Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
ГОУ ВПО
РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
(РГГМУ)

Допущен к защите,
заведующий кафедрой
докт. физ.-мат. наук, проф. А. Д. Кузнецов

Кафедра
экспериментальной физики
атмосферы

ДИПЛОМНЫЙ ПРОЕКТ

Микропроцессорная система автоматизации проведения лабораторных экспериментов

Выполнил
М.Ю. Дроздов, гр. И-559
Руководитель
канд. физ. мат. наук, доцент В.В.Чукин

Санкт-Петербург 2012

Содержание

Сокращения

ЭВМ – электронная вычислительная машина

ПЭВМ – персональная электронная вычислительная машина

АЦП – аналого-цифровой преобразователь

ЦАП – цифро-аналоговый преобразователь

УВХ – устройство выборки-хранения

В – вольт

мВ – милливольт

мкВ – микровольт

К – температура в Кельвинах

С – температура в Цельсиях

дБ – децибел

мкм – микрометр

КПК – карманный персональный компьютер

ТКС – температурный коэфицент сопротивления

TCP – Transmission Control Protocol (протокол управления передачей)

IP – Internet Protocol (межсетевой протокол)

DNS – Domain Name System (система доменных имён)

HTTP – HyperText Transfer Protocol (протокол передачи гипертекста)

WWW – World Wide Web (всемирная паутина)

FTP – File Transfer Protocol (протокол передачи файлов)

SMTP – Simple Mail Transfer Protocol (простой протокол передачи почты)

SSH – Secure SHell (безопасная оболочка)

СУБД – Систему Управления Базами Данных

БД – База Данных

CSS – Cascading Style Sheets (каскадные таблицы стилей)

HTML – HyperText Markup Language (язык разметки гипертекста)

Введение

Измерительная техника прошла путь от «показывающих», аналоговых, самопишущих, автоматических и цифровых приборов к информационно-измерительным системам.

Конец XIX века характеризуется первыми успехами в радиосвязи и электронике. Ее развитие привело к необходимости создания средств измерительной техники нового типа, реагирующей на малые входные сигналы, высокие частоты и высокоомные входы. В этих новых средствах измерительной техники использовались радиоэлектронные компоненты.

Таким образом, расширение номенклатуры и качественных показателей средств измерительной техники непрерывно связано с достижениями радиоэлектроники. Одним из современных направлений развития измерительной техники, базирующихся на радиоэлектронике, являются цифровые приборы с дискретной формой представления информации. Такая форма оказалась удобной для преобразования, передачи, хранения и обработки информации.

Широкие возможности открылись перед измерительной техникой после появления микропроцессоров и ЭВМ. Благодаря им значительно расширились области применения измерительной техники, улучшились их технические характеристики, повысились надежность и быстродействие, открылись пути реализации задач, которые ранее не могли быть решены. По широте и эффективности применения микропроцессоров одно из первых мест занимает измерительная техника, причем все более широко применяются микропроцессоры в системах управления.

В настоящее время любая автоматическая измерительная система не может быть представлена без использования вычислительной техники. А так

как современная вычислительная техника работает с цифровыми сигналами, то важнейшей частью современной измерительной системы является преобразование в цифровую форму аналоговых сигналов, получаемых с различных измерительных систем, и последующее представление результатов измерений пользователям в удобной форме.

В данной дипломной работе рассматриваются вопросы автоматизации процесса и визуализации результатов измерений при проведении лабораторных исследований с помощью современных информационных технологий.

Цель дипломной работы - создание автоматизированной измерительной системы. Система должна осуществлять измерения в течение длительного периода времени с высокой частотой дискретизации, а полученные данные должны быть доступны в любой момент времени, в любом месте, где есть персональный компьютер и доступ к сети Internet.

Для достижения поставленной цели, необходимо решить следующие задачи:

1) создать систему автоматической регистрации данных;
2) создать систему обработки данных измерений;
3) создать систему хранения данных измерений;
4) создать систему представления данных пользователю.

В первой главе рассмотрены цифровые измерительные системы, их назначение, методы аналого-цифрового преобразования, а также форматы передачи цифровой информации.

Вторая глава посвящена обзору методов хранения и представления данных измерений с использованием современных информационных технологий, таких как HTML, CSS, AJAX, XML, jqPlot.

В третьей главе представлено описание разрабатываемой системы автоматизации проведения лабораторных измерений, в частности, приводится информация о разработанном программном обеспечении.

Результаты тестовой эксплуатации разработанной системы представлены в четвертой главе.

1 Цифровые измерительные системы

1.1 Назначение измерительных систем

Измерительная система - совокупность функционально объединенных мер, измерительных приборов, измерительных преобразователей, ЭВМ и других технических средств, размещенных в разных точках контролируемого пространства с целью измерений одной или нескольких физических величин, свойственных этому пространству.

Измерительная система предназначена для выработки сигналов измерительной информации в форме, удобной для автоматической обработки, передачи и/или использования в автоматических системах управления.

В зависимости от назначения измерительные системы подразделяются на: измерительные информационные, измерительные контролирующие, измерительные управляющие и др.

Автоматическая измерительная система - измерительная система, снабженная средствами автоматического получения и обработки измерительной информации.

Так как современные ЭВМ работают с цифровыми сигналами, то далее следует рассмотреть методы преобразования аналогового сигнала в цифровой.

1.2 Методы аналого-цифрового преобразования

Аналого-цифровой преобразователь (АЦП) — устройство, преобразующее входной аналоговый сигнал в дискретный код (цифровой

сигнал). Обратное преобразование осуществляется при помощи ЦАП (цифро-аналогового преобразователя).

Аналого-цифровое преобразование – это процесс преобразования входной физической величины в ее числовое представление. Аналого-цифровой преобразователь – устройство, выполняющее такое преобразование. Формально, входной величиной АЦП может быть любая физическая величина – напряжение, ток, сопротивление, емкость, частота следования импульсов, угол поворота вала и тому подобное.

Как правило, АЦП— электронное устройство, преобразующее напряжение в двоичный цифровой код. Тем не менее, некоторые не электронные устройства с цифровым выходом, следует также относить к АЦП, например, некоторые типы преобразователей угол-код. Простейшим одноразрядным двоичным АЦП является компаратор.

Компаратор (аналоговых сигналов англ. comparator— сравнивающее устройство)— электронная схема, принимающая на свои входы два аналоговых сигнала и выдающая логическую «1», если сигнал на прямом входе («+») больше чем на инверсном входе («−»), и логический «0», если сигнал на прямом входе меньше, чем на инверсном входе.

Схема компаратора представлена на рисунке 1.1.

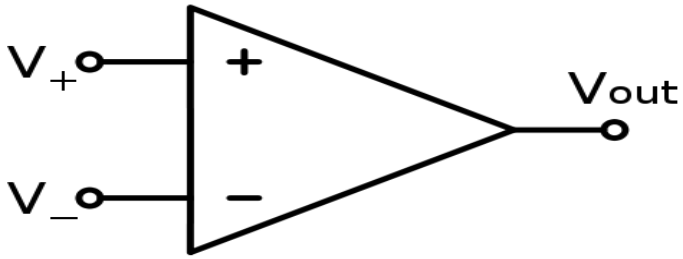

Рис1.1 Схема аналогового компаратора.

Графики преобразования аналогового сигнала в цифровой представлены на рисунке 1.2.

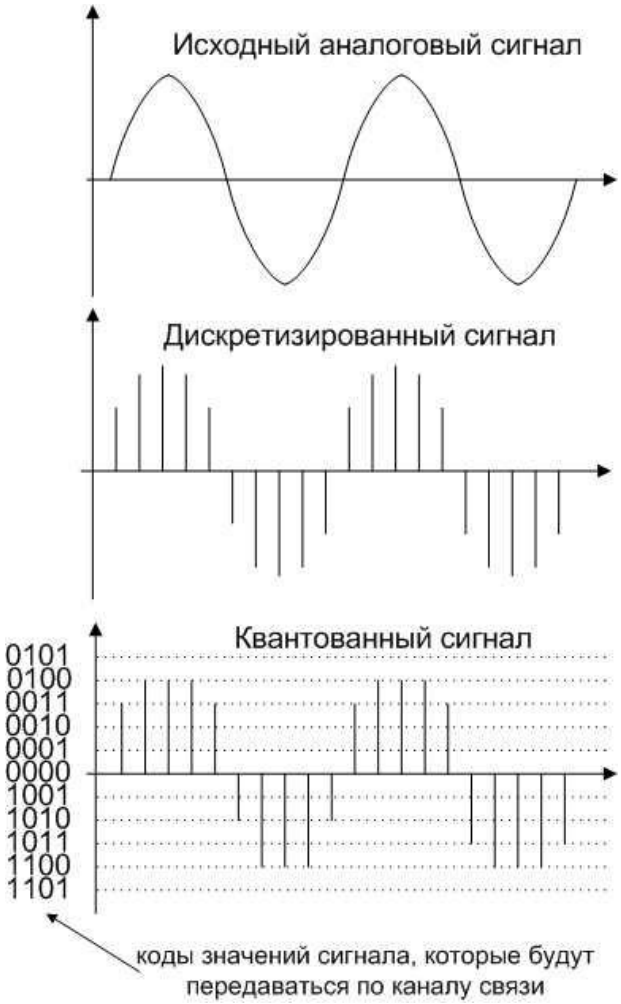

Рис 1.2 Графики преобразования аналогового сигнала в цифровой.

