе имеется возможность в реальном времени управлять поведением устройств.

Ядро системы посредством подсистемы управления устройствами отправляет им соответствующие команды (рис. 2). Сбор данных осуществляется попрежнему через подсистему сбора данных. Вместе, подсистема управления и подсистема сбора данных называются – подсистема взаимодействия с устройствами. Две подсистемы изображены на рисунке отдельными блоками для большей наглядности. При реализации они могут представлять собой единую подсистему.

Внутренняя структура подсистемы управления, также может представлять собой либо монолитный компонент, для внесения дополнений в который необходимо его перекомпилировать, либо представлять собой платформу для работы с подгружаемыми динамически плагинами, которые будут отвечать, каждый, за управления определенной модели устройств.

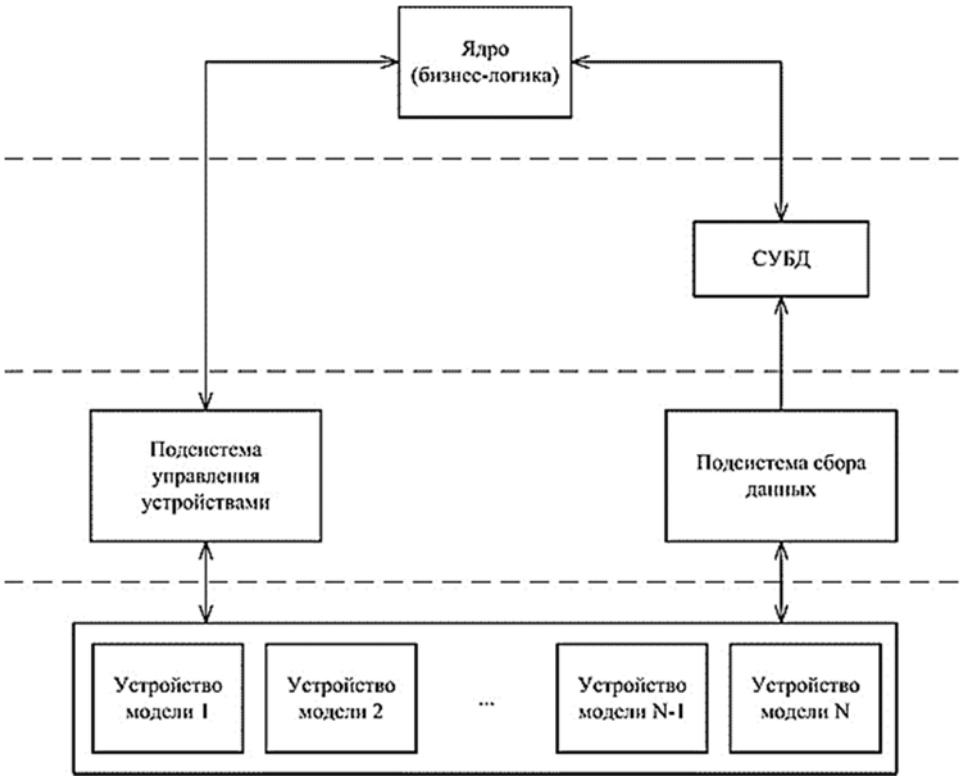

Рис. 2. Архитектура активной системы с различными моделями устройств

На *уровне данных*, в любой системе имеется необходимость в хранении определенной информации. В данном случае подсистема хранения данных вынесена в отдельный уровень, который, по сути, является внешним для системы в целом, однако имеются различные варианты его организации, которые могут оказать существенное влияние на производительность системы.

Наиболее простым вариантом является архитектура, в которой используется централизованная база данных. В такой базе данных может храниться как основные данные системы, так и картографические данные, в случае предоставления системой собственного сервиса карт. Для небольших систем это может быть тот же сервер, на котором расположены два соседних уровня.

Картографические данные в системе могут храниться в той же базе данных, что и основные данные системы. Огромное количество обращений к картографическим данным может значительно замедлить время отклика системы, за счет загрузки векторных данных и их последующей растеризации. В связи с этим, используется подсистема кеширования, которая позволяет сохранять результаты запросов (так называемые плитки, то есть фрагменты карты равного размера). Такая стратегия позволяет избежать повторной генерации плиток, что значительно уменьшает нагрузку на систему.

Однако помещенные в кеш данные имеют определенный срок хранения, по истечении которого они считаются устаревшими и удаляются из кеша, что позволяет избежать бесконечного разрастания объема хранимых в кеше данных.

Наиболее оптимальным вариантом является использование отдельного сервера для хранения картографических данных.

Достаточно популярным является вариант, при котором в системе используются картографические сервисы, предоставляемые другими компаниями. Такие сервисы предлагают, например, компании Google и Яндекс. Однако коммерческое использование таких сервисов является платным, что влечет за собой использование многими системами бесплатных аналогов.

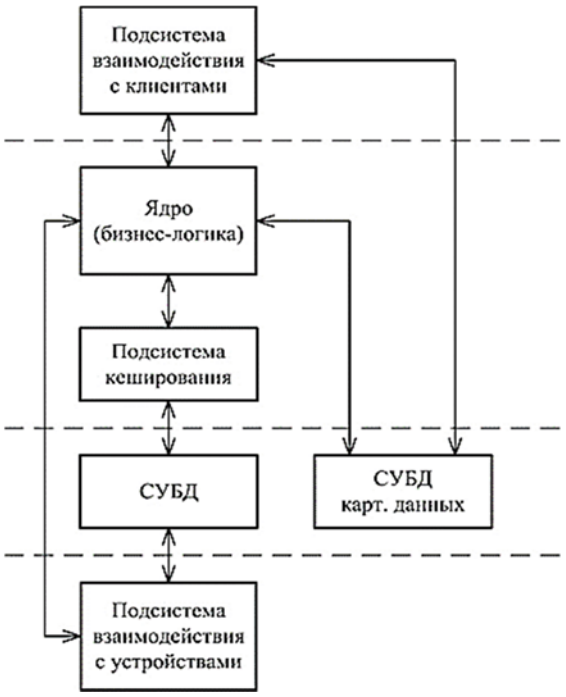

Рис. 3. Архитектура с отдельной БД для картографических данных

Уровень бизнес-логики является наиболее обширным в системе, вследствие чего и наиболее сложным. Здесь инкапсулируется вся прикладная логика обработки данных. То есть практически вся функциональность системы. Таким образом, рассмотрение данного уровня выходит за рамки статьи и не является ключевым для решения поставленной задачи.

Промежуточным между уровнем бизнес-логики и уровнем клиентов является *уровень взаимодействия с клиентами*. Данный уровень определяет каким образом и в каком виде клиенты получают данные из системы.

В наиболее простом и наименее гибком варианте данный уровень представляет собой программный модуль, который выполняет генерацию веб-контента в ответ на запрос, то есть формирует готовые веб-страницы либо их части (рис. 4).

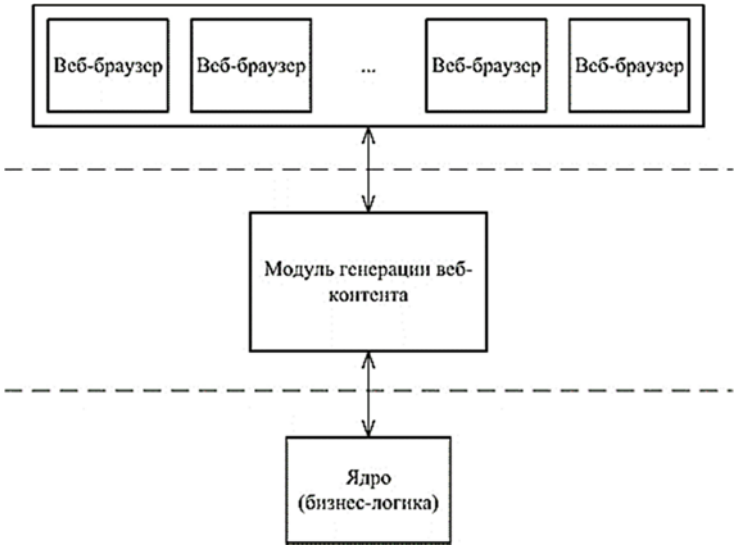

Рис. 4. Интегрированная архитектура

В качестве достоинств здесь можно привести относительную простоту реализации. Такой вариант является в некоторой степени кроссплатформенным, однако имеется ряд нюансов, связанных с адаптацией веб-контента под различные мобильные устройства. Также, в такой архитектуре отсутствует прямая возможность обращаться в систему из приложений, разработанных под определенные платформы.

Следующий вариант является доработкой предыдущего (рис. 5). Взаимодействие с клиентами может происходить как через модуль генерации веб-контента, так и посредством определенных точек входа (веб-служб). Точки входа

могут представлять собой как полноценные веб-службы с WSDL и XSD описанием посредством SOAP, так и более простые варианты, в которых используется свой протокол на основе языка XML (расширяемый язык разметки) либо JSON (текстовый формат обмена данными). Такой вариант позволяет разрабатывать нативные приложения для различных платформ.

При использовании в системе приложений для мобильных платформ возникает необходимость в уведомлении пользователей о наличии для них определенной новой информации (например, превышение автомобилем заданного порога скорости). Для этого используются push-уведомления. Для этого в системе добавляется подсистема уведомлений.

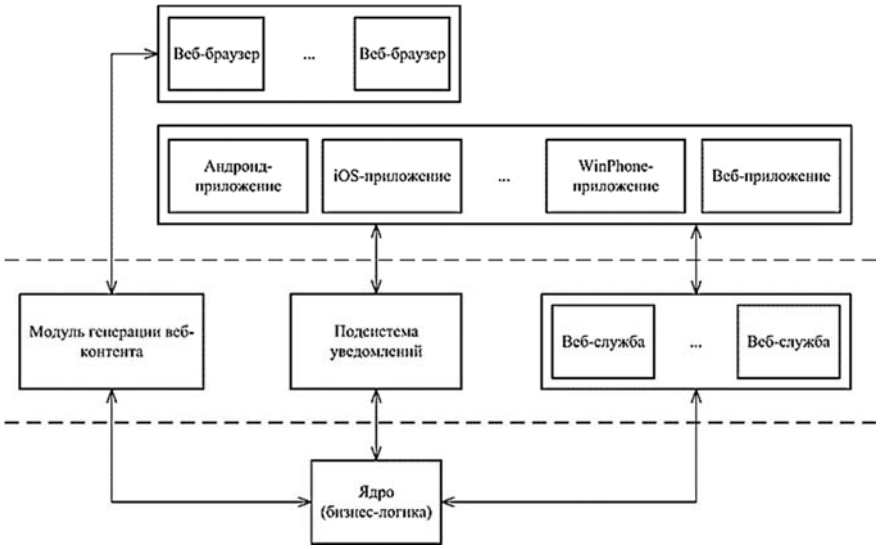

Рис. 5. Комбинированная архитектура

На самом высоком уровне располагается *уровень клиентов*. Структура уровня может иметь большое количество вариаций, что является следствием различных задач, которые ставятся для различных систем. На сегодняшний день можно выделить несколько наиболее часто применяющихся вариантов:

1. Приложение под одну из самых популярных платформ. В данном случае в качестве клиента в системе разработано программное средство под операционную систему Windows, которая безусловно занимает значительную долю рынка настольных программных платформ. Такой вариант, хоть и является приемлимым, все же не позволяет охватить как можно большую аудиторию пользователей.

2. Веб-приложение ориентированное на различные платформы. Из рассмотренных в данной работе систем можно привести, в качестве примера, систему TrackGPS. Любая платформа, на которой имеется доступ в интернет и существует возможность использовать веб-браузер позволяет использовать такой вариант реализации системы. Однако, следует упомянуть о следующих недостатках:

– сложность адаптации веб-содержания под различные устройства (обычные мониторы различных размеров, мобильные планшеты и телефоны с различными размерами экранов);

– неиспользование нативных возможностей платформ, которые во многих случаях могут более эффективными и удобными для использования.

3. Нативные приложения под различные платформы. Разработка приложений под все существующие платформы является конечно же недостижимой задачей, но разработка приложений под самые популярные платформы (например, Android, iOS, WinPhone, Windows, Mac OS) может охватить до 99% аудитории пользователей. Реализация данного варианта является довольно трудозатратной и, как следствие, потребует значительных финансовых вложений.

4. Нативные приложения под мобильные платформы и веб-приложение для настольных платформ. В данном варианте, для всех настольных платформ (Windows, Mac OS, Unix, Linux), разрабатывается одно веб-приложение. Для мобильных же платформ (Android, iOS, WinPhone) разрабатываются уже нативные приложения. Такой вариант является наиболее оптимальным, так как позволяет охватить как можно большую аудиторию пользователей, при этом, более оптимальным образом распределив ресурсы по сравнению с предыдущим.

В результате выполненного анализа можно отметить следующие результаты и выводы:

1. Описание общей архитектуры ССМТ в виде разделения структурных элементов системы на уровни, как самостоятельных подсистем.

2. Вывод о возможности отсутствия некоторых уровней в системе, либо интеграции одного уровня в другой.

3. Описание существующих на сегодняшний день методов спутникового мониторинга транспорта, используемых на каждом уровне архитектуры, с указанием на их недостатки.

Также, в связи с тем, что на теперешнем этапе стремительного развития информационных технологий и техники возможны частые изменения в требованиях к различным параметрам системы (таким как гибкость, масштабируемость, производительность) – при проектировании методов спутникового мониторинга транспорта необходимо это учитывать. В ином случае, недостатки в вышеуказанных параметрах выливаются в значительные временные затраты, и, как следствие – финансовые, а в некоторых случаях в невозможность выполнения новых требований.

В связи с вышеизложенным можно сделать вывод о целесообразности проведения исследования и проектирования архитектуры систем спутникового мониторинга транспорта, которая будет максимальным образом учитывать все недостатки существующих и использовать наиболее эффективные методы.

Список литературы

1. Savuriar M.A Low Cost Vehicle Monitoring System for Fixed Routes Using Global Positioning System (GPS) / M. Savuriar, N. Chandrasekharan, C. Venkatratnam, S. Ali // Proceedings of the International Conference on Electrical, Electronics, Computer Engineering and their Applications – Kuala Lumpur, 2014 – P. 97–103.

2. El-Medany W.A Cost Effective Real-Time Tracking System Prototype Using Integrated GPS/GPRS Module / W. El-Medany, A.Al-Omary, R. Al-Hakim, S. Al-Irhayim, M. Nusaif // Wireless and Mobile Communications (ICWMC) – 2010. – P. 20–25.

3. Khalifa A. Salim. Design and Implementation of Web-Based GPS-GPRS Vehicle Tracking System / Dr. Khalifa A. Salim, Ibrahim Mohammed Idrees // International Journal of Computer Science Engineering and Technology (IJCSET) – 2013. – Vol. 3. – №12 – P. 443–448.

4. Ramani R. Vehicle Tracking and Locking System Based on GSM and GPS / R. Ramani, S. Valarmathy, N. SuthanthiraVanitha, S. Selvaraju // I.J. Intelligent Systems and Applications – 2013. – №9. – P. 86–93.

5. Adnan I. GPS-based Vehicle Tracking System-on-Chip / Adnan I. Yaqzan, Issam W. Damaj, and Rached N. Zantout // International Journal of Electrical & Computer Sciences – 2010. – Vol. 10. – №4. – P. 7–12.