Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет имени Гагарина Ю.А.»

Институт прикладных информационных технологий и коммуникаций

Кафедра «Информационно-коммуникационные системы и программная инженерия»

Курсовой проект

по дисциплине «Современные технологии интернет-программирования»

РАЗРАБОТКА ОБЛАЧНОГО АНАЛИЗАТОРА ДАННЫХ АВТОМАТИЗИРОВАННЫХ СИСТЕМ ДОМАШНЕГО ПОЛЬЗОВАНИЯ

Выполнил: студент группы бПИНЖ31

Ребров Евгений Алексеевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись студента

Руководитель: доцент кафедры ИКСП

Торопова Ольга Анатольевна

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись руководителя

Саратов 2018

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc516618522)

[Глава 1. Обзор аналогов и выбор средства разработки анализатора 5](#_Toc516618523)

[1.1 Мониторинг поступающей информации с датчиков 5](#_Toc516618524)

[1.2 Обзор аналогов 7](#_Toc516618525)

[1.3 Разработка прототипа анализатора 17](#_Toc516618526)

[Глава 2. Разработка прототипа облачного анализатора данных автоматизированных систем домашнего пользования 22](#_Toc516618527)

[2.1 Требования к разрабатываемому облачному анализатору данных автоматизированных систем домашнего пользования 22](#_Toc516618528)

[2.2 Разработка алгоритма анализа данных 27](#_Toc516618529)

[2.2.1 Проектирование алгоритма анализа данных 27](#_Toc516618530)

[2.2.2 Реализация алгоритма анализа данных на языке PHP 29](#_Toc516618531)

[2.3 Разработка базы данных облачного анализатора данных автоматизированных систем домашнего пользования. 30](#_Toc516618532)

[2.3.1 Формирование сущностей и атрибутов, построение схемы модели базы данных. 30](#_Toc516618533)

[2.3.2 Создание базы данных для облачного анализатора с помощью средства phpMyAdmin 36](#_Toc516618534)

[2.4 Разработка клиентской части анализатора 38](#_Toc516618535)

[2.4.1 Разработка функции, формирующей график 38](#_Toc516618536)

[2.5 Подготовка данных для разрабатываемого алгоритма 40](#_Toc516618537)

[2.5.1 Регистратор данных 40](#_Toc516618538)

[2.5.2 Сбор данных для алгоритма и их занесение в БД анализатора 44](#_Toc516618539)

[ЗАКЛЮЧЕНИЕ 47](#_Toc516618540)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 48](#_Toc516618541)

# ВВЕДЕНИЕ

В настоящее время стали популярны автоматизированные системы домашнего пользования. Для их функционирования используются множество подсистем, одной из них является система датчиков. С ее помощью осуществляется мониторинг всевозможных характеристик каких-либо объектов или процессов. При этом возникает потребность в удаленном и постоянном доступе к данным, которые формирует система датчиков. А также эти данные нуждаются в дополнительной обработке. Для этих целей существуют специализированные системы, именуемые облачными анализаторами данных автоматизированных систем домашнего пользования.

Т. к. данная тема является актуальной, то в рамках курсовой работы была поставлена цель - разработать прототип облачного анализатора данных автоматизированных систем домашнего пользования.

Для того чтобы решить поставленную цель, были сформулированы следующие задачи:

* Провести обзор аналогов облачного анализатора данных для автоматизированных систем домашнего пользования;
* Осуществить выбор средства для разработки анализатора;
* Разработать алгоритм анализа данных;
* Выполнить разработку базы данных для анализатора;
* Разработать клиентскую часть анализатора;
* Подготовить пробные данные для разрабатываемого алгоритма.

# Глава 1. Обзор аналогов и выбор средства разработки анализатора

В данной главе будут рассмотрены принципы мониторинга поступающей информации с датчиков, будет проведен обзор аналогов разрабатываемого прототипа и будет выбрано программное обеспечение, с помощью которого будет проведена его разработка.

## Мониторинг поступающей информации с датчиков

*Мониторинг* (от латинского Monitor - предостерегающий) – специально организованное, систематическое наблюдение за состоянием объектов, явлений и процессов с целью их оценки, контроля или прогноза.

Система мониторинга, как правило, включает в себя средства для получения, хранения, обработки и вывода данных, поступающих извне. Разберем их подробнее.

Для получения данных извне средства мониторинга используют разнообразные решения, такие как SOAP,RESTили такое частное средство как SMS сообщения. Если средство мониторинга реализуется на физическом устройстве, а не в облаке, то у него появляется возможность непосредственного контакта с разнообразными датчикамии для связи с ними используют протоколы 1-Ware или беспроводной Wi-Fi.

Хранение данных осуществляется посредством популярных СУБД, таких как MySQL, PostgreSQL, SQLite или OracleDatabase. В частных случаях, когда не требуется долговременное хранение данных,применяются RRDбазы данных.

Обработка данных проводится с помощью написанных самими разработчиками функций, либо с помощью уже готовых фреймворков, таких как Hadoop,Spark, Storm и иных решений.

Вывод данных осуществляется посредством собственных средств разработчиков, либо готовых решений, например RRDtool, Dygraphs, Chart.js и т. д.

## Обзор аналогов

Чтобы определить функционал разрабатываемой системы, проведем анализ имеющихся на рынке аналогов. Анализатор данных представляет собой средство для сбора и анализа данных, поступающих с различных устройств. К средствам со схожим с ним функционалом можно отнести платформы для управления домашней автоматикой и системы мониторинга сетевых устройств.

Платформы для управления домашней автоматикой – это, иначе говоря, системы «Умный дом», - сочетают в себе не только функции управления всевозможными устройствами, имеющимися в доме, но и предоставляют возможность увидеть в реальном времени состояния тех или иных устройств и показания с различных датчиков, установленных в частных домах. Одними из популярных систем на данный момент являются MajorDoMo, DeviceHive и 1-М Умный дом. Критерием для выбора аналогов из числа платформ для управления домашней автоматикой послужила их частая цитируемость в отечественных обзорах данных систем и большое количество статей с примерами разработки систем «Умный дом» с использованием данных платформ. Остановимся на них подробнее.



Рисунок 1– Интерфейс отображения графиков в MajorDoMo

*MajorDoMo*(см. рис. 1) - представляет собой бесплатную и открытую российскую программную платформу для комплексного управления домашней автоматикой, а так же для информационной поддержки жизнедеятельности. Данная система может быть установлена практически на любой персональный компьютер (на платформе Windows и Linux) и совершенно не требовательна к ресурсам. Даже без привязки к оборудованию она может быть использована для организации персональной инфо-среды. Не обладает средствами для анализа и создания прогнозов по поступающей информации с датчиков. Из интересующих нас возможностей можно выделить:

* Кросс-платформенность (Windows/Linux);
* Бесплатная для личного или коммерческого использования;
* Поддержка различного оборудования;
* Количество подключаемых устройств ограничена возможностями системы, на которую установлена платформа.
* Мульти-язычный интерфейс (Русский/English);
* Веб-доступ с любого устройства;
* Веб-интерфейс с обновлением в реальном времени;
* Отображение разнообразной информации о контролируемой системе;
* Интеграция со сторонними веб-сайтами и сервисами;
* CloudSync — облачная синхронизация и простой доступ из любого места;
* Построена на веб-технологиях (PHP/JS/HTML5);
* Программирование с помощью PHP.

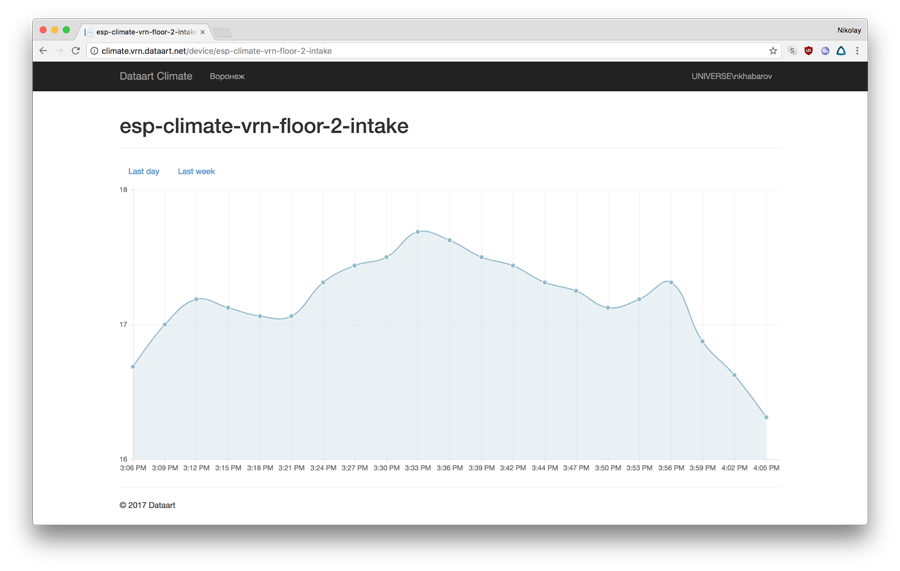
**

Рисунок 2- Пример работы DeviceHive

*DeviceHive* (см. рис. 2) – бесплатный инструмент, способный сделать частью Интернета вещей любое подключенное к нему устройство. Разработан компанией американской DataArt. Он обеспечивает коммуникационный уровень, программное обеспечение управления и много платформенные библиотеки для начальной загрузки и разработки умных устройств, домашней автоматизации, дистанционного зондирования, телеметрии, программного обеспечения для дистанционного управления и мониторинга и многое другое. Подключение встраиваемых Linux-систем с использованием Python или C++, протоколов Json. Разработка клиентских приложений с использованием библиотек HTML5/JavaScript, iOS и Android. Из его особенностей выделим:

* Подключение любого устройства через RESTAPI;
* Не ограниченное количество подключаемых устройств;
* Возможность подключить низкоуровневые устройства с поддержкой Wi-Fi;
* Легкая интеграция с любыми другими облаками и платформами;
* Дает основу для быстрого построения аналитики;
* Открытый исходный код.
* Может выводить с помощью сторонних средств отображения информацию, поступающую с различных устройств.

*1-M Core* – Открытое программное ядро 1-M Core отечественной разработки превращает Android-устройство, обычный роутер или другое Linux- или Windows- устройство в центральный контроллер Умного Дома. Ядро 1-M Core выполняет прием данных от датчиков, обработку сценариев, управление устройствами и другие функции системы Умного Дома.

Поддерживается радиосвязь с датчиками и устройствами фирм: Livolo, Noolite, RAEX, Telis, TKBHOME, GLANZEN, Chuango, Homesecur, Oregon, Nice, Visonic, AnMotors, CAME, Eazylift, EmilLux, FalconEye, Focus, Gant, Ginzzu, Hormann, Kelvin, Baisheng, TechLiCo, Feron, EmilLux, Expert, Fuhaoteидр. Количество подключаемых средств ограничено возможностями системы, на которую установлено данное ядро. Не обладает возможностями построения аналитики по поступающей с устройств информации. Отметим его особенности:

* Оповещения о событиях дома: вторжение, дым, потоп и т. д.
* Мониторинг и удаленное управление оборудованием при наличии сторонних средств отображения информации.
* Подключение облачных сервисов: прогноз погоды, геопозиционирование и др.
* Более эффективная, слаженная работа подсистем: отопления, полива, освещения и др.
* Ведение статистики работы устройств на облаке.

Системы мониторинга серверов предназначены для снятия и отображения метрических показателей с различных сетевых устройств. Они предоставляют мощный функционал для системных администраторов и значительно упрощают их работу. Порой на практике их применяют и в целях отображения показаний с датчиков, применяемых в системах домашнего пользования. Остановимся на наиболее используемых системах мониторинга:PRTG Network, Zabbix и Cacti. Критерием для выбора данных систем послужила их большая популярность в среде системного администрирования. Опишем приведенные системы мониторинга.

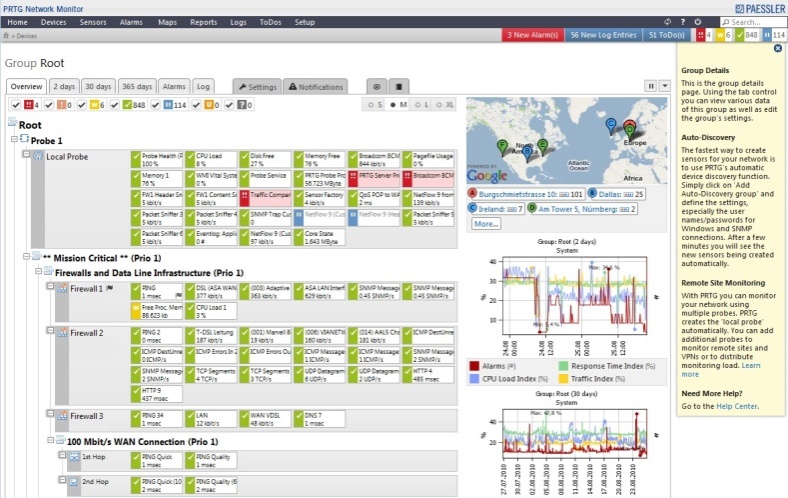


Рисунок 3- Рабочее окно PRTGNetwork

*PRTG Network* (см. рис. 3) – условно-бесплатная программа (пробный период — 30 дней) для бесплатной версии число подключаемых датчиков ограничено 30-ю, разработана в Германии и предназначена длямониторингаиспользования сети, работает в семействе операционных систем Windows. Из возможностей данной программы:

* Сбор информации о потоках данных, проходящих через конкретные устройства, с сохранением её в базе данных программы;
* Просмотр статистики в базе данных в виде графиков и таблиц;
* Просмотр результатов в режиме реального времени или за определенный промежуток времени в прошлом на разных устройствах;
* Построение аналитики по поступающей информации;
* Сбор данных о нагрузке на подсистемы памяти и процессора.

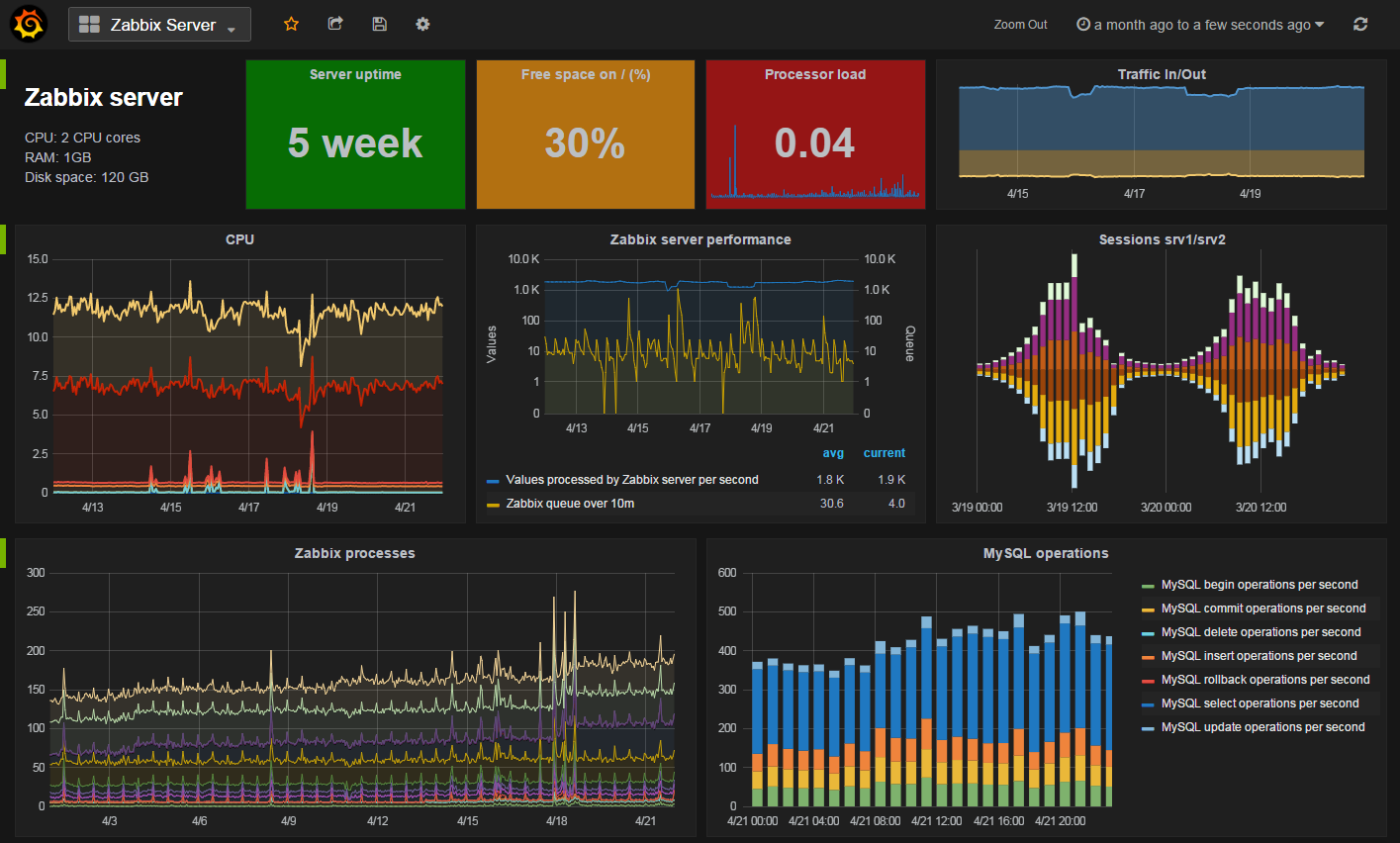


Рисунок 4- Рабочее окно Zabbix

*Zabbix* (см. рис. 4) – свободная система мониторинга и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования, разработана латвийской компанией Zabix LLC. Для хранения данных используется MySQL, PostgreSQL, SQLite или OracleDatabase, веб-интерфейс написан на PHP. Устанавливается на физический сервер, количество снимаемых метрик с контролируемой системы ограничивается возможностями сервера. Из возможностей данной программы:

* Распределённый мониторинг — до нескольких тысяч узлов. Конфигурация младших узлов полностью контролируется старшими узлами, находящимися на более высоком уровне иерархии;
* Сценарии на основе мониторинга;
* Централизованный мониторинг журналов;
* Веб-интерфейс для администрирования, настройки и отображения разнообразных показателей контролируемой системы;
* Отчётность и тенденции;
* Комплексная реакция на события;
* Расширение за счёт выполнения внешних скриптов;
* Гибкая система шаблонов и групп.

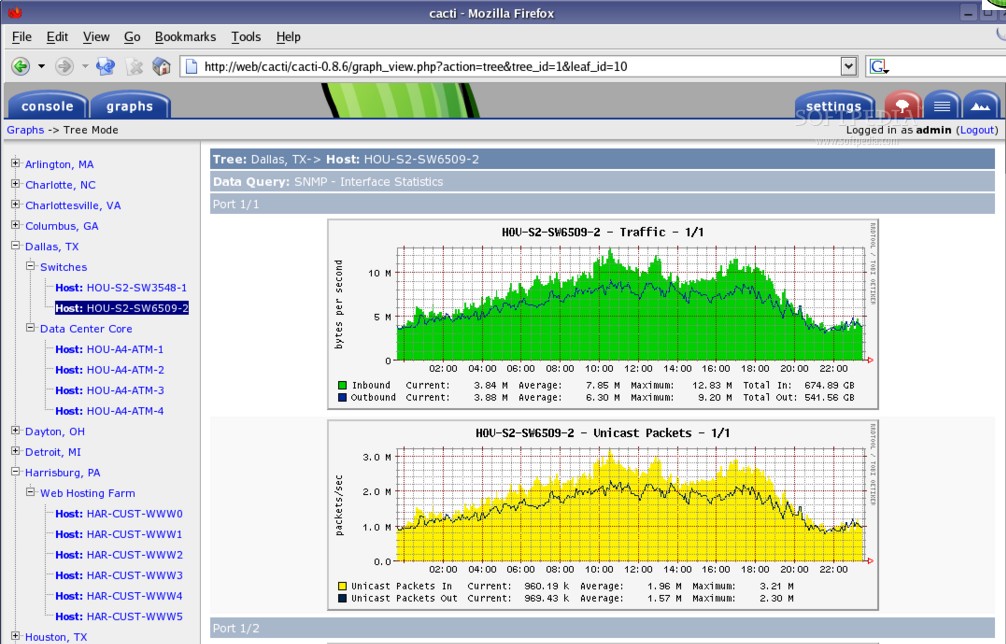


Рисунок 5- Рабочее окно Cacti

*Cacti* (см. рис. 5) – open-sourceвеб-приложение, разработанное в Америке и устанавливающееся на физический сервер. Оно позволяет строить графики при помощи RRDtool (набор утилит для работы с кольцевыми базами данных). Cacti собирает статистические данные за определённые временные интервалы и отображает их в графическом виде. Не обладает возможностью строить прогнозы по текущим показателям состояния системы. Количество снимаемых метрик с контролируемой системы ограничивается возможностями сервера.Преимущественно используются стандартные шаблоны для отображения статистики по загрузке процессора, выделению оперативной памяти, количеству запущенных процессов, использованию входящего/исходящего трафика. Из возможностей Cacti:

* Неограниченное количество элементов графика может быть определено для каждого графика;
* С помощью математических функций CDEF (позволяет применять различныематематическиефункциик графику для изменения выходных данных) можно управлять данными графа встроенными в RRDTool. Эти функции CDEF можно определить в Cacti и использовать глобально на каждом графике;
* Параметры RRD настраивают для пользователя возможность сбора данных по нестандартным временным интервалам при одновременном хранении различных данных;
* Содержит механизм ввода данных, который позволяет пользователям определять пользовательские сценарии, может быть использовано для сбора данных;
* Шаблоны графиков позволяют группировать общие графики по шаблону.

Для того чтобы провести оценку приведенных ранее приложений сперва выделим те качества, которые нас интересуют, а именно: стоимость приобретения, страну происхождения, место развертывания (ПК, облако, физический сервер), возможность строить графики из поступающей информации, возможность обрабатывать поступающие данные, возможность делать прогнозы и количество подключаемых датчиков. Приведем краткую характеристику приведенных выше аналогов в виде таблицы (см. табл. 1).

Таблица1- Краткая характеристика аналогов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **MajorDoMo** | **DeviceHive** | **1-МCore** | **PRTG Network** | **Zabbix** | **Cacti** |
| **Стоимость** | Бесплатно | Бесплатно | Бесплатно | Бесплатно на 30 дней | Бесплатно | Бесплатно |
| **Страна** | Россия | США | Россия | Германия | Латвия | США |
| **Расположение** | ПК | Облако | ПК | ПК/Физический сервер | Физический сервер | Физический сервер |
| **Наличие графика** | Присутствует | Отсутствует | Отсутствует | Присутствует | Присутствует | Присутствует |
| **Обработка данных** | Отсутствует | Присутствует | Отсутствует | Присутствует | Присутствует | Присутствует |
| **Наличие прогноза** | Отсутствует | Отсутствует | Отсутствует | Отсутствует | Присутствует | Отсутствует |
| **Количество датчиков** | Не ограничено | Не ограничено | Не ограничено | 30 | Не ограничено | Не ограничено |

Как видно из приведенной выше таблицы, платформы управления домашней автоматикой только отображают поступающую информацию с датчиков и отмечают значения, отклоняющиеся от заданных границ. Лишь системы мониторинга, такие как PRTG Network, Zabbix и Cacti, обладают достаточным функционалом, чтобы должным образом анализировать поступающую информацию и даже давать краткосрочные предсказания по значениям поступающих данных (Zabbix). Но их недостатками является то, что они разворачиваются не в облаке, а на физических серверах, следовательно, доступ к данным через сеть Интернетвозможен только после договоренности с провайдером, поставляющем интернет услуги. Критичным так же является и то, что они не являются продуктами отечественной разработки.

После проведенного обзора можно определить функционал разрабатываемого облачного анализатора данных. Он должен:

* Располагаться в облаке;
* Быть бесплатным или на его реализацию должно быть затрачено минимальное количество средств;
* Отображать разнообразную статистическую информацию, поступающую с датчиков;
* Проводить анализ поступающей информации;
* Строить прогнозы исходя из имеющейся информации;
* Предоставлять возможность подключать неограниченное количество датчиков.

Итак, мы определили принципы работы средств мониторинга информации, поступающей с датчиков, и выяснили характеристики, которыми будет обладать разрабатываемый анализатор.

## 1.3 Разработка прототипа анализатора

Перед началом разработки анализатора данных необходимо выбрать средство, с помощью которого оно будет разрабатываться. Исходя из того что нам требуется система, которая должна будет располагаться в облаке, то для разработки прототипа нам потребуется средство, которое будет эмулировать локальный сервер.

*Локальный сервер* **-** это специальная программа, которая устанавливается на локальный компьютер позволяющая веб-разработчикам разрабатывать сайты и веб-сервисы на локальном компьютере, без необходимости выхода в Интернет. Это даже не одна программа, а комплекс программ, выполняющих различные функции. Локальный сервер аналогичен настоящему реальному серверу, расположенному в Интернете.

На данный момент можно найти множество средств, но среди них выделяются Denver, OpenServerи XAMPP. Разберем их подробнее.

*Denver* - (от сокр. Д.н.w.р или ДНВР — джентльменский набор [Web](https://ru.wikipedia.org/wiki/Web)-разработчика) — набор дистрибутивов (локальный сервер WAMP) и программная оболочка, предназначенные для создания и отладки сайтов (веб-приложений, прочего динамического содержимого интернет-страниц) на локальном ПК (без необходимости подключения к сети Интернет) под управлением ОС [Windows](https://ru.wikipedia.org/wiki/Windows).

В 1999 году Дмитрий Котеров написал статью «Apache + Perl + PHP3 + MySQL для Windows 95/98: руководство по установке». Как он рассказывает в своём интервью, отвечая на письма читателей, он задумался над автоматизацией процесса. Так появился «джентльменский набор web-разработчика (д.н.w.р.)». Поначалу это был просто [zip](https://ru.wikipedia.org/wiki/ZIP)-архив размером около 1,5 Мбайта, с урезанными версиями Apache, Perl, PHP, MySQL и phpMyAdmin + скриптами на [Perl](https://ru.wikipedia.org/wiki/Perl) для запуска/остановки компонентов и работы с несколькими виртуальными хостами одновременно. Все компоненты системы уже были сконфигурированы так, чтобы работать только с относительными файловыми путями, вне зависимости от каталога, куда ставится пакет. 15 марта 2002 года вышла первая официальная версия «Denwer», в котором появился простой инсталлятор, который копировал файлы в указанное место и предлагал добавить ярлыки в папку автозагрузки [Windows](https://ru.wikipedia.org/wiki/Windows_NT). В дальнейшем он активно развивался и завоевал лидерские позиции в числе локальных серверов. В 2010 году его совершенствование прекратилось,и только совсем недавно началась дальнейшая работа над ним. Из его возможностей выделим:

* Мобильность, сервер можно развернуть из флэш накопителя;
* Простота, удобно для новичков;
* Русская локализация;
* Малый размер системы;
* Apache 2.2.22;
* PHP 5.3.13 + XDebug;
* MySQL 5.5;
* phpMyAdmin 3.5;

*OpenServer* - это портативная серверная платформа и программная среда, созданная специально для веб-разработчиков с учётом их рекомендаций и пожеланий.

Программный комплекс имеет богатый набор серверного программного обеспечения, удобный, многофункциональный продуманный интерфейс, обладает мощными возможностями по администрированию и настройке компонентов. Платформа широко используется с целью разработки, отладки и тестирования веб-проектов, а так же для предоставления веб-сервисов в локальных сетях.

Старт проекта OpenServer состоялся 1 августа 2010 года. Идея создать новую и более совершенную WAMP платформу родилась после многолетнего ожидания обновления другой известной WAMP сборки — Denwer.

Первые экспериментальные версии программы были написаны в сотрудничестве с талантливым программистом из Украины. Позднее разработка велась на языке PHP в среде быстрой разработки приложений (RAD) для Microsoft Windows под названием PHP Devel Studio. В дальнейшем, ввиду того, что возможности среды разработки были сильно ограничены и не позволяли проекту развиваться дальше, программа была переписана с нуля и начиная с версии 4.0 разработка ведётся на языке Delphi. Из числа его особенностей выделим:

* Мобильность;
* Простора и удобство пользования;
* Русская локализация;
* Большой вес дистрибутива;
* Наличие сразу нескольких версий компонентов ПО, входящих в сборку;
* Apache 2.2.34 / 2.4.29;
* MySQL 5.1.73 / 5.5.58 / 5.6.38 / 5.7.20;
* PHP 7.2.00; (PDFlib 9.1.1p3, MongoDB 1.3.4, Redis 3.1.5)
* PHPMyAdmin 4.7.3;

*XAMPP*— кроссплатформенная сборка веб-сервера (развитие [LAMP](https://ru.wikipedia.org/wiki/LAMP)), содержащая [Apache](https://ru.wikipedia.org/wiki/Apache_HTTP_Server), [MySQL](https://ru.wikipedia.org/wiki/MySQL), интерпретатор скриптов [PHP](https://ru.wikipedia.org/wiki/PHP), язык программирования [Perl](https://ru.wikipedia.org/wiki/Perl) и большое количество дополнительных библиотек, позволяющих запустить полноценный [веб-сервер](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80).XAMPP популярен за рубежом и не имеет русскую локализацию.

Изначально XAMPP создавался как инструмент для разработчиков, позволяя веб-дизайнерам и программистам тестировать свою работу, не используя Интернет. Для упрощения работы некоторые возможности и настройки безопасности отключены по умолчанию, и в целом XAMPP рекомендуется к использованию только в очень дружественном окружении. Однако XAMPP иногда используется и во всемирной паутине. Также программа поддерживает создание и управление базами данных [MySQL](https://ru.wikipedia.org/wiki/MySQL) и [SQLite](https://ru.wikipedia.org/wiki/SQLite). Из числа его характеристик выделим:

* Возможность установки на различные операционные системы;
* Наличие портативной облегченной версии;
* Apache 2.4.33;
* MySQL 5.1.41 (Community Server);
* PHP 7.2.5 / 7.1.17 / 7.0.30 / 5.6.36;
* phpMyAdmin 4.8.0.1;

Кратко отобразим вышеперечисленные характеристикипрограммных платформ в табл. 2.

Таблица 2 - Характерные особенности локальных серверов

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Denver** | **Open Server** | **XAMPP** |
| **Мобильность** | + | + | + |
| **Русская локализация** | + | + | - |
| **Удобство пользования** | + | + | - |
| **Версия Apache** | 2.2.22 | 2.4.29 | 2.4.33 |
| **Версия MySQL** | 5.5 | 5.7.20 | 5.1.41 |
| **Версия PHP** | 5.3.13 | 7.2.00 | 7.2.5 |
| **Версия phpMyAdmin** | 3.5 | 4.7.3 | 4.8.0.1 |

Как видно из приведенной выше таблицы по большинству показателей выделяется OpenServer благодаря наличию более новых версий программных компонентов. Но его недостатком является большой размер дистрибутива. Denver не значительно отличается по показателям от Open Serwer, и даже в плане занимаемой памяти и ресурсов системы превосходит его, т. к. в его сборке не имеются малоиспользуемые элементы, входящих в состав Denwer библиотек. Наличие в Denwer более старых версий программных компонентов может незначительно сказаться на разработке первого прототипа анализатора данных. Поэтому было принято решение выбрать платформу Denwer в качестве средства для разработки прототипа облачного анализатора данных автоматизированных систем домашнего пользования.

Итак, мы определили принципы работы средства мониторинга данных, провели обзор аналогов, выяснили характеристики, которыми будет обладать разрабатываемый прототип анализатора и выбрали программное обеспечение, с помощью которого будет проводиться разработка прототипа.

# Глава 2. Разработка прототипа облачного анализатора данных автоматизированных систем домашнего пользования

В данной главе будут рассмотрены требования к разрабатываемой системе, будет разработана база данных для прототипа анализатора, алгоритм, применяемый в нем, клиентская часть сервиса и будут подготовлены пробные данные для проверки работы прототипа.

## 2.1 Требования к разрабатываемому облачному анализатору данных автоматизированных систем домашнего пользования

Сформулируем требования к разрабатываемой системе.

В загородном доме планируется установить устройство, которое измеряет показатели температуры в различных точках системы отопления и будет отправлять измеренные показатели через интернет на специальный анализатор данных, обрабатывающий эти данные. Именно прототип этого анализатора данных и планируется разработать в ходе данной курсовой работы. Необходимо, чтобы с помощью разрабатываемого анализатора была возможность обрабатывать информацию, получаемую от устройства, получать результаты анализа и просматривать все данные, получаемые анализатором.

Облачный анализатор данных автоматизированных систем домашнего пользования представляет собой средство для сбора, агрегации и анализа данных, поступающих через сеть Интернет с различных датчиков. Он расположен на сервере и взаимодействует с пользователем через браузер или специальное приложение. В нем реализована возможность долговременного хранения информации, ее обработки, отображения в реальном времени поступающей информации с датчиков. Он способен постоянно анализировать информацию и с учетом вновь поступающей производить переоценку сделанных ранее выводов. Стоит отдельно отметить наличие особого алгоритма, который способен обнаруживать в потоке поступающих данных отклонения, не характерные для основной массы поступающей информации и именуемые далее «аномалиями». Так же этот алгоритм может идентифицировать такие аномалии из числа других аномалий по характерным «сигнатурам» – наличию, величине, количеству и продолжительности экстремумов в поступающей информации, а так же после возникновения первых отклонений от нормы в поступающей информации предсказывать определенные аномалии с вычисленной вероятностью их появления.

Функциональные требования:

* Получение и запись в БД данных датчиков, поступающих через сеть по протоколу HTTP;
* Добавление нового датчика в анализатор;
* Информация о датчиках хранится в отдельной таблице БД;
* Возможность идентифицировать конкретный датчик по данным, поступающим с него;
* Возможность создавать кластеры данных, поступающих из определенных датчиков;
* Информация, определяющая кластеры данных, хранится в отдельной таблице БД;
* Долговременное хранение в БД данных с датчиков;
* Хранит информацию об ошибках в отдельной таблице;
* Отображение всех имеющихся в БД данных;
* Отображение данных из БД за определенный срок;
* Отображение всех данных из БД одного датчика;
* Отображение из БД данных определенного датчика или группы датчиков за определенный срок;
* Отображение всех аномалий за весь период времени на графике;
* Отображение местоположений конкретной аномалии на графике;
* Информация о местоположениях конкретной аномалии на графике хранится в отдельной таблице;
* Информирование об обнаружении аномалии;
* Отображение вероятности идентификации новой аномалии при ее обнаружении как той или иной уже сохраненной;
* Автоматическое определение новой аномалии;
* Аномалия определяется только для определенных кластеров данных;
* Отображение подробной информации о конкретной аномалии;
* Отображение графика в выводе;
* Постоянное обновление данных в выводе;
* Отображение линии тренда на графике, рассчитанного для данных за определенный период времени;
* Отображение регрессии на графике, рассчитанного для данных за определенный период времени;
* Отображение доверительного интервала на графике, рассчитанного для данных за определенный период времени;
* Возможность предсказывать на ограниченный срок значения, поступающих с датчиков данных;
* Доступ к функционалу анализатора через авторизацию;
* В анализаторе имеется 2 группы пользователей: администраторы и клиенты;
* Хранение логинов и паролей клиентов и администраторов в отдельных таблицах БД;
* Разные клиенты не могут обратиться к одинаковым данным БД;
* Группа «Клиенты»:
  + Имеет возможность зарегистрироваться;
  + Может авторизоваться;
  + Может добавлять и удалять датчики, с которых поступает информация;
  + Может делать вышеуказанные выборки данных из БД;
  + Получать анализ данных;
  + Имеет возможность описывать обнаруживаемые аномалии;
  + Имеет возможность удалять сигнатуры аномалий;
* Группа «Администраторы»:
  + Может авторизоваться;
  + Имеет возможность создавать новых администраторов;
  + Может удалить администратора;
  + Может удалить клиента и привязанные к нему данные;
  + Может добавить нового клиента;
  + Имеет возможность просматривать записи из всех таблиц БД;
  + Имеет возможность удалять записи из всех таблиц БД;
  + Имеет возможность читать информацию об ошибках;
  + Имеет возможность добавлять свои сигнатуры аномалий;
  + Имеет возможность удалять сигнатуры аномалий;
  + Имеет возможность описывать обнаруживаемые аномалии;
  + Может вносить изменения в параметры работы алгоритмов анализа;
  + Может делать вышеуказанные выборки данных из БД;
  + Получает анализ данных;

Нефункциональные требования:

* Время отклика анализатора не более 20 с.;
* Анализатор должен быть отказоустойчивым;
* Данные в БД хранятся не более 5 лет;
* Анализатор должен предоставлять удобный и понятный пользовательский интерфейс;
* Анализатор предоставляет возможность для одновременной работы 20-ти пользователей.

## 2.2 Разработка алгоритма анализа данных

### 2.2.1 Проектирование алгоритма анализа данных

После анализа требований к разрабатываемому анализатору был спроектирован алгоритм анализа данных. Блок-схема алгоритма приведена на рис. 6.

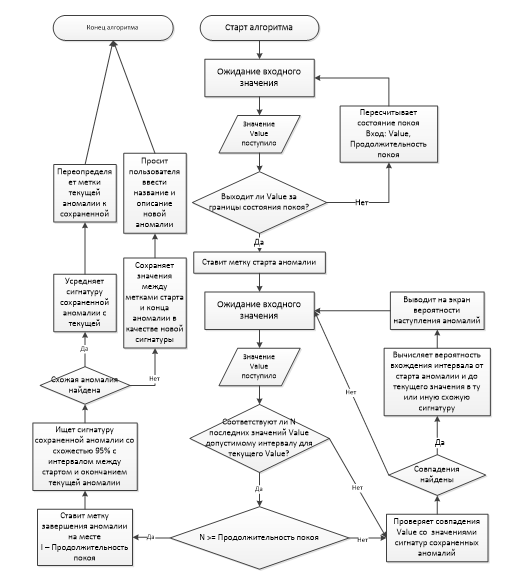


Рисунок 6- Блок-схема алгоритма анализа данных.

Данный алгоритм проводит поиск аномальных наборов значений в потоке поступающей информации от датчиков, классифицирует их по уникальным сигнатурам и сохраняет их параметры в БД анализатора.

Работа алгоритма начинается с ожидания поступающей величины, после получения определенных значений он проверяет, можно ли отнести поступившую величину к состоянию покоя (состояние покоя характеризуется незначительным отклонением значений от определенной величины на определенном промежутке последних полученных значений). Если полученная величина относится к состоянию покоя, то алгоритм продолжает ожидать входящие значения, иначе он ставит метку (запоминает момент начала аномалии)и продолжает считывать входящие значения, ожидая наступления состояния покоя, и параллельно начинает искать схожие сигнатуры сохраненных аномалий с точностью до 95%, подсчитывая вероятности наступления тех или иных аномалий. Когда обнаруживается состояние покоя, алгоритм ставит метку окончания аномалии и проводит окончательный поиск схожих аномалий. Если схожая аномалия обнаруживается, алгоритм пересчитывает сигнатуру сохраненной аномалии (её значения становятся средними арифметическими между обнаруженной сигнатурой и сохраненной). В случае, если алгоритм не обнаруживает схожую аномалию, он создает новую и просит пользователя ввести ее название и описание, далее он заносит в БД анализатора значения ее сигнатуры и относящиеся к ней метки, отражающие точки времени начала аномалии и ее конца.

### 2.2.2 Реализация алгоритма анализа данных на языке PHP

После завершения проектирования алгоритма перейдем к его разработке. На момент начала разработки было решено для демонстрации работы прототипа реализовать алгоритм так, чтобы он не ожидал входящих значений, а анализировал сразу все данные, находящиеся в БД анализатора, а так же было решено временно отказаться от того, чтобы алгоритм по обнаружению новой аномалии запрашивал у пользователя ее описание.

Алгоритм написан на языке PHP, который поддерживает выбранный ранее локальный сервер. Особенность использования данного языка заключаются в том, что все вычисления выполняются на сервере, а не на локальном компьютере пользователя, что не маловажно при анализе больших объемов данных, с которыми могут справиться не все потенциальные устройства, имеющие доступ к возможностям анализатора. Код разработанного алгоритма приведен в приложении 1.

Результат работы алгоритма представлен на рис. 8.

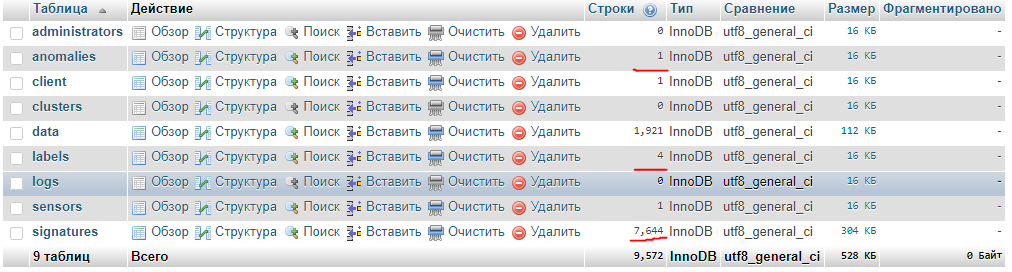


Рисунок 7 - Результат работы алгоритма

Как видно из приведенного выше изображения, разработанный алгоритм обнаружил одну аномалию, 4 метки и сигнатуру для найденной аномалии.

## 2.3 Разработка базы данных облачного анализатора данных автоматизированных систем домашнего пользования.

### 2.3.1 Формирование сущностей и атрибутов, построение схемы модели базы данных.

После разработки алгоритма анализа перейдем к созданию базы данных для анализатора. Согласно требованиям была разработана схема модели базы данных для облачного анализатора, она была построена с помощью средств MicrosoftVisio (см. рис. 9).

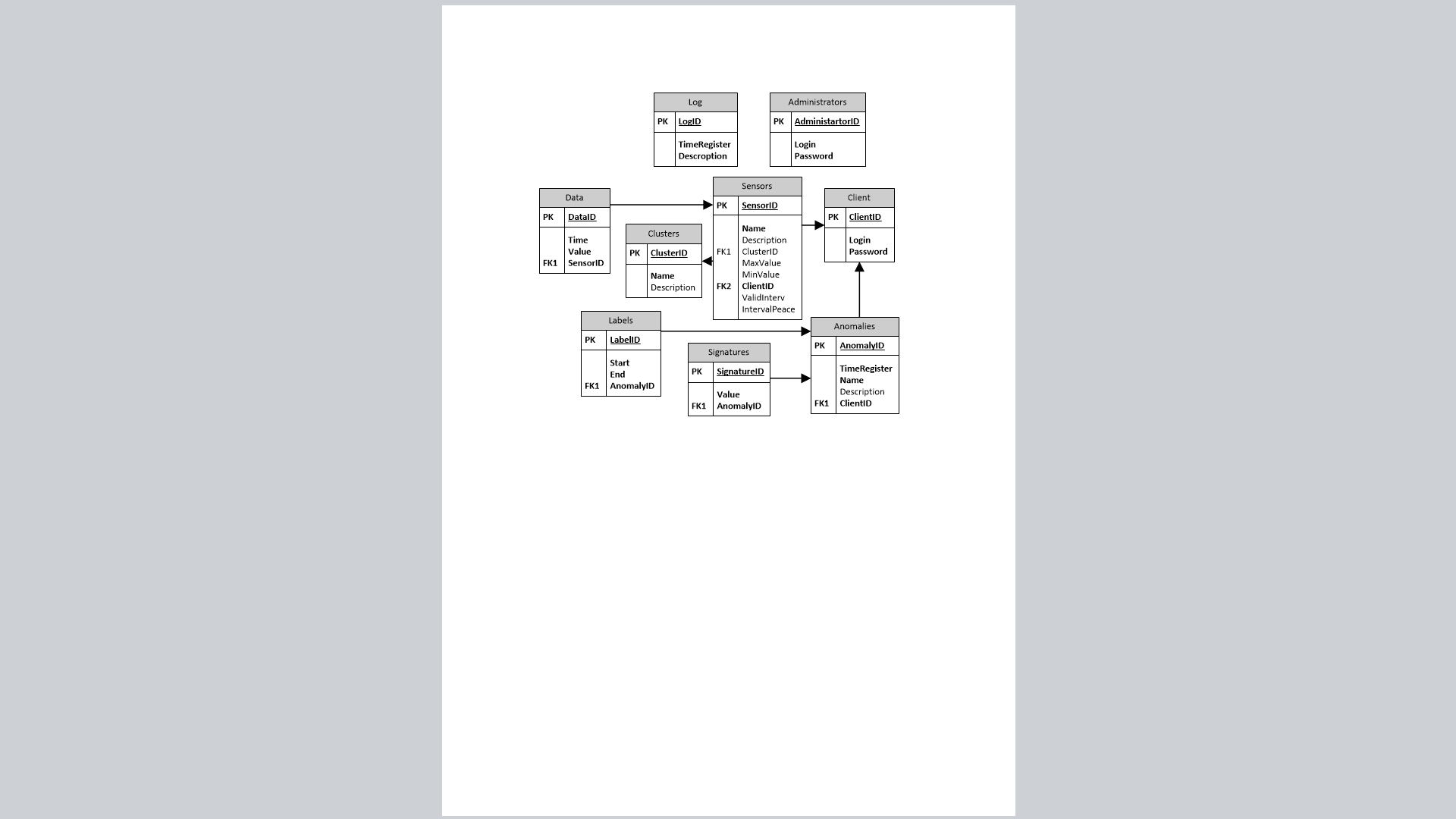


Рисунок 8- Схема модели базы данных

Согласно приведенной выше схеме и требованиям была составлена таблица сущностей и их атрибутов, которые будут применены в БД анализатора (см. табл.3).

Таблица 3- Таблица атрибутов и сущностей.

|  |  |  |  |
| --- | --- | --- | --- |
| **Название сущности** | **Название атрибута** | **Назначение атрибута** | **Описание атрибута** |
| Data | DataID | Идентификатор записи | Величина типа bigint. Первичный ключ. Номер записи. Не пустое. |
| Time | Время измерения | Поле типа datetime.Содержит в себе время измерения показаний. Не пустое. |
| Value | Измеренное значение | Поле типа float. Содержит в себе величину, измеренную датчиком. Не пустое. |
| SensorID | Идентификатор датчика | Поле типа int. Внешний ключ. Не пустое. |
| Sensors | SensorID | Идентификатор датчика | Величина типа int. Первичный ключ. Номер датчика. Не пустое. |
| Name | Имя датчика | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация о датчике | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| ClusterID | Принадлежность кластеру | Величина типа int. Внешний ключ. Может быть пустым. |
| MaxValue | Максимальная возможная величина, исходящая от датчика | Величина типа float. Может быть пустым. |
| MinValue | Минимальная возможная величина, исходящая от датчика | Величина типа float. Может быть пустым. |
| ClientID | Идентификатор клиента | Поле типа int. Внешний ключ. Не пустое. |
| ValidInterv | Определяет размер допустимого интервала | Поле типа float.  Может быть пустым. |
| IntervalPeace | Определяет продолжительность состояния покоя | Поле типа int  Может быть пустым. |
| Clusters | ClusterID | Идентификатор кластера | Величина типа int. Первичный ключ. Номер кластера. Не пустое. |
| Name | Имя кластера | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация о кластере | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| Client | ClientID | Идентификатор клиента | Величина типа int. Первичный ключ. Номер клиента. Не пустое. |
| Login | Логин клиента | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Password | Пароль клиента | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Administrators | AdministratorID | Идентификатор администратора | Величина типа int. Первичный ключ. Номер администратора. Не пустое. |
| Login | Логин администратора | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Password | Пароль администратора | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Logs | LogID | Идентификаторзаписи в журнале ошибки | Величина типа int. Первичный ключ. Номер записи в журнале. Не пустое. |
| TimeRegister | Время регистрации ошибки | Поле типа datetime.Содержит в себе время регистрации ошибки. Не пустое. |
| Description | Описание ошибки | Величина типа varchar. Содержит не более 1000 символов. Не пустое. |
| Anomalies | AnomalyID | Идентификатор аномалии | Величина типа int. Первичный ключ. Номер аномалии. Не пустое. |
| TimeRegister | Время регистрации аномалии | Поле типа datetime.Содержит в себе время регистрации аномалии. Не пустое. |
| Name | Название аномалии | Величина типа varchar. Содержит не более 50 символов. Не пустое. |
| Description | Информация об аномалии | Величина типа varchar. Содержит не более 1000 символов. Может быть пустым. |
| ClientID | Идентификатор клиента | Поле типа int. Внешний ключ. Не пустое. |
| Signatures | SignatureID | Идентификатор позиции всигнатуре | Величина типа bigint. Первичный ключ. Номер позиции в сигнатуре. Не пустое. |
| Value | Величина позиции сигнатуры | Величина типа float. Не пустое. |
| AnomalyID | Идентификатор аномалии | Поле типа int. Внешний ключ. Не пустое. |
| Labels | LabelID | Идентификатор метки | Величина типа int. Первичный ключ. Номер метки. Не пустое. |
| Start | Время начала метки | Поле типа datetime.Содержит в себе время начала метки. Не пустое. |
| End | Время конца метки | Поле типа datetime.Содержит в себе время окончания метки. Не пустое. |
| AnomalyID | Идентификатор аномалии | Поле типа int. Внешний ключ. Не пустое. |

Данная таблица понадобится при разработке БД для первого прототипа анализатора.

### 2.3.2 Создание базы данных для облачного анализатора с помощью средства phpMyAdmin

Когда модель БД построена и заполнена таблица атрибутов и сущностей, можно переходить непосредственно к созданию базы данных с помощью такого средства, встроенного в Denwer, как phpMyAdmin.

Во вкладке «Базы данных» создаем новую базу данных с именем «handler» (см. рис. 10).

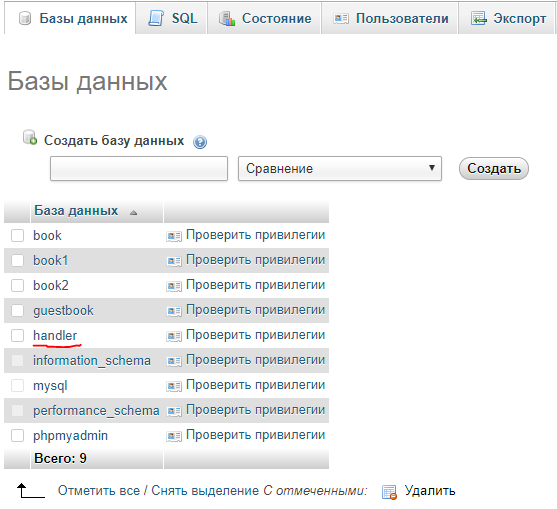


Рисунок 9- Создание БД "handler"

Когда база данных создана, переходим в ее меню и добавляем таблицы и поля к ним согласно таблице атрибутов и сущностей (см. рис. 11, 12).

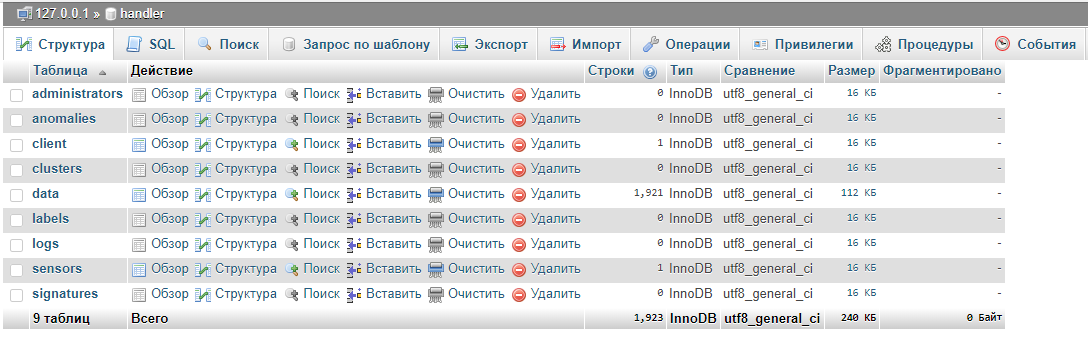


Рисунок 10- Таблицы, созданные в БД "handler"

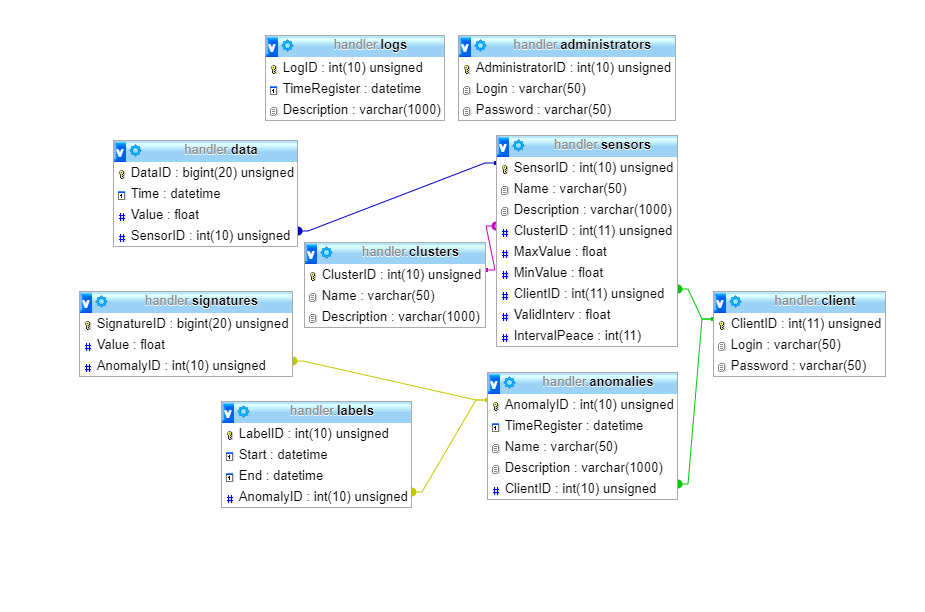


Рисунок 11- Отображение схемы БД во вкладке "дизайнер"

## 

## 2.4 Разработка клиентской части анализатора

### 2.4.1 Разработка функции, формирующей график

По завершении разработки БД для анализатора приступим к разработке клиентской части прототипа.

При создании прототипа анализатора было решено реализовать отображение данных из БД в виде криволинейного графика, предоставить возможность выбирать интервалы выборки данных для отображения на графике, а так же выводить в виде списка все метки обнаруженных аномалий, содержащие в себе информацию о названии аномалии, времени начала и времени ее окончания.

На основании разработанных требований была сформирована диаграмма вариантов использования для разрабатываемого прототипа (см. рис. 13).

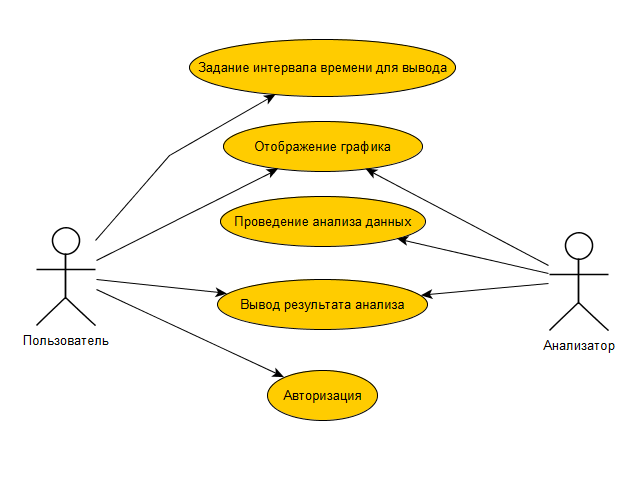


Рисунок 12 - Диаграмма вариантов использования для разрабатываемого прототипа

Для отображения данных в виде графика используется библиотека pChart. [pChart](http://pchart.sourceforge.net/)— это PHP-библиотека для создания графиков, гистограмм и диаграмм. В отличие от большинства схожих библиотек, pChart является полностью бесплатной (распространяется под GPL лицензией). Помимо класса, который генерирует изображения, есть классы для подготовки данных для построения графиков, а так же встроенный механизм кэширования. Отметим, что важной особенностью использования такой библиотеки является то, что ресурсы, затрачиваемые на построение графика, расходуются только на сервере, что дает возможность просматривать данные в виде графика на любых устройствах, имеющих доступ в Интернет.

Результат работы данной библиотеки изображен на рис. 14.

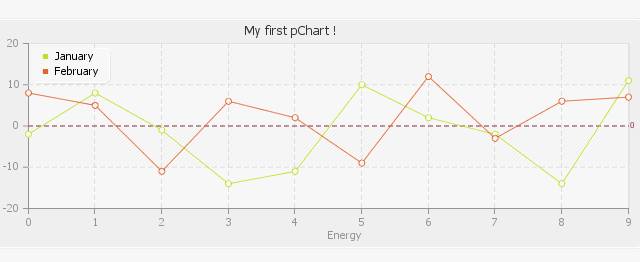


Рисунок 13 - График, построенный с использованием pChart

Далее был разработана диаграмма деятельности для варианта использования «Вывод диаграммы» (см. рис. 15).

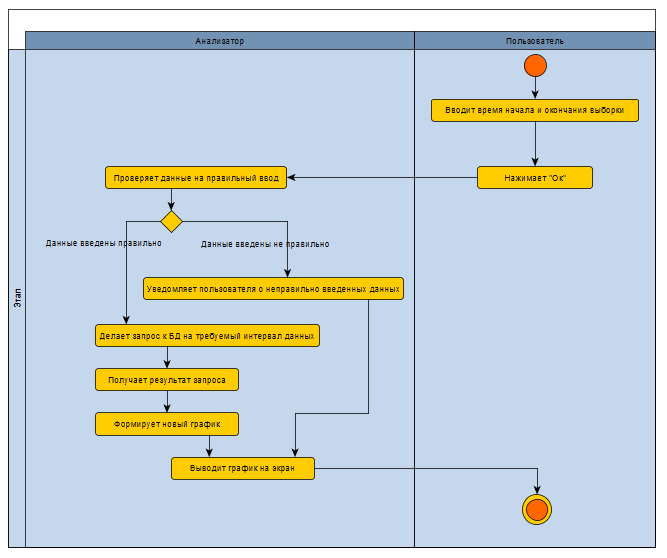


Рисунок 14 - Диаграмма деятельности для варианта использования "Вывод графика"

Затем была написана основная страница для клиентской части прототипа (см. рис. 16.).

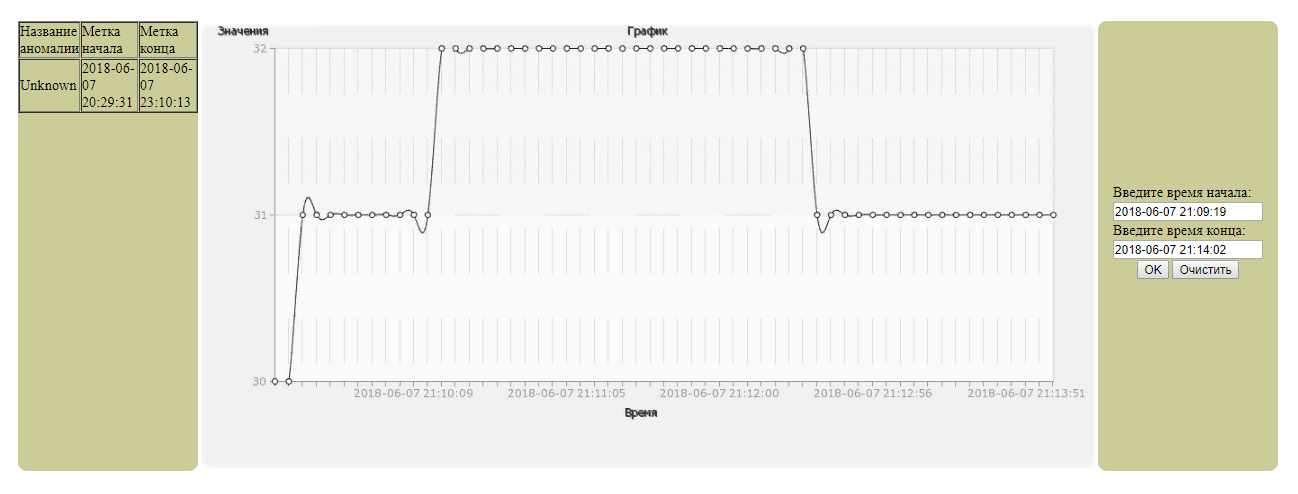


Рисунок 15 - Основная страница клиентской части разрабатываемого прототипа

## 

## 2.5 Подготовка данных для разрабатываемого алгоритма

Для проверки разрабатываемого алгоритма необходимо подготовить набор пробных данных. Было решено использовать для этого максимально приближенные к реальности значения. Поэтому для создания набора пробных данных использовалась упрощенная версия регистратора с датчиком температуры, который планируетсяиспользоватьдля получения значений температуры с различных участков системы отопления.

### 2.5.1 Регистратор данных

Приведем описание регистратора данных. Упрощенная версия регистратора данных состоит из контроллера Arduino Uno, платы Data Logging Shield и датчика температуры DallasDS18B20. Ниже приведем информацию о них.

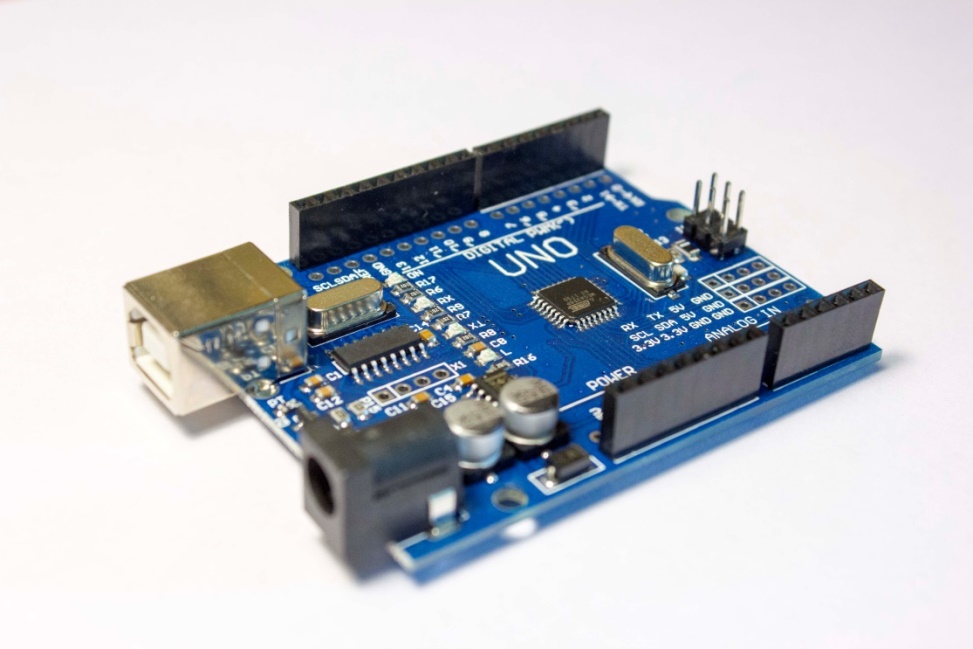


Рисунок 16 - Контроллер Arduino Uno

*Контроллер Arduino Uno* (см. рис. 17)- построен на микрочипеATmega328P. Платформа имеет 14 цифровых вход/выходов (6 из которых могут использоваться как выходы ШИМ), 6 аналоговых входов, кварцевый генератор 16 МГц, флэш-память – 32 Кб, оперативная память – 2 Кб,энергонезависимая память – 1 Кб, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание при помощи адаптера AC/DC или батареи. Ключевой особенностью данной платы является невысокая стоимость, уровень производительности достаточный для выполнения требуемых нам операций и значительное количество разнообразныхрасширяющих плат и датчиков.

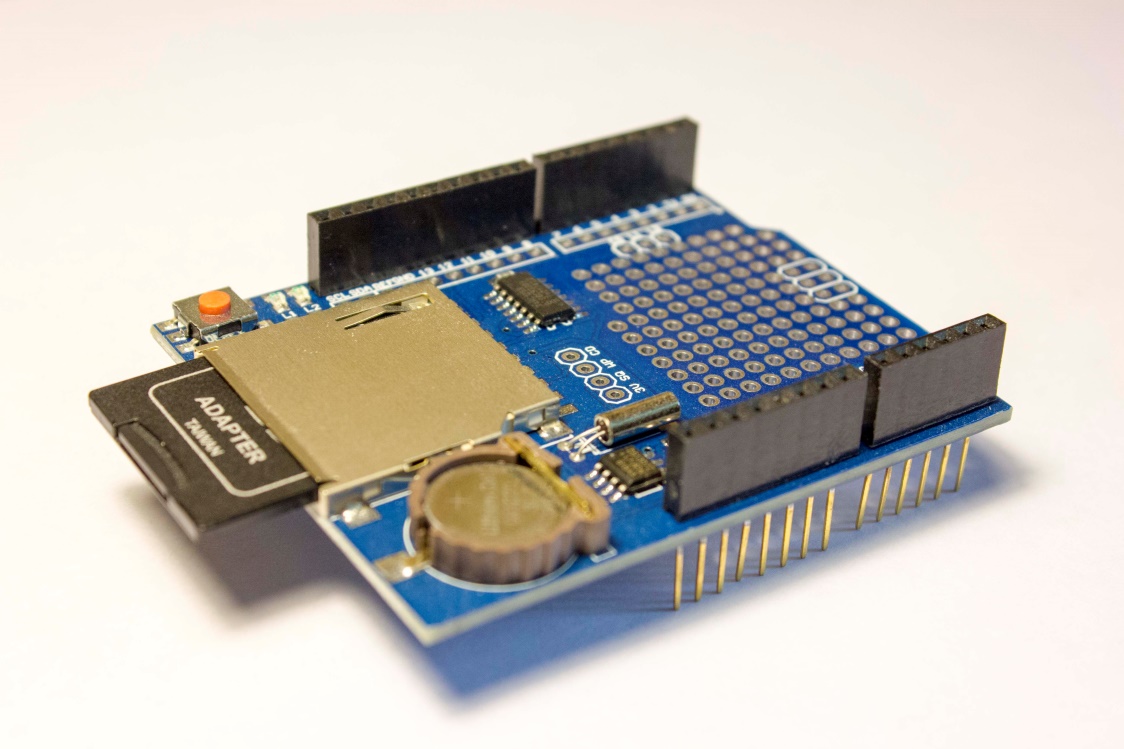


Рисунок 17 - Плата Data Logging Shield

*Плата Data Logging Shield* (см. рис. 18) - Модуль регистрации и хранения данных работает с картами памяти SD отформатированными в FAT16 или FAT32. Напряжение питания для SD карты 3,3 В,работу с SD картой обеспечивает микросхема RT9193-33. Модуль имеет в наличии часы реального времени (RTC), которые работают при отключении основного питания благодаря снимаемой батарее.

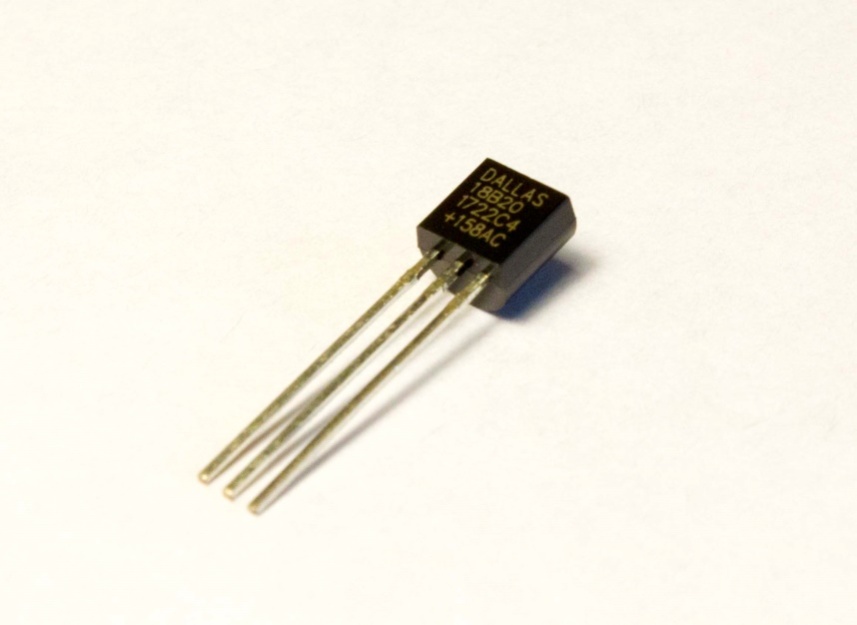


Рисунок 18 - Датчик температуры Dallas DS18B20

*Датчик температуры Dallas DS18B20* (см. рис. 19) - DS18B20 это цифровой измеритель температуры, с разрешением преобразования 9 - 12 разрядов и функцией тревожного сигнала контроля за температурой. Параметры контроля могут быть заданы пользователем и сохранены в энергонезависимой памяти датчика.

DS18B20 обменивается данными с микроконтроллером по однопроводной линии связи, используя протокол интерфейса 1-Wire.

Питание датчик может получать непосредственно от линии данных, без использования внешнего источника. В этом режиме питание датчика происходит от энергии, запасенной на паразитной емкости.

Диапазон измерения температуры составляет от -55 до +125 °C. Для диапазона от -10 до +85 °C погрешность не превышает 0,5 °C.

У каждой микросхемы DS18B20 есть уникальный серийный код длиной 64 разряда, который позволяет нескольким датчикам подключаться на одну общую линию связи. Т.е. через один порт микроконтроллера можно обмениваться данными с несколькими датчиками, распределенными на значительном расстоянии.

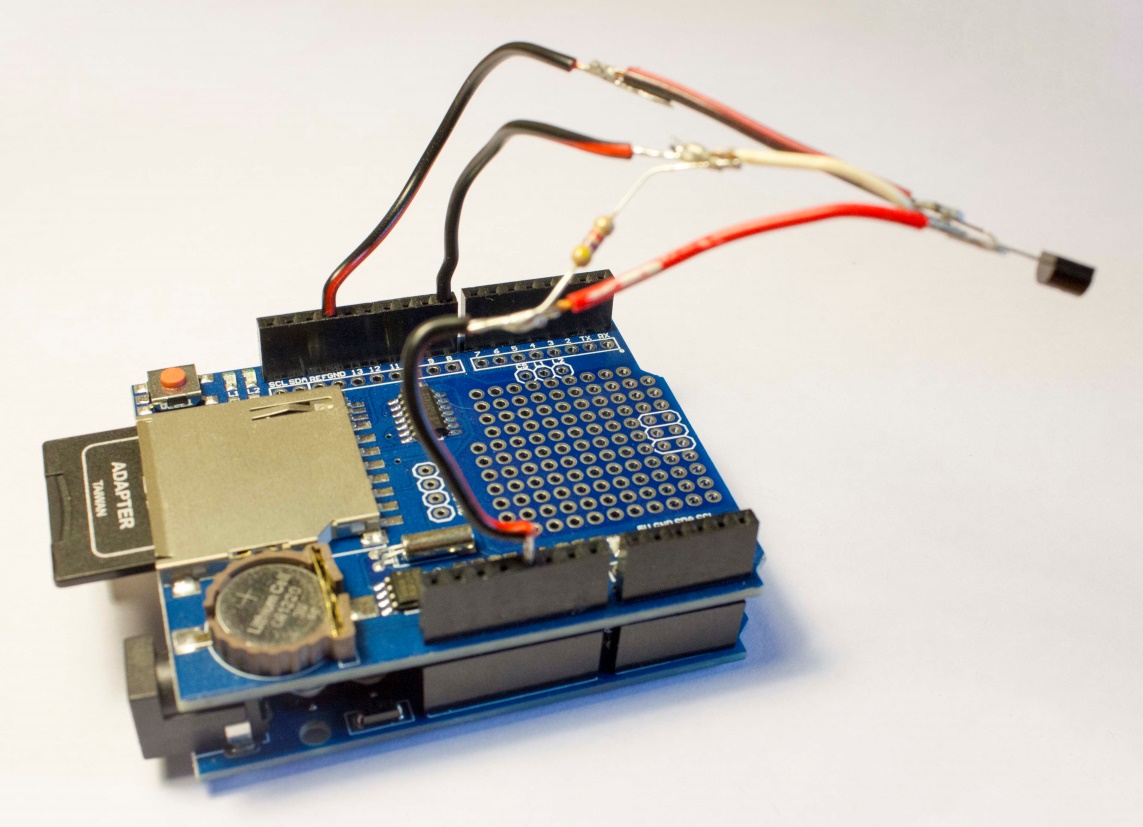


Рисунок 19 - Упрощенная версия регистратора данных

В сборе регистратор выглядит согласно рисунку 20. Отметим, что в полной версии регистратора планируется добавить дополнительно специальный модуль SIM800L, который реализует непосредственную отправку данных на анализатор через сеть GSM и свинцовый аккумулятор, выдающий напряжение 12В при силе тока 2А для обеспечения бесперебойной работы регистратора при отсутствии питания в сети.

Регистрация данных проводится следующим образом - каждые 5 секунд контроллер измеряет значения температуры в градусах Цельсия через датчик температуры и записывает их вместе с точным временем измерения построчно в файл DATALOG.txt, расположенный на SD карточке (см. рис. 21).

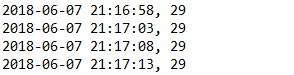


Рисунок 20 Пример записи данных в файл DATALOG.txt

### 2.5.2 Сбор данных для алгоритма и их занесение в БД анализатора

Проведем непосредственный сбор данных с регистратора и занесем их в базу данных анализатора.

Запись данных проводилась с источника тепла с динамически изменяющимися показателями теплоотдачи. По завершении записи данных у нас получился набор данных, состоящий из нескольких тысяч записей и занесенных в файл DATALOG.txt. Теперь необходимо перенести все записи БД анализатора.Первое, что необходимо сделать, это подготовить БД для внесения тестовых данных. Для этого создаемв таблице Client пробного пользователя с заданным именем и паролем (см. рис. 22).

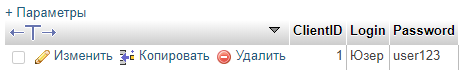


Рисунок 21 Пробный пользователь в таблице Client

Затем добавляем в таблицу Sensors данные о пробном датчике (см. рис. 23.).

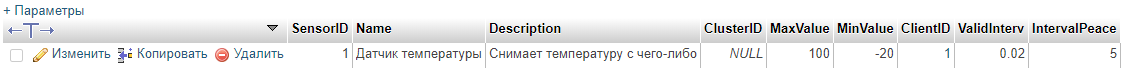


Рисунок 22- Пробный датчик в таблице Sensors

Для занесения в БД анализатора полученных измерений создадим скрипт на языке PHP, который сразу записывает все показания из файла DATALOG.txt. Этот скрипт построчно читает файл с данными и заносит их в таблицу Data. Текст разработанного скрипта приведен в приложении 2.

Когда скрипт готов при запущенном локальном сервере вызываем егов браузере через адресную строку. Результат его работы можно проверить с помощью интерфейса phpMyAdmin (см. рис. 24).

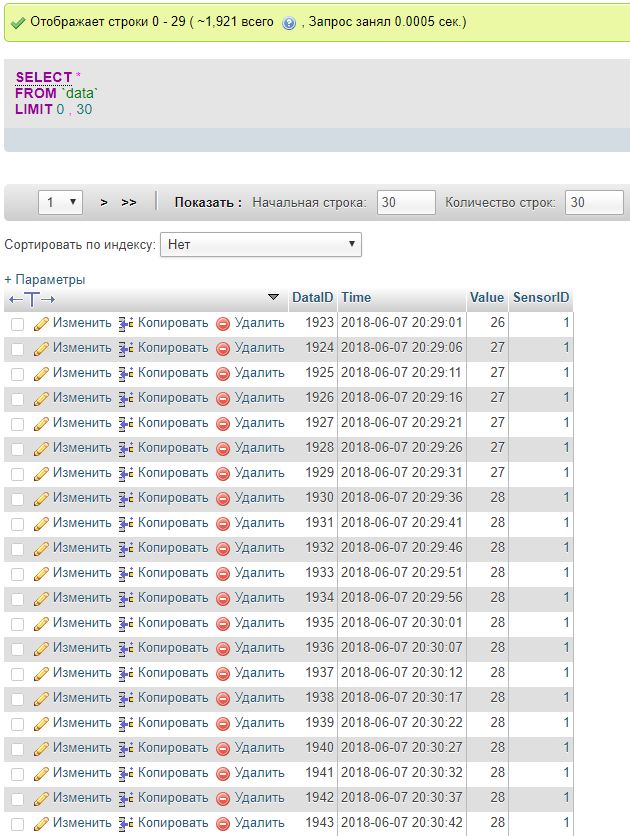


Рисунок 23 - Результат работы скрипта для переноса данных

В итоге был разработан прототип облачного анализатора данных автоматизированных систем домашнего пользования. В процессе его разработки мы рассмотрели требования для разрабатываемой системы, разработали алгоритм анализа данных, применяемый в ней, базу данных и клиентскую часть анализатора. На рисунке 25 представлен процесс работы алгоритма по поиску аномалий.

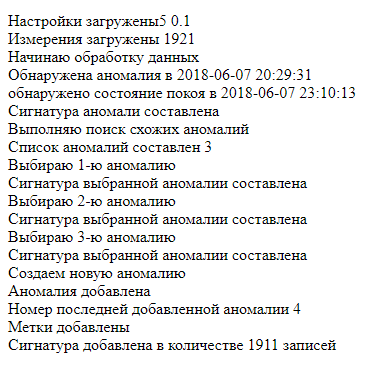


Рисунок 24 - Результат работы алгоритма по поиску аномалий

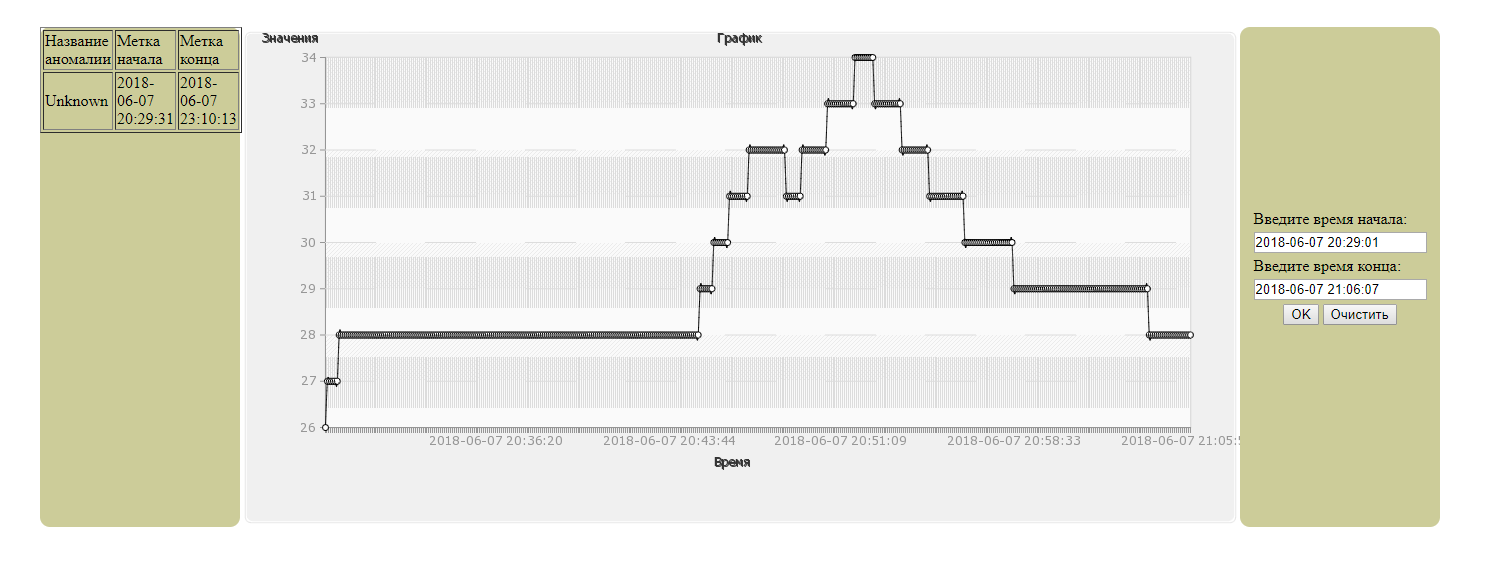
На рисунке 25 представлен результат вывода клиентской части анализатора.

Рисунок 25 - Результат вывода клиентской части анализатора

# ЗАКЛЮЧЕНИЕ

В ходе проекта разработан прототип облачного анализатора данных автоматизированных систем домашнего пользования, выполнены все поставленные задачи, а именно:

* Проведен обзор аналогов облачного анализатора данных для автоматизированных систем домашнего пользования.
* Осуществлен выбор средства для разработки анализатора.
* Разработан алгоритм анализа данных.
* Выполнена разработка базы данных для анализатора.
* Разработана клиентская часть анализатора.
* Подготовлены пробные данные для разрабатываемого алгоритма.

Разрабатываемое приложение будет актуально для тех пользователей, которые заинтересованы автоматизацией процесса удалённого сбора, анализа и отображения информации с различных датчиков, расположенных в частном домовладении.

В будущем планируется перенести анализатор на хостинг, оптимизировать алгоритм анализа, добавить другие методики анализа и доработать клиентский интерфейс.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. С., Монк. Программируем Arduino. Основы работы со скетчами [Текст]: пособие / Монк. С.. – СПб.: Питер, 2016. – 176 c.
2. У., Соммер. Программирование микроконтроллерных плат Arduino/Freeduino [Текст]: перевод с английского/ Соммер. У.. – СПб.: БХВ-Петербург, 2012. – 256 c.
3. Д., Котеров. PHP 5 [Текст]: пособие / Котеров. Д., Костарев. А.. – СПб.: БХВ-Петербург, 2016. – 1062 c.
4. Е., Р. Горяинова. Прикладные методы анализа статистических данных [Текст]: пособие / Р. Горяинова. Е., Р. Панков. А., Н. Платонов. Е.. – М.: Высшая школа экономики, 2012. – 312 c.
5. Л. Н. Полякова. Основы SQL [Текст]: пособие / Н. Полякова. Л.. – М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2007. – 224 c.
6. И. Симдянов. PHP 7. В подлиннике [Текст]: пособие / Симдянов. И., Котеров. Д.. – СПб.: БВХ-Петербург, 2016. – 1088 c.
7. Обзор локальных серверов для тестирования и разработки сайтов [Электронный ресурс]. – Режим доступа: http://www.maiboroda.ru/2012/03/03/obzor-lokalnyx-serverov-dlya-testirovaniya-i-razrabotki-sajtov/
8. Обзор систем мониторинга серверов. Заменяем munin на... [Электронный ресурс]. – Режим доступа : https://habr.com/post/331016/
9. Руководство и FAQ по phpMyAdmin. Материалы для веб-разработчиков [Электронный ресурс]. – Режим доступа : https://php-myadmin.ru/doc/
10. Online documentation [Электронный ресурс]. – Режим доступа: http://wiki.pchart.net/
11. IoT за копейки, или Что может DeviceHive [Электронный ресурс]. – Режим доступа: https://habr.com/company/dataart/blog/260115/
12. База знаний MajorDoMo [Электронный ресурс]. – Режим доступа: https://kb.smartliving.ru/?utm\_source=smartliving&utm\_medium=link&utm\_campaign=mainmenukb

# ПРИЛОЖЕНИЯ

Приложение 1. Код алгоритма анализа данных

<?php

$idSensors = 1;

$ValidInterv = 0.05;

$IntervalPeace = 10;

$Peace = 1;

$Start = ""; //метка начала аномалии

$End = ""; //метка конца аномалии

$User = 1;

$db = mysql\_connect("localhost", "root", "");

if ( $db == "" ) { echo " DB Connection error...\r\n"; exit(); }

mysql\_select\_db("handler",$db);

//снимаем настройки сенсора

$selectSettingsSensors =

"SELECT ValidInterv, IntervalPeace From sensors WHERE ClientID = '$User' AND SensorID = '$idSensors'";

$resultSensor = mysql\_query($selectSettingsSensors,$db);

if($Settings = mysql\_fetch\_row($resultSensor))

{

$ValidInterv = $Settings[0];

$IntervalPeace = $Settings[1];

echo "</br>Настройки загружены".$IntervalPeace." ".$ValidInterv;

}

//делаем выборку всех данных

$requiteData =

"SELECT data.Value, data.Time FROM data

INNER JOIN sensors ON sensors.SensorID = data.SensorID

WHERE sensors.ClientID = '$User' AND data.SensorID = '$idSensors'";

$result = mysql\_query($requiteData,$db);

$DataArray = array();

while ($row = mysql\_fetch\_array($result))

{

//echo "".$row['Value'];

$DataArray[] = array($row['Value'], $row['Time']);

}

echo "</br>Измерения загружены ".count($DataArray);

//главный цикл

echo "</br>Начинаю обработку данных";

for ($i = 0; $i<=count($DataArray); $i++)

{

if ($i > $IntervalPeace)

{

$Peace = CalculatePeace($DataArray[$i][0], $DataArray, $IntervalPeace, $Peace);

if (isPeace($DataArray[$i][0], Peace))

{

echo "</br>".$DataArray[$i][0]." В покое";

continue;

}

else

{

echo "</br>Обнаружена аномалия в ".$DataArray[$i][1];

$Start = $DataArray[$i][1];

for(; $i <= count($DataArray); $i++)

{

$Peace = CalculatePeace($DataArray[$i][0], $DataArray, $IntervalPeace, $Peace);

if (isPeace($DataArray[$i][0], $Peace))

{

echo "</br>обнаружено состояние покоя в ".$DataArray[$i - $IntervalPeace][1];

$End = $DataArray[$i - $IntervalPeace][1];

$CreateSignature =

"SELECT Value FROM data

INNER JOIN sensors ON sensors.SensorID = data.SensorID

WHERE (Time BETWEEN '$Start' AND '$End') AND sensors.ClientID = '$User'";

$result = mysql\_query($CreateSignature,$db);

$Signatura = array();

while($row = mysql\_fetch\_array($result))

{

$Signatura[] = $row;

}

echo "</br>Сигнатура аномалии составлена";

Comparator($Signatura, $User, $db, $Start, $End);

}

}

}

}

}

//пересчитывает интервал покоя

function CalculatePeace($State, $Data, $IntervalPeace, $Peace)

{

$ArrayPeace = array();

$SummPeace = 0;

for ($i = $State - $IntervalPeace; i<=$State; $i++)

{

if (!isPeace($Data[$i][0], $State))

return $Peace;

$SummPeace+=$Data[$i][0];

}

return $SummPeace/$IntervalPeace;

}

//Проверяет на принадлежность покою

function isPeace($Value, $State)

{

global $ValidInterv;

if($State \* ($ValidInterv + 1) <= $Value && $State \* (- $ValidInterv + 1) >= $Value)

return true;

else return false;

}

//проверяет на совпадение со всеми сигнатурами

function Comparator($A, $User, $db, $Start, $End)

{

echo "</br>Выполняю поиск схожих аномалий";

$selectAnomalies =

"SELECT AnomalyID FROM anomalies

WHERE ClientID = '$User'";

$result = mysql\_query($selectAnomalies,$db);

$AnomaliesArray = array();

while ($row = mysql\_fetch\_array($result))

{

$AnomaliesArray[] = $row['AnomalyID'];

}

echo "</br>Список аномалий составлен ".count($AnomaliesArray);

foreach ($AnomaliesArray as &$Anomaly)

{

echo "</br>Выбираю ".$Anomaly."-ю аномалию";

$selectSignatures =

"SELECT Value, SignatureID FROM signatures

WHERE AnomalyID = '$Anomaly'";

$result = mysql\_query($selectSignatures,$db);

$SignatureArray = array();

while ($row = mysql\_fetch\_array($result))

{

$SignatureArray[] = array($row['Value'], $row['SignatureID']);

}

echo "</br>Сигнатура выбранной аномалии составлена";

if (Compare($A, $SignatureArray)) //если совпадение найдено

{

echo "</br>Найдена схожая сигнатура ".$Anomaly."-й аномалии";

$newSignature = AVGator($A, $SignatureArray);

echo "</br>Производим перезапись сигнатуры".$Anomaly."-й аномалии";

if(count($A) <= count($SignatureArray)) // запись, если обнаруженная сигнатура меньше сохраненной

{

foreach ($newSignature as $row)

{

$AddSignature =

"UPDATE signatures

SET value = '$row[0]'

WHERE SignatureID = '$row[1]'";

$result = mysql\_query($AddSignature,$db);

}

}

else // запись если обнаруженная сигнатура больше сохраненной

{

$number=0;

for($i=0; $i <= count($SignatureArray); $i++)

{

$AddSignature =

"UPDATE signatures

SET value = '$newSignature[$i][0]'

WHERE SignatureID = '$newSignature[$i][1]'";

$result = mysql\_query($AddSignature,$db);

$number = $i;

}

for($i=$number; $i <= count($newSignature); $i++)

{

$AddSignature =

"INSERT INTO signatures(Value, AnomalyID)

VALUES ('$newSignature[$i][0]', '$Anomaly')";

$result = mysql\_query($AddSignature,$db);

}

}

$AddLabels =

"INSERT INTO labels(Start, End, AnomalyID)

VALUES ('$Start', '$End', '$Anomaly')";

$result = mysql\_query($AddLabels,$db);

echo "</br>Метка для ".$Anomaly."-й аномалии добавлена";

return;

}

}

/// если совпадений не найдено создаем новую аномалию

echo "</br>Создаем новую аномалию";

$AddAnomaly =

"INSERT INTO anomalies(TimeRegister, Name, Description, ClientID)

VALUES ('GETDATE()', 'Unknown', 'Unknown', '$User')";

$createAnomaly = mysql\_query($AddAnomaly,$db);

if ($createAnomaly) echo "</br>Аномалия добавлена";

else echo "</br>Аномалия не добавлена";

$GetIDLastAnomaly =

"SELECT MAX(AnomalyID) AS ID FROM anomalies";

$getAnomaly = mysql\_query($GetIDLastAnomaly,$db);

$row = mysql\_fetch\_array($getAnomaly);

$NewID = $row['ID'];

echo "</br>Номер последней добавленной аномалии ".$NewID;

$AddLabels =

"INSERT INTO labels(Start, End, AnomalyID)

VALUES ('$Start', '$End', '$NewID')";

$result = mysql\_query($AddLabels,$db);

echo "</br>Метки добавлены";

foreach($A as $row)

{

$AddSignature =

"INSERT INTO signatures(Value, AnomalyID)

VALUES ('$row[0]', '$NewID')";

$result = mysql\_query($AddSignature,$db);

}

echo "</br>Сигнатура добавлена в количестве ".count($A)." записей";

//работа завершена, аномалия добавлена

}

//сравнивает сигнатуры

function Compare($A, $B)

{

$Valid = 0.95;

$countCompare = 0;

if(count($A)<=count($B))

{

for ($i=0; $i<=count($A); $i++)

{

if (isPeace($B[$i][0], $A[$i][0]))

{

$countCompare++;

}

}

if($countCompare/count($B) >= $Valid)

return true;

else return false;

}

else

{

for ($i=0; $i<=count($B); $i++)

{

if (isPeace($A[$i][0], $B[$i][0]))

{

$countCompare++;

}

}

if($countCompare/count($A) >= $Valid)

return true;

else return false;

}

}

//возвращает усредненную сигнатуру

function AVGator($A, $B)

{

$C = array();

if(count($A)<=count($B))

{

for ($i=0; $i<=count($A); $i++)

{

$C[$i][0] = ($A[i][0]+$B[i][0])/2;

$C[$i][1] = $B[i][1];

}

}

else

{

for ($i=0; $i<=count($B); $i++)

{

$C[$i][0] = ($A[i][0]+$B[i][0])/2;

$C[$i][1] = $B[i][1];

}

}

return $C;

}

?>

Приложение 2. Код скрипта для экспорта данных в БД

<?php

$dbServer="localhost";

//название БД

$dbName="Handler";

//имя пользователя

$dbUser="root";

// пароль

$dbPassword="";

//идентификатор датчика

$SensorID = 1;

//источник данных

$filename = "DATALOG.txt";

/////////////////////////////////////////////////////////

// подключаемся в БД

$checkconnect= @mysql\_connect($dbServer,$dbUser,$dbPassword);

if(!$checkconnect)

{

echo"<P>Подключение прошло неудачно.".mysql\_error()."</P>";

exit();

}

else echo"<p>Подключено к серверу</p>";

//выбираем БД

if(!@mysql\_select\_db($dbName))

{

echo"<P>Доступ к БД невозможен. ".mysql\_error()."</P>";

exit();

}

else echo"<p>Подключено к БД</p>";

// настройка кодировки соединения

mysql\_query("SET NAMES 'utf8\_general\_ci'");

/////////////////////////////////////////////////////////

//подключаемся к файлу

$file = fopen($filename, 'r');

if ($file)

{

echo "<p>Файл открыт.</p>";

//записываем построчно в БД

while(!feof($file))

{

$Data = preg\_split("/(, )/", fgets($file, 999));

if(count($Data) != 2) exit();

$query = "INSERT INTO data(Time, Value, SensorID)

VALUES ('$Data[0]', '$Data[1]', '$SensorID')";

$load = mysql\_query($query);

if (!$load)

{

echo "<p>Ошибка добавления данных. ".mysql\_error()."</p>";

return;

}

}

echo "<p>Загрузка завершена.</p>";

}

else echo"<p>Ошибка открытия файла</p>";

fclose($file);

?>