Аналоговый сигнал является непрерывной функцией времени, в АЦП он преобразуется в последовательность цифровых значений. Следовательно, необходимо определить частоту выборки цифровых значений из аналогового сигнала. Частота, с которой производятся цифровые значения, получила название частота дискретизации АЦП.

Непрерывно меняющийся сигнал с ограниченной спектральной полосой подвергается оцифровке (то есть значения сигнала измеряются через интервал времени T— период дискретизации) и исходный сигнал может быть точно восстановлен из дискретных во времени значений путем интерполяции. Точность восстановления ограничена ошибкой квантования. Однако в соответствии с теоремой Котельникова— Шеннона точное восстановление возможно только если частота дискретизации выше, чем удвоенная максимальная частота в спектре сигнала.

Поскольку реальные АЦП не могут произвести аналого-цифровое преобразование мгновенно, входное аналоговое значение должно удерживаться постоянным по крайней мере от начала до конца процесса преобразования (этот интервал времени называют время преобразования). Эта задача решается путем использования специальной схемы на входе АЦП— устройства выборки-хранения (УВХ). УВХ, как правило, хранит входное напряжение на конденсаторе, который соединен со входом через аналоговый ключ: при замыкании ключа происходит выборка входного сигнала (конденсатор заряжается до входного напряжения), при размыкании— хранение. Многие АЦП, выполненные в виде интегральных микросхем содержат встроенное УВХ.

Разрешение АЦП — минимальное изменение величины аналогового сигнала, которое может быть преобразовано данным АЦП — связано с его

разрядностью. В случае единичного измерения без учета шумов разрешение напрямую определяется разрядностью АЦП.

Разрядность АЦП характеризует количество дискретных значений, которые преобразователь может выдать на выходе. В двоичных АЦП измеряется в битах, в троичных АЦП измеряется в тритах. Например, двоичный 8-ми разрядный АЦП, способен выдать 256 дискретных значений, поскольку , троичный 8-ми разрядный АЦП, способен выдать 6561 дискретное значение, поскольку .

Разрешение по напряжению равно разности напряжений, соответствующих максимальному и минимальному выходному коду, деленной на количество выходных дискретных значений. Например:

Диапазон входных значений = от 0 до 10 вольт;

Разрядность двоичного АЦП 12 бит: 212 = 4096 уровней квантования ;

Разрешение двоичного АЦП по напряжению: (10-0)/4096 = 0,00244 В = 2,44 мВ ;

Разрядность троичного АЦП 12 трит: 312 = 531441 уровень квантования ;

Разрешение троичного АЦП по напряжению: (10-0)/531441 = 0,0188 мВ = 18,8 мкВ ;

Квантование (англ. quantization) — разбиение диапазона значений непрерывной или дискретной величины на конечное число интервалов. Существует также векторное квантование — разбиение пространства возможных значений векторной величины на конечное число областей. Простейшим видом квантования является деление целочисленного значения на натуральное число, называемое коэффициентом квантования.

При оцифровке сигнала уровень квантования называют также глубиной дискретизации или битностью. Глубина дискретизации измеряется в битах и обозначает количество бит, выражающих амплитуду сигнала. Чем больше глубина дискретизации, тем точнее цифровой сигнал соответствует аналоговому. В случае однородного квантования глубину дискретизации называют также динамическим диапазоном и измеряют в децибелах (1 бит ≈ 6 дБ). [1]

1.3 Форматы передачи цифровой информации

1.3.1 Форматы локальной передачи данных RS-232

RS-232 (Recommended Standard 232) — используемый в телекоммуникациях, стандарт последовательной асинхронной передачи двоичных данных между терминалом и коммуникационным устройством.

RS-232 — интерфейс передачи информации между двумя устройствами на расстоянии до 15 метров. Информация передается по проводам цифровым сигналом с двумя уровнями напряжения. Логическому "0" соответствует положительное напряжение (от +5 до +15 В для передатчика), а логической "1" отрицательное (от -5 до -15 В для передатчика). Асинхронная передача данных осуществляется с фиксированной скоростью при самосинхронизации фронтом стартового бита.

Интерфейс RS-232-C был разработан для простого применения, однозначно определяемого по его названию: «Интерфейс между терминальным оборудованием и связным оборудованием с обменом по последовательному двоичному коду».

Чаще всего, используется в промышленном и узкоспециальном оборудовании, встраиваемых устройствах. Присутствует на несколько устаревших стационарных компьютерах, в современных чаще всего доступен через дополнительный контроллер/преобразователь (как правило, RS-232 не ставят на портативных компьютерах - на ноутбуках, нетбуках, КПК и тому подобное).

По структуре это обычный асинхронный последовательный протокол, то есть передающая сторона по очереди выдает в линию 0 и 1, а принимающая отслеживает их и запоминает.

Данные передаются пакетами по одному байту (обычно 8 бит).

Вначале передаётся стартовый бит, противоположной полярности состоянию незанятой (idle) линии, после чего передаётся непосредственно кадр полезной информации, от 5 до 8 бит.

Увидев стартовый бит, приемник выжидает интервал T1 и считывает первый бит, потом через интервалы T2 считывает остальные информационные биты. Последний бит — стоповый бит (состояние незанятой линии), говорящий о том, что передача завершена. Возможно 1, 1,5 или 2 стоповых бита.

В конце байта, перед стоп битом, может передаваться бит чётности (parity bit) для контроля качества передачи. Он позволяет выявить ошибку в нечетное число бит (используется, так как наиболее вероятна ошибка в 1 бит). [2]

1.3.2 Форматы передачи по сети Интернет TCP/IP

Стек протоколов TCP/IP (англ. Transmission Control Protocol/Internet Protocol — протокол управления передачей) — набор сетевых протоколов разных уровней модели сетевого взаимодействия DOD, используемых в сетях. Протоколы работают друг с другом в стеке (англ. stack, стопка) — это означает, что протокол, располагающийся на уровне выше, работает «поверх» нижнего, используя механизмы инкапсуляции. Например, протокол TCP работает поверх протокола IP.

Стек протоколов TCP/IP основан на модели сетевого взаимодействия UDOD и включает в себя протоколы четырёх уровней: прикладного, транспортного, сетевого, канального и физического.

Физический уровень

Физический уровень описывает среду передачи данных (будь то коаксиальный кабель, витая пара, оптическое волокно или радиоканал), физические характеристики такой среды и принцип передачи данных (разделение каналов, модуляцию, амплитуду сигналов, частоту сигналов, способ синхронизации передачи, время ожидания ответа и максимальное расстояние).

Канальный уровень

Канальный уровень описывает, каким образом передаются пакеты данных через физический уровень, включая кодирование (то есть специальные последовательности бит, определяющих начало и конец пакета данных). Ethernet, например, в полях заголовка пакета содержит указание того, какой машине или машинам в сети предназначен этот пакет.

Примеры протоколов канального уровня — Ethernet, IEEE 802.11 Wireless Ethernet, SLIP, Token Ring, ATM и MPLS.

MPLS занимает промежуточное положение между канальным и сетевым уровнем и, строго говоря, его нельзя отнести ни к одному из них.

Канальный уровень иногда разделяют на 2 подуровня — LLC и MAC.

Сетевой уровень

Сетевой уровень изначально разработан для передачи данных из одной сети в другую. Примерами такого протокола является X.25 и IPC в сети ARPANET.

С развитием концепции глобальной сети в уровень были внесены дополнительные возможности по передаче из любой сети в любую сеть, независимо от протоколов нижнего уровня, а также возможность запрашивать данные от удалённой стороны, например в протоколе ICMP (используется для передачи диагностической информации IP-соединения) и IGMP (используется для управления multicast-потоками).

ICMP и IGMP расположены над IP и должны попасть на следующий — транспортный — уровень, но функционально являются протоколами сетевого уровня, и поэтому их невозможно вписать в модель OSI.

Пакеты сетевого протокола IP могут содержать код, указывающий, какой именно протокол следующего уровня нужно использовать, чтобы извлечь данные из пакета. Это число — уникальный IP-номер протокола. ICMP и IGMP имеют номера, соответственно, 1 и 2.

К этому уровню относятся: DHCP, DVMRP, ICMP, IGMP, MARS, PIM, RIP, RIP2, RSVP

Транспортный уровень

Протоколы транспортного уровня могут решать проблему негарантированной доставки сообщений («дошло ли сообщение до адресата?»), а также гарантировать правильную последовательность прихода данных. В стеке TCP/IP транспортные протоколы определяют, для какого именно приложения предназначены эти данные.

Протоколы автоматической маршрутизации, логически представленные на этом уровне (поскольку работают поверх IP), на самом деле являются частью протоколов сетевого уровня; например OSPF (IP идентификатор 89).

TCP (IP идентификатор 6) — «гарантированный» транспортный механизм с предварительным установлением соединения, предоставляющий приложению надёжный поток данных, дающий уверенность в безошибочности получаемых данных, перезапрашивающий данные в случае потери и устраняющий дублирование данных. TCP позволяет регулировать нагрузку на сеть, а также уменьшать время ожидания данных при передаче на большие расстояния. Более того, TCP гарантирует, что полученные данные были отправлены точно в такой же последовательности. В этом его главное отличие от UDP.

UDP (IP идентификатор 17) протокол передачи датаграмм без установления соединения. Также его называют протоколом «ненадёжной» передачи, в смысле невозможности удостовериться в доставке сообщения адресату, а также возможного перемешивания пакетов. В приложениях, требующих гарантированной передачи данных, используется протокол TCP.

UDP обычно используется в таких приложениях, как потоковое видео и компьютерные игры, где допускается потеря пакетов, а повторный запрос затруднён или не оправдан, либо в приложениях вида запрос-ответ (например,

запросы к DNS), где создание соединения занимает больше ресурсов, чем повторная отправка.

И TCP, и UDP используют для определения протокола верхнего уровня число, называемое портом.

Прикладной уровень

На прикладном уровне работает большинство сетевых приложений.

Эти программы имеют свои собственные протоколы обмена информацией, например, HTTP для WWW, FTP (передача файлов), SMTP (электронная почта), SSH (безопасное соединение с удалённой машиной), DNS (преобразование символьных имён в IP-адреса) и многие другие.

В массе своей эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например:

HTTP на TCP-порт 80 или 8080,

FTP на TCP-порт 20 (для передачи данных) и 21 (для управляющих команд),

SSH на TCP-порт 22,

запросы DNS на порт UDP (реже TCP) 53,

обновление маршрутов по протоколу RIP на UDP-порт 520.

К этому уровню относятся: Echo, Finger, Gopher, HTTP, HTTPS, IMAP, IMAPS, IRC, NNTP, NTP, POP3, POPS, QOTD, RTSP, SNMP, SSH, Telnet, XDMCP. [3][4]

1.3.3 Стандарт беспроводной связи IEEE 802.15.4

IEEE 802.15.4 — стандарт, который определяет физический слой и управление доступом к среде для беспроводных персональных сетей с низким уровнем скорости. Стандарт поддерживается рабочей группой IEEE 802.15. Является базовой основой для протоколов ZigBee, WirelessHART, и MiWi, каждый из которых, в свою очередь, предлагает решение для построения сетей посредством постройки верхних слоёв, которые не регламентируются стандартом.

Цель стандарта IEEE 802.15 — предложить нижние слои основания сети для сетей типа беспроводных персональных сетей, ориентированных на низкую стоимость, низкую скорость повсеместной связи между устройствами (по контрасту с многими более конечно-ориентированных на пользователя сетями, как например Wi-Fi). Акцент делается на очень низкой стоимости связи с ближайшими устройствами, совсем без (или с небольшой) базовой структурой, с целью эксплуатации на доселе небывалом низком уровне энергии.

Основной предел приёма — 10-метровая область связи со скоростью передачи 250 кбит/с. Компромиссы возможны в пользу более радикально встраиваемых устройств с ещё более низкой потребностью в энергии, путём определения не одного, а нескольких физических уровней. Первоначально были определены низкие скорости передачи в 20 и 40 кбит/с, скорость в 100 кбит/с была добавлена в текущем перевыпуске.

Ещё более низкие скорости передачи могут быть рассмотрены с результирующим эффектом снижения энергопотребления. Как уже упоминалось, главной отличительной особенностью стандарта 802.15.4 среди

беспроводных персональных сетей важным является низкая стоимость производства и расходов по эксплуатации, простота технологии.

В ряду важнейших функций находятся обеспечение работы в режиме реального времени посредством сохранения временных слотов, предотвращение одновременного доступа и комплексная поддержка защиты сетей. Устройства также включают функции управления расходом энергии, такие как качество соединений и детектирование энергии. Совместимые со стандартом 802.15.4 устройства могут использовать одну из трёх возможных частотных полос для работы.

Подводя итог данной главы можно сказать, что рассмотренные выше технологии следует применить при создании измерительной системы. Следующим этапом работы автоматической измерительной системы, является хранение и представление данных измерений. Данный вопрос рассмотрен в главе 2. [5]

2 Методы хранения и представления информации

2.1 Методы хранения данных

В зависимости от типа данных хранить их можно в различном виде – это может быть текстовый файл, изображение, звукозапись, видеозапись и другое.

Для хранения числовых данных измерений наиболее удобно использовать базы данных.

2.1.1Системы управления базами данных

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

Основные функции СУБД: управление данными во внешней памяти (на дисках), управление данными в оперативной памяти с использованием дискового кэша, журнализация изменений, резервное копирование и восстановление базы данных после сбоев, поддержка языков БД (язык определения данных, язык манипулирования данными).

Обычно современная СУБД содержит следующие компоненты: ядро, которое отвечает за управление данными во внешней и оперативной памяти, и журнализацию, процессор языка базы данных, обеспечивающий оптимизацию запросов на извлечение и изменение данных и создание, как правило, машинно-независимого исполняемого внутреннего кода, подсистему поддержки времени исполнения, которая интерпретирует программы манипуляции данными, создающие пользовательский интерфейс с СУБД,

сервисные программы (внешние утилиты), обеспечивающие ряд дополнительных возможностей по обслуживанию информационной системы.

Классификации СУБД

По модели данных:

а) Иерархические. Иерархические базы данных могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй — объекты второго уровня и т. д. Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок не имеет потомков или имеет их несколько, тогда как у объекта-потомка обязательно только один предок. Объекты, имеющие общего предка, называются близнецами;

б) Сетевые. К основным понятиям сетевой модели базы данных относятся: уровень, элемент (узел), связь. Узел — это совокупность атрибутов данных, описывающих некоторый объект. На схеме иерархического дерева узлы представляются вершинами графа. В сетевой структуре каждый элемент может быть связан с любым другим элементом. Сетевые базы данных подобны иерархическим, за исключением того, что в них имеются указатели в обоих направлениях, которые соединяют родственную информацию. Несмотря на то, что эта модель решает некоторые проблемы, связанные с иерархической моделью, выполнение простых запросов остается достаточно сложным процессом. Также, поскольку логика процедуры выборки данных зависит от физической организации этих данных, то эта модель не является полностью

независимой от приложения. Другими словами, если необходимо изменить структуру данных, то нужно изменить и приложение;

в) Реляционные. Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных. Реляционная модель ориентирована на организацию данных в виде двумерных таблиц. Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

    1. каждый элемент таблицы — один элемент данных;
    2. все ячейки в столбце таблицы однородные, то есть все элементы в столбце имеют одинаковый тип (числовой, символьный и т. д.);
    3. каждый столбец имеет уникальное имя;
    4. одинаковые строки в таблице отсутствуют;
    5. порядок следования строк и столбцов может быть произвольным;

г) Объектно-ориентированные. Эта система управления обрабатывает данные как абстрактные объекты, наделенные свойствами, в виде неструктурированных данных, и использующие методы взаимодействия с другими объектами;

д) Объектно-реляционные. Объектно-реляционная СУБД — реляционная СУБД , поддерживающая некоторые технологии, реализующие объектно-ориентированный подход.

По степени распределенности:

а) Локальные СУБД (все части локальной СУБД размещаются на одном компьютере);
б) Распределенные СУБД (части СУБД могут размещаться на двух и более компьютерах).

По способу доступа к БД:

а) Файл-серверные. В файл-серверных СУБД файлы данных располагаются централизованно на файл-сервере. СУБД располагается на каждом клиентском компьютере (рабочей станции). Доступ СУБД к данным осуществляется через локальную сеть. Синхронизация чтений и обновлений осуществляется посредством файловых блокировок. Преимуществом этой архитектуры является низкая нагрузка на процессор файлового сервера. Недостатки: потенциально высокая загрузка локальной сети; затрудненность или невозможность централизованного управления; затрудненность или невозможность обеспечения таких важных характеристик как высокая надежность, высокая доступность и высокая безопасность. Применяются чаще всего в локальных приложениях, которые используют функции управления БД; в системах с низкой интенсивностью обработки данных и низкими пиковыми нагрузками на БД. На данный момент файл-серверная технология считается устаревшей. Примеры: Microsoft Access, Paradox, dBase, FoxPro, Visual FoxPro;
б) Клиент-серверные. Клиент-серверная СУБД располагается на сервере вместе с БД и осуществляет доступ к БД непосредственно, в монопольном режиме. Все клиентские запросы на обработку данных обрабатываются клиент-серверной СУБД централизованно. Недостаток клиент-серверных СУБД состоит в повышенных требованиях к серверу. Достоинства: потенциально более низкая загрузка локальной сети;

удобство централизованного управления; удобство обеспечения таких важных характеристик как высокая надежность, высокая доступность и высокая безопасность. Примеры: Oracle, Firebird, Interbase, IBM DB2, Informix, MS SQL Server, Sybase Adaptive Server Enterprise, PostgreSQL, MySQL, Caché, ЛИНТЕР;

в) Встраиваемые. Встраиваемая СУБД — СУБД, которая может поставляться как составная часть некоторого программного продукта, не требуя процедуры самостоятельной установки. Встраиваемая СУБД предназначена для локального хранения данных своего приложения и не рассчитана на коллективное использование в сети. Физически встраиваемая СУБД чаще всего реализована в виде подключаемой библиотеки. Доступ к данным со стороны приложения может происходить через SQL либо через специальные программные интерфейсы. Примеры: OpenEdge, SQLite, BerkeleyDB, Firebird Embedded, Microsoft SQL Server Compact, ЛИНТЕР. [17]

2.1.2 Реляционная СУБД MySQL

MySQL - свободная система управления базами данных (СУБД). MySQL является собственностью компании Oracle Corporation, получившей ее вместе с поглощенной Sun Microsystems, осуществляющей разработку и поддержку приложения. Распространяется под GNU General Public License или под собственной коммерческой лицензией. Помимо этого разработчики создают функциональность по заказу лицензионных пользователей, именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP. Обычно MySQL используется в качестве сервера, к

которому обращаются локальные или удаленные клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

Платформы

MySQL портирована на большое количество платформ: AIX, BSDi, FreeBSD, HP-UX, Linux, Mac OS X, NetBSD, OpenBSD, OS/2 Warp, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows NT, Windows 2000, Windows XP, Windows Server 2003, WinCE, Windows Vista и Windows 7. Существует также порт MySQL к OpenVMS. Важно отметить, что на официальном сайте СУБД для свободной загрузки предоставляются не только исходные коды, но и откомпилированные и оптимизированные под конкретные операционные системы готовые исполняемые модули СУБД MySQL.

Языки программирования

MySQL имеет API для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, PureBasic, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

Технические характеристики

Максимальные размеры таблиц в My SQL, начиная с версии 3.23 до 8 миллионов терабайт.

Размер таблицы ограничен ее типом. В общем случае тип MyISAM ограничен предельным размером файла в файловой системе операционной системы. Например в NTFS этот размер теоретически может быть до 32 эксабайт. В случае InnoDB одна таблица может храниться в нескольких файлах, представляющих единое табличное пространство. Размер последнего может достигать 64 терабайт. [6][7]

Для представления информации хранимой в базе данных используются современные технологии, о которых пойдёт речь в пункте 2.2.

2.2 Методы представления данных

2.2.1 Гипертекстовый язык разметки HTML

HTML (от англ. HyperText Markup Language — «язык разметки гипертекста») — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме.

Язык HTML был разработан британским ученым Тимом Бернерсом-Ли приблизительно в 1989—1991 годах в стенах Европейского совета по ядерным исследованиям в Женеве (Швейцария). HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области верстки. HTML успешно справлялся с проблемой сложности SGML путем определения небольшого

набора структурных и семантических элементов — дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простой, но красиво оформленный документ. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

Изначально язык HTML был задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой HTML должен был без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащенностью (цветной экран современного компьютера, монохромный экран органайзера, ограниченный по размерам экран мобильного телефона или устройства и программы голосового воспроизведения текстов). Однако современное применение HTML очень далеко от его изначальной задачи. Например, тег <TABLE>, несколько раз использованный для форматирования страницы, которую вы на данный момент читаете, предназначен для создания в документах самых обычных таблиц, но, как можно убедиться, здесь нет ни одной таблицы. С течением времени, основная идея платформонезависимости языка HTML была отдана в своеобразную жертву современным потребностям в мультимедийном и графическом оформлении.

Текстовые документы, содержащие разметку на языке HTML (такие документы традиционно имеют расширение .html или .htm), обрабатываются специальными приложениями, которые отображают документ в его форматированном виде. Такие приложения, называемые «браузерами» или «интернет-обозревателями», обычно предоставляют пользователю удобный интерфейс для запроса веб-страниц, их просмотра (и вывода на иные внешние устройства) и, при необходимости, отправки введенных пользователем данных на сервер. Наиболее популярными на сегодняшний день браузерами

являются Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome и Opera.

В настоящее время Консорциум всемирной паутины разрабатывает HTML версии 5. Черновой вариант спецификации языка появился в Интернете 20 ноября 2007 года.

Сообществом WHATWG (англ. Web Hypertext Application Technology Working Group), начиная с 2004 года, разрабатывается спецификация Web Applications 1.0, часто неофициально называемая «HTML 5», которая расширяет HTML (впрочем, имея и совместимый с XHTML 1.0 XML-синтаксис) для лучшего представления семантики различных типичных страниц, например форумов, сайтов аукционов, поисковых систем, онлайн-магазинов и т. д., которые не очень удачно вписываются в модель XHTML 2.

Структура HTML-документа

HTML — теговый язык разметки документов. Любой документ на языке HTML представляет собой набор элементов, причем начало и конец каждого элемента обозначается специальными пометками — тегами. Элементы могут быть пустыми, то есть не содержащими никакого текста и других данных (например, тег перевода строки
). В этом случае обычно не указывается закрывающий тег. Кроме того, элементы могут иметь атрибуты, определяющие какие-либо их свойства (например, размер шрифта для элемента font). Атрибуты указываются в открывающем теге. Пример фрагментов HTML – документа представлен на рисунке 2.1.

```
<strong>Текст между двумя тегами — открывающим и закрывающим.</strong>

<a href="http://www.example.com">Здесь элемент содержит атрибут href.</a>
```

Рисунок 2.1 – Пример HTML – документа

А вот пример пустого элемента:

Регистр, в котором набрано имя элемента и имена атрибутов, в HTML значения не имеет (в отличие от XHTML). Элементы могут быть вложенными. Например, следующий код представленный на рисунке 2.2 , показывает пример выделения текста.

```
<b>
 Этот текст будет полужирным,
 <i>а этот - еще и курсивным</i>
</b>
```

Этот текст будет полужирным, ***а этот — еще и курсивным***.

Рисунок 2.2 – Выделение текста в HTML

Кроме элементов, в HTML-документах есть и сущности (англ. entities) — «специальные символы». Сущности начинаются с символа амперсанда и имеют вид &имя; или &#NNNN;, где NNNN — код символа в Юникоде в десятичной системе счисления.

Например, © — знак авторского права (©). Как правило, сущности используются для представления символов, отсутствующих в кодировке документа, или же для представления «специальных» символов: & — амперсанда (&), < — символа «меньше» (<) и > — символа «больше» (>), которые некорректно записывать «обычным» образом, из-за их особого значения в HTML.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, должен начинаться со строки объявления версии HTML <!DOCTYPE…>, которая обычно выглядит примерно так как представлено на рисунке 2.3.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

Рисунок 2.3 – Строка спецификации HTML

Если эта строка не указана, то добиться корректного отображения документа в браузере становится труднее.

Далее обозначается начало и конец документа тегами <html> и </html> соответственно. Внутри этих тегов должны находиться теги заголовка (<head></head>) и тела (<body></body>) документа.

Варианты DOCTYPE для HTML 4.01

Строгий: не содержит элементов, помеченных как «устаревшие» или «не одобряемые» . Объявление строгой спецификации для HTML 4.01 представлено на рисунке 2.4.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

Рисунок 2.4 – Объявление строгой спецификации для HTML 4.01.

Переходный : содержит устаревшие теги в целях совместимости и упрощения перехода со старых версий HTML. Объявление переходной спецификации для HTML 4.01 представлено на рисунке 2.5.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
```

Рисунок 2.5 – Объявление переходной спецификации для HTML 4.01

С фреймами : аналогичен переходному, но содержит также теги для создания наборов фреймов. Объявление спецификации с фреймами для HTML 4.01 представлено на рисунке 2.6.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN"
   "http://www.w3.org/TR/html4/frameset.dtd">
```

Рисунок 2.6 – Объявление спецификации с фреймами для HTML 4.01

Варианты DOCTYPE для HTML 5

В HTML 5 используется только один вариант DOCTYPE, представленный на рисунке 2.7. [8][9]

```
<!DOCTYPE HTML>
```

Рисунок 2.7 – Спецификация для HTML 5

HTML служит только для разметки internet-страниц, а для визуального оформления внешнего вида документа используются каскадные таблицы стилей, о которых речь идет в пункте 2.2.2.

2.2.2 Каскадные таблицы стилей CSS

CSS (англ. Cascading Style Sheets — каскадные таблицы стилей) — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML.

Цель создания CSS

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом. Кроме того, CSS позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана), или при выводе устройствами, использующими шрифт Брайля.

История создания и развития CSS

CSS — одна из широкого спектра технологий, одобренных консорциумом W3C и получивших общее название «стандарты Web». В

1990-х годах стала ясна необходимость стандартизировать Web, создать какие-то единые правила, по которым программисты и веб-дизайнеры проектировали бы сайты. Так появились языки HTML 4.01 и XHTML и стандарт CSS.

В начале 1990-х различные браузеры имели свои стили для отображения веб страниц. HTML развивался очень быстро и был способен удовлетворить все существовавшие на тот момент потребности по оформлению информации, поэтому CSS не получил тогда широкого признания.

Термин «каскадные таблицы стилей» был предложен Хокон Виум Ли в 1994 году. Совместно с Бертом Босом он стал развивать CSS.

В отличие от многих существовавших на тот момент языков стиля, CSS использует наследование от родителя к потомку, поэтому разработчик может определить разные стили, основываясь на уже определенных ранее стилях.

В середине 1990-х Консорциум Всемирной паутины (W3C) стал проявлять интерес к CSS, и в декабре 1996 года была издана рекомендация CSS1.

Уровень 1 (CSS1)

Рекомендация W3C, принята 17 декабря 1996 года, откорректирована 11 января 1999 года. Среди возможностей, предоставляемых этой рекомендацией:

а) параметры шрифтов. Возможности по заданию гарнитуры и размера шрифта, а также его стиля — обычного, курсивного или полужирного;
б) цвета. Спецификация позволяет определять цвета текста, фона, рамок и других элементов страницы;

в) атрибуты текста. Возможность задавать межсимвольный интервал, расстояние между словами и высоту строки (то есть межстрочные отступы);
г) выравнивание для текста, изображений, таблиц и других элементов;
д) свойства блоков, такие как высота, ширина, внутренние (padding) и внешние (margin) отступы и рамки. Так же в спецификацию входили ограниченные средства по позиционированию элементов, такие как float и clear.

Уровень 2 (CSS2)

Рекомендация W3C, принята 12 мая 1998 года. Основана на CSS1 с сохранением обратной совместимости за несколькими исключениями. Добавление к функциональности:

а) блочная верстка. Появились относительное, абсолютное и фиксированное позиционирование. Позволяет управлять размещением элементов по странице без табличной верстки;
б) типы носителей. Позволяет устанавливать разные стили для разных носителей (например монитор, принтер, КПК);
в) звуковые таблицы стилей. Определяет голос, громкость и т. д. для звуковых носителей (например для слепых посетителей сайта);
г) страничные носители. Позволяет, например, установить разные стили для элементов на четных и нечетных страницах при печати;
д) расширенный механизм селекторов;
е) указатели;
ж) генерируемое содержимое. Позволяет добавлять содержимое, которого нет в исходном документе, до или после нужного элемента.

В настоящее время W3C больше не поддерживает CSS2 и рекомендует использовать CSS2.1.

Уровень 2, ревизия 1 (CSS2.1)

Рекомендация W3C, принята 7 июня 2011 года.

CSS2.1 основана на CSS2. Кроме исправления ошибок, в новой ревизии изменены некоторые части спецификации, а некоторые и вовсе удалены. Удаленные части могут в будущем быть добавлены в CSS3.

Уровень 3 (CSS3)

Разрабатываемая версия. Сильно расширена по сравнению с предыдущими версиями. Нововведения, начиная с малых, вроде закругленных углов блоков, заканчивая трансформацией (анимацией) и, возможно, введением переменных.

Уровень 4 (CSS4)

Разрабатывается W3C с 29 сентября 2011 года.

Способы подключения CSS к документу

Правила CSS пишутся на формальном языке CSS и располагаются в таблицах стилей, то есть таблицы стилей содержат в себе правила CSS. Эти таблицы стилей могут располагаться как в самом веб-документе, внешний вид которого они описывают, так и в отдельных файлах, имеющих формат CSS. (По сути, формат CSS — это обычный текстовый файл. В файле .css не содержится ничего, кроме перечня правил CSS и комментариев к ним.)

То есть, эти таблицы стилей могут быть подключены, внедрены в описываемый ими веб-документ четырьмя различными способами:

когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством тега <link>, располагающегося в этом документе между тегами <head> и </head>как показано на рисунке 2.8. (Тег <link> будет иметь атрибут href, имеющий значением адрес этой таблицы

стилей). Все правила этой таблицы действуют на протяжении всего документа;

```
<head>  <link rel="stylesheet" type="text/css" href="style.css"></head>
```

Рисунок 2.8 – подключение CSS таблицы стилей к HTML документу

когда таблица стилей находится в отдельном файле, она может быть подключена к веб-документу посредством директивы @import, располагающейся в этом документе между тегами <style> и </style> (которые, в свою очередь, располагаются в этом документе между тегами <head> и </head>) сразу после тега <style>, которая также указывает (в своих скобках, после слова url) на адрес этой таблицы стилей. Пример подключения представлен на рисунке 2.9. Все правила этой таблицы действуют на протяжении всего документа;

```
<head>
.....
  <style type="text/css" media="all">
    @import url(style.css);
  </style>
</head>
```

Рисунок 2.9 – подключение CSS таблицы стилей к HTML документу

когда таблица стилей описана в самом документе, она может располагаться в нем между тегами <style> и </style> как представлено на рисунке 2.10 (которые, в свою очередь, располагаются в этом документе между тегами

<head> и </head>). Все правила этой таблицы действуют на протяжении всего документа;

```
<head>
.....
 <style type="text/css">
  body {
    color: red;
  }
 </style>
</head>
```

Рисунок 2.10 – внедрение CSS таблицы стилей в HTML документ

когда таблица стилей описана в самом документе, она может располагаться в нем в теле какого-то отдельного тега (посредством его атрибута style) этого документа, как представлено на рисунке 2.11. Все правила этой таблицы действуют только на содержимое этого тега.

```
<p style="font-size: 21px; color: green;">Рассказ о том, как вредно красить батареи</p>
```

Рисунок 2.11 – применение CSS таблицы стилей к отдельному тегу HTML документа

В первых двух случаях говорят, что к документу применены внешние таблицы стилей, а во вторых двух случаях — внутренние таблицы стилей.

Для добавления CSS к XML-документу, последний должен содержать специальную ссылку на таблицу стилей как показано на рисунке 2.12.

```
<?xml-stylesheet type="text/css" href="style.css"?>
```

Рисунок 2.12 – Добавления CSS к XML-документу

Иерархия элементов внутри документа

Как известно, HTML-документы строятся на основании иерархии элементов, которая может быть наглядно представлена в древовидной форме. Элементы HTML друг для друга могут быть родительскими, дочерними, элементами-предками, элементами-потомками, сестринскими.

Элемент является родителем другого элемента, если в иерархической структуре документа он находится сразу, непосредственно над этим элементом. Элемент является предком другого элемента, если в иерархической структуре документа он находится где-то выше этого элемента.

Пускай, например, в документе присутствуют два абзаца p, включающие в себя шрифт с полужирным начертанием b. Тогда элементы b будут дочерними элементами своих родительских элементов p, и потомками своих предков body. В свою очередь, для элементов p элемент body будет являться только родителем. И кроме того, эти два элемента p будут являться сестринскими элементами, как имеющими одного и того же родителя — body.

В CSS могут задаваться, при помощи селекторов, не только одиночные элементы, но и элементы, являющиеся потомками, дочерними или сестринскими элементами других элементов (см. подраздел «виды селекторов»).

Правила построения CSS

В первых трех случаях подключения таблицы CSS к документу (см. выше) каждое правило CSS из таблицы стилей имеет две основные части — селектор и блок объявлений. Селектор, расположенный в левой части правила, определяет, на какие части документа распространяется правило. Блок объявлений располагается в правой части правила. Он помещается в фигурные скобки, и, в свою очередь, состоит из одного или более объявлений, разделенных знаком «;». Каждое объявление представляет собой сочетание свойства CSS и значения, разделенных знаком «:». Селекторы могут группироваться в одной строке через запятую. В таком случае свойство применяется к каждому из них. Пример правильного построения CSS документа представлен на рисунке 2.13.

```
селектор, селектор {
  свойство: значение;
  свойство: значение;
}
```

Рисунок 2.13 – Пример правильного построения CSS документа

В четвертом случае подключения таблицы CSS к документу (см. список) правило CSS (являющееся значением атрибута style тега, на который оно

действует) представляет собой перечень объявлений («свойство CSS : значение»), разделенных знаком «;».

Виды селекторов

Селекторы правила CSS могут быть:

селекторами элементов;

```
p {font-family: Garamond, serif;}
```

селекторами классов;

```
.note {color: red; background: yellow; font-weight: bold;}
```

селекторами идентификаторов;

```
#paragraph1 {margin: 0;}
```

селекторами атрибутов;

```
 a[href="http://www.somesite.com"]{font-weight:bold;}
```

селекторами потомков (контекстными селекторами);

```
div#paragraph1 p.note {color: red;}
```

селекторами дочерних элементов;

```
p.note > b {color: green;}
```

селекторами сестринских элементов;

```
h1 + p {font-size: 24pt;}
```

селекторами псевдоклассов;

```
a:active {color:yellow;}
```

селекторами псевдоэлементов.

```
p::first-letter {font-size: 32px;}
```

Также в CSS существует так называемый универсальный селектор, обозначающий любой элемент, встречающийся в документе. Например, * {color:red;}. Перед любым селектором, задающим класс или идентификатор, можно поставить знак универсального селектора, в результате получится эквивалентное выражение, например, .first {...} и *.first {...} имеют один и тот же смысл.

Классы элементов. Идентификаторы элементов

Класс или идентификатор может быть присвоен какому-нибудь элементу (тегу) HTML посредством атрибутов class или id этого элемента (тега), как показано на рисунке 2.14.

```
<div id="first"> ... </div>

<p class="big"> ... </p>
```

Рисунок 2.14 – Присвоение классов и идентификаторов к элементам(тегам)

Основное отличие между классами элементов и идентификаторами элементов в том, что в документе какой-нибудь класс может быть присвоен сразу нескольким элементам, а идентификатор - только одному. Также отличие в том, что могут существовать множественные классы (когда класс элемента состоит из нескольких слов, разделенных пробелами). Для идентификаторов такое невозможно.

Важно отметить следующее отличие идентификатора от класса: идентификаторы широко используются в JavaScript для нахождения уникального элемента в документе.

Имена классов и идентификаторов, в отличие от названий тегов и их атрибутов, чувствительны к регистру ввода букв.

Свойства классов и идентификаторов задаются с помощью соответствующих селекторов. Причем может быть задано как свойство класса в целом (в таком случае селектор начинается с «.»), или свойство идентификатора самого по себе (в таком случае селектор начинается с «#»), так и свойство какого-нибудь элемента этого класса или с этим идентификатором.

В CSS помимо классов, задаваемых автором страницы, существует также ограниченный набор так называемых псевдоклассов, описывающих вид гиперссылок с определенным состоянием в документе, вид элемента, на котором находится фокус ввода, а также вид элементов, являющихся первыми дочерними элементами других элементов. Также в CSS существует четыре так называемых псевдоэлемента: первая буква, первая строка, применение специальных стилей до и после элемента.

Наследование. Каскадирование. Приоритеты стилей CSS

Применение CSS к документам HTML основано на принципах наследования и каскадирования. Принцип наследования заключается в том, что свойства CSS, объявленные для элементов-предков, наследуются элементами потомками. Но, естественно, не все свойства CSS наследуются — например, если для тега параграфа p средствами CSS задана рамка, то она не будет наследоваться ни одним тегом, содержащимся в данном теге p, а вот если для параграфа p средствами CSS задан цвет шрифта (например, color:green;), то это свойство будет унаследовано каждым элементом-тегом, находящимся в параграфе.

Принцип каскадирования применяется в случае, когда какому-то элементу HTML одновременно поставлено в соответствие более одного

правила CSS, то есть, когда происходит конфликт значений этих правил. Чтобы разрешить такие конфликты вводятся правила приоритета:

а) наиболее низким приоритетом обладает стиль браузера;
б) следующим по значимости является стиль, заданный пользователем браузера в его настройках;
в) и наиболее высоким приоритетом обладает стиль, заданный непосредственно автором страницы;
г) самым низким приоритетом обладают стили, наследуемые в документе элементом от своих предков;
д) более высоким приоритетом обладают стили, заданные во внешних таблицах стилей, подключенных к документу;
е) еще более высоким приоритетом обладают стили, заданные непосредственно селекторами всех десяти видов (см. подраздел «виды селекторов»), содержащимися в контейнерах style данного документа. Нередки случаи, когда к какому-нибудь элементу имеют отношение, задают его вид, несколько таких селекторов. Такие конфликты между ними разрешаются с помощью расчета специфичности каждого такого селектора и применения этих селекторов к данному элементу в порядке убывания их специфичностей. При расчете специфичности селектора принимается во внимание;
ж) количество идентификаторов (#id) в селекторе —- ((1,0,0) за каждый объявленный идентификатор в селекторе правила CSS);
з) количество классов (.class) и псевдоклассов (:pseudoclass) в селекторе —- ((0,1,0) за каждый объявленный класс и псевдокласс в селекторе правила CSS);
и) количество тегов в селекторе —- ((0,0,1) за каждый объявленный тег в селекторе правила CSS). (Принцип расчета таков, что, например, (1,0,0) будет иметь большую специфичность, соответственно — больший

приоритет, чем даже (0,10,0), а (0,1,0) будет иметь большую специфичность, больший приоритет, чем (0,0,10). Если же рассчитанные таким образом специфичности окажутся одинаковыми, то к элементу будет применено правило, описанное селектором, расположенным в документе ниже других.);

к) еще более высоким приоритетом обладают стили, объявленные непосредственно в теге данного элемента посредством атрибута style этого тега;

л) и наконец самым высоким приоритетом обладают стили, объявленные автором страницы или пользователем, с помощью сопроводительного слова !important. Если таких свойств несколько, то предпочтение отдается в первую очередь стилям, заданным пользователем, а для остальных свойств (которые будут являться задаваемыми автором страницы) потребуется определить их специфичности по принципам, описанным выше, и применять эти свойства в порядке убывания этих их специфичностей.

Пример таблицы стилей

Пример таблицы стилей (в таком виде она может быть либо размещена в отдельном файле .css, либо же обрамлена тегами <style> и размещена в «шапке» той самой веб-страницы, на которую она действует) представлен на рисунке 2.15.

```
  font-size: 110 %;
  color: red;
  background: white;
}
.note {
  color: red;
  background: yellow;
  font-weight: bold;
}
p#paragraph1 {
  margin: 0;}
a:hover {
  text-decoration: none;
}
#news p {
  color: blue;
}
```

Рисунок 2.15 – Пример таблицы стилей

Здесь приведено шесть правил CSS с селекторами p, h2, .note, p#paragraph1, a:hover и #news p.

Первое правило присвоено HTML-элементу p (абзацу) — назначен стиль. Абзацы будут отображаться шрифтом Garamond, или, если такой шрифт недоступен, каким-либо другим шрифтом с засечками («serif»).

Второе правило присвоено HTML-элементу h2 (заголовку второго уровня). Заголовок второго уровня будет отображаться красным на белом фоне с увеличенным кеглем.

Третье правило будет применено к ЛЮБОМУ элементу, атрибут class которого равен 'note'. Например, к параграфу: Этот абзац будет выведен полужирным шрифтом красного цвета на желтом фоне.

Четвертое правило будет применяться только к элементу p, атрибут id которого равен paragraph1. Такой элемент не будет иметь внешних отступов (margin).

Пятое правило определяет стиль hover для элементов a — гиперссылок. По умолчанию, в большинстве браузеров текст элементов a подчеркивается. Это правило уберет подчеркивание, когда указатель мыши находится над этими элементами.

Последнее, шестое правило, применяется для элементов p, которые находятся внутри КАКОГО-ЛИБО элемента с атрибутом id, равным «news» (#news p — это типичный случай селектора потомков, см. 5-й пункт списка выше).

CSS-верстка

До появления CSS оформление веб-страниц осуществлялось исключительно средствами HTML, непосредственно внутри содержимого документа. Однако с появлением CSS стало возможным принципиальное разделение содержания и представления документа. За счет этого нововведения стало возможным легкое применение единого стиля

оформления для массы схожих документов, а также быстрое изменение этого оформления.

Преимущества:

Несколько дизайнов страницы для разных устройств просмотра. Например, на экране дизайн будет рассчитан на большую ширину, во время печати меню не будет выводиться, а на КПК и сотовом телефоне меню будет следовать за содержимым.

Уменьшение времени загрузки страниц сайта за счет переноса правил представления данных в отдельный CSS-файл. В этом случае браузер загружает только структуру документа и данные, хранимые на странице, а представление этих данных загружается браузером только один раз и может быть закешировано.

Простота последующего изменения дизайна. Не нужно править каждую страницу, а лишь изменить CSS-файл.

Дополнительные возможности оформления. Например, с помощью CSS-верстки можно сделать блок текста, который остальной текст будет обтекать (например для меню) или сделать так, чтобы меню было всегда видно при прокрутке страницы.

Недостатки:

Различное отображение верстки в различных браузерах (особенно устаревших), которые по разному интерпретируют одни и те же данные CSS.

Часто встречающаяся необходимость на практике исправлять не только один CSS-файл, но и теги HTML, которые сложным и ненаглядным способом связаны с селекторами CSS, что иногда сводит на нет простоту применения единых файлов стилей и значительно удлиняет время редактирования и тестирования. [9][10]

2.2.3 Технология AJAX

AJAX, Ajax (от англ. Asynchronous Javascript and XML — «асинхронный JavaScript и XML») — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. В результате, при обновлении данных, веб-страница не перезагружается полностью, и веб-приложения становятся более быстрыми и удобными.

Технология

AJAX — не самостоятельная технология, а концепция использования нескольких смежных технологий. AJAX базируется на двух основных принципах:

использование технологии динамического обращения к серверу «на лету», без перезагрузки всей страницы полностью, например:

а) с использованием XMLHttpRequest (основной объект);
б) через динамическое создание дочерних фреймов;
в) через динамическое создание тега <script>;
г) через динамическое создание тега <img>, как это реализовано в google analytics;
д) использование DHTML для динамического изменения содержания страницы.

В качестве формата передачи данных могут использоваться фрагменты простого текста, HTML-кода, JSON или XML.

Для обмена данными с сервером используется специальный объект XmlHttpRequest, который умеет отправлять запрос и получать ответ с сервера.

Кроссбраузерно создать такой объект можно так, как показано на рисунке 2.16. [11][12

```
function getXmlHttp(){

var xmlhttp;

try {

xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");

} catch (e) {

try {

xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");

} catch (E) {

xmlhttp = false;

}

}

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {

xmlhttp = new XMLHttpRequest();

}

return xmlhttp;

}
```

Рисунок 2.16 – Пример кроссбраузерного создания объекта XmlHttpRequest

Для динамической передачи данных из базы данных в AJAX скрипт используется технология XML, о которой рассказывается в главе 2.2.4.

2.2.4 Формат обмена данными XML

—

- -]) — рекомендованный Консорциумом Всемирной паутины язык разметки, фактически представляющий собой свод общих синтаксических правил. XML — текстовый формат, предназначенный для хранения структурированных данных (взамен существующих файлов баз данных), для обмена информацией между программами.

Правильно построенные и действительные документы XML

Стандартом определены два уровня правильности документа XML:

а) правильно построенный (англ. well-formed). Правильно построенный документ соответствует всем общим правилам синтаксиса XML, применимым к любому XML-документу. И если, например, начальный тег не имеет соответствующего ему конечного тега, то это неправильно построенный документ XML. Документ, который неправильно построен, не может считаться документом XML; XML-процессор (парсер) не должен обрабатывать его обычным образом и обязан классифицировать ситуацию как фатальная ошибка;

б) действительный (англ. valid). Действительный документ дополнительно соответствует некоторым семантическим правилам. Это более строгая дополнительная проверка корректности документа на соответствие заранее определенным, но уже внешним правилам, в целях минимизации количества ошибок, например, структуры и состава данного, конкретного документа или семейства документов. Эти правила могут быть разработаны как самим пользователем, так и сторонними разработчиками, например, разработчиками словарей или стандартов обмена данными. Обычно такие правила хранятся в

специальных файлах — схемах, где самым подробным образом описана структура документа, все допустимые названия элементов, атрибутов и многое другое. И если документ, например, содержит не определенное заранее в схемах название элемента, то XML-документ считается недействительным; проверяющий XML-процессор (валидатор) при проверке на соответствие правилам и схемам обязан (по выбору пользователя) сообщить об ошибке.

Синтаксис XML

XML — это описанная в текстовом формате иерархическая структура, предназначенная для хранения любых данных. Визуально структура может быть представлена как дерево элементов. Элементы XML описываются тегами.

Первая строка XML-документа называется объявление XML (англ. XML declaration) — это строка, указывающая версию XML. В версии 1.0 объявление XML может быть опущено, в версии 1.1 оно обязательно. Также здесь может быть указана кодировка символов и наличие внешних зависимостей. Пример объявление XML – документа представлен на рисунке 2.17.

```
<?xml version="1.0" encoding="UTF-8"?>
```

Рисунок 2.17 – Пример объявление XML – документа

Спецификация требует, чтобы процессоры XML обязательно поддерживали Юникод-кодировки UTF-8 и UTF-16 (UTF-32 не обязателен). Признаются допустимыми, поддерживаются и широко используются (но не обязательны) другие кодировки, основанные на стандарте ISO/IEC 8859, также допустимы другие кодировки, например, русские Windows-1251, KOI-8.

Важнейшее обязательное синтаксическое требование заключается в том, что документ имеет только один корневой элемент (англ. root element) (также иногда называемый элемент документа (англ. document element)). Это означает, что текст или другие данные всего документа должны быть расположены между единственным начальным корневым тегом и соответствующим ему конечным тегом. На примере показанном на рисунке 2.18, корневым элементом является тег <data>, а теги <time> - дочернии.

```
<data>
<time>2012-04-24 14:35:09</time>
<time>2012-04-24 14:35:03</time>
<time>2012-04-24 14:34:57</time>
<time>2012-04-24 14:34:50</time>
<time>2012-04-24 14:34:45</time>
<time>2012-04-24 14:34:38</time>
<time>2012-04-24 14:34:33</time>
</data>
```

Рисунок 2.18 – Пример правильно построенного XML документа

Остальная часть этого XML-документа состоит из вложенных элементов, некоторые из которых имеют атрибуты и содержимое. Элемент обычно состоит из открывающего и закрывающего тегов, обрамляющих текст и другие элементы. Открывающий тег состоит из имени элемента в угловых скобках, например, <time>, а закрывающий тег состоит из того же имени в угловых скобках, но перед именем еще добавляется косая черта, например, </time>. Имена элементов, как и имена атрибутов, не могут содержать пробелы, но могут быть на любом языке, поддерживаемом кодировкой XML-документа. Имя может начинаться с буквы, подчеркивания, двоеточия. Остальными символами имени могут быть те же символы, а также цифры, дефис, точка.

Содержимым элемента (англ. content) называется все, что расположено между открывающим и закрывающим тегами, включая текст и другие (вложенные) элементы. На рисунке 2.19 приведен пример XML-элемента, который содержит открывающий тег, закрывающий тег и содержимое элемента.

```
<time>2012-04-24 14:35:09</time>

<time>2012-04-24 14:35:03</time>

<time>2012-04-24 14:34:57</time>

<time>2012-04-24 14:34:50</time>

<time>2012-04-24 14:34:45</time>

<time>2012-04-24 14:34:38</time>

<time>2012-04-24 14:34:33</time>
```

Рисунок 2.19 – Пример XML-элемента, который содержит открывающий тег, закрывающий тег и содержимое элемента

Кроме содержания у элемента могут быть атрибуты — пары имя-значение, добавляемые в открывающий тег после названия элемента. Значения атрибутов всегда заключаются в кавычки (одинарные или двойные), одно и то же имя атрибута не может встречаться дважды в одном элементе. Не рекомендуется использовать разные типы кавычек для значений атрибутов одного тега. [13]

Пример использования XML тегов с атрибутами представлен на рисунке 2.20.

```
<time step=”1”>2012-04-24 14:34:33</time>
```

Рисунок 2.20 – Пример использования XML тегов с атрибутами

2.2.5 Визуализация данных средствами jQplot

JqPlot является JQuery плагином для реализации построения графиков на веб-страницах.

JqPlot требует использования JavaScript – библиотеки JQuery версии 1.4.3 и выше и каскадных таблиц стилей CSS. Подключение этих файлов к веб-странице осуществляется как показано на рисунке 2.21.

```
<script type="text/javascript" language="javascript" src="jquery.min.js"> </ script ><script type="text/javascript" language="javascript" src="jquery.jqplot.min.js"> </ script ><link rel="stylesheet" type="text/css" href="jquery.jqplot.css" />
```

Рисунок 2.21 – Подключение JavaScript – библиотеки JQuery и таблицы стилей CSS.

Добавление контейнера для построение графика производится по средствам добавление в разметку веб-страницы тега с дополнительными параметрами.

Id – является идентификатором блока, и указывает jqPlot в каком именно блоке следует строить график, а параметры height и width задают высоту и ширину области построения.

Пример создания контейнера для построения графика представлен на рисунке 2.22.

```
<div id="graf1" style="height:400px;width:300px; "> </ div>
```

Рисунок 2.22 – Пример создания контейнера для построения графика

Для добавления данных вызывается jqplot плагин с идентификатором graf1 блока и данными заключенными в двойные квадратные скобки. Добавление данных осуществляется так, как показано на рисунке 2.23.

```
$(document).ready(function(){

  var plot1 = $.jqplot ('graf1', [[3,7,9,1,4,6,8,2,5]]);

});
```

Рисунок 2.23 – Пример добавления данных

Результат выполнения представленного кода изображён на рисунке 2.24.

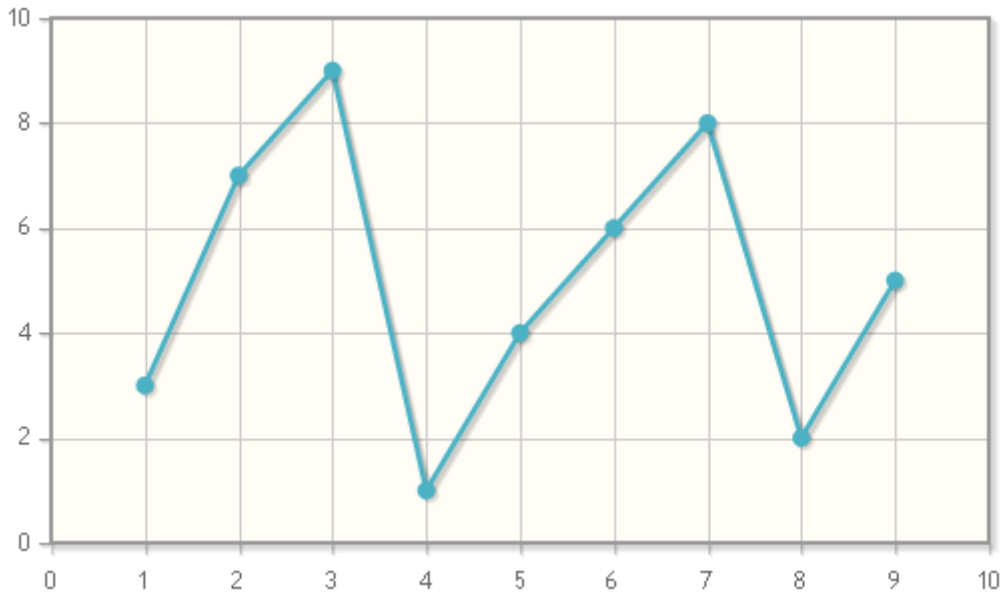

Рисунок 2.24 – Результат построения графика средствами jqplot

Jqplot имеет множество дополнительных плагинов позволяющих расширить возможности построения и отображения информации. [14]

3 Система автоматизации проведения лабораторных экспериментов

В систему автоматизации проведения лабораторных экспериментов измерений входят:

а) беспроводной модуль Sun SPOT и набор датчиков;

б) базовая станция Sun SPOT;

в) стационарный компьютер;

г) сервер meteolab.ru с базой данных MySQL и Internet-сайтом.

Блок схема системы представлена на рисунке 3.1.

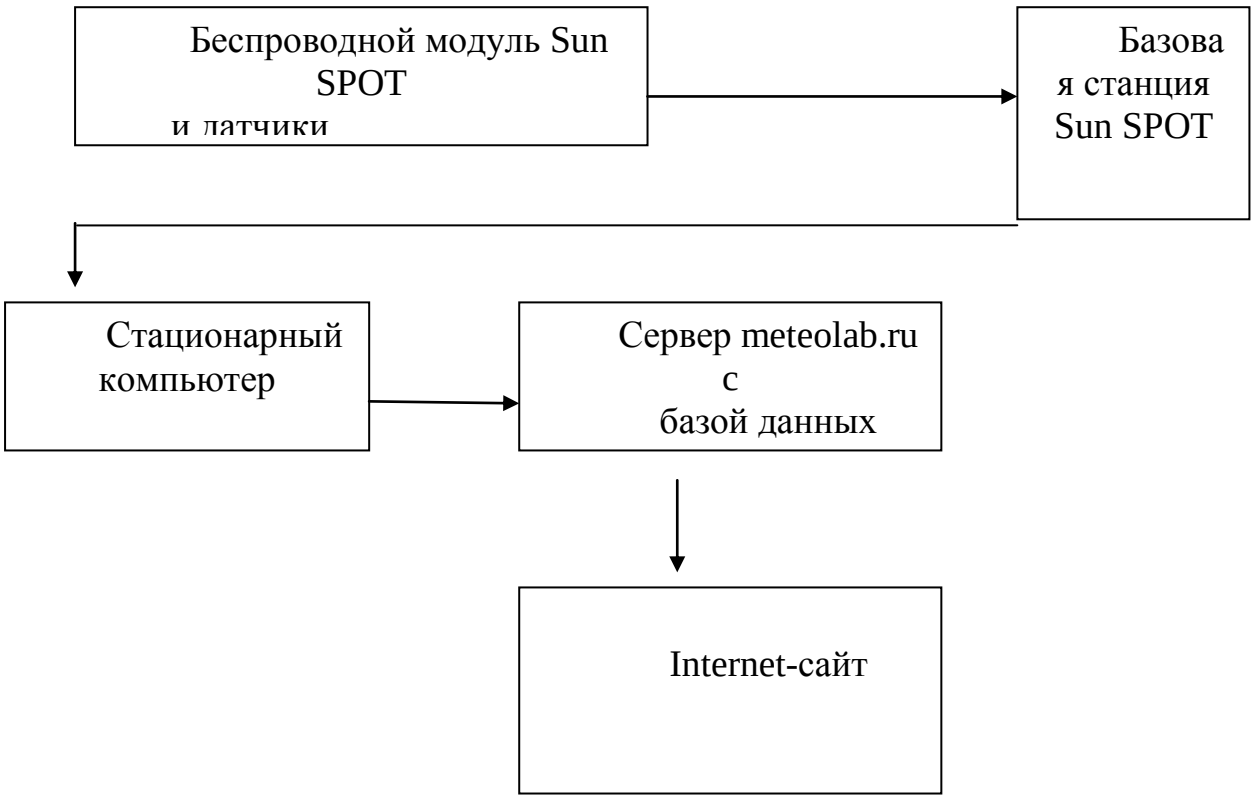

Рисунок 3.1 – Блок схема автоматизированной измерительной системы

3.1 Измерительные датчики автоматизированной системы

Температурные датчики

Практически все температурные датчики, применяемые в современном производстве, используют принцип преобразования измеряемой температуры в электрические сигналы. Такое преобразование основано на том, что электрический сигнал возможно передавать с высокой скоростью на большие расстояния, в электрические же сигналы могут быть преобразованы любые физические величины. Преобразованные в цифровой код эти сигналы могут быть переданы с высокой точностью, а кроме того введены для обработки в компьютер. Такие температурные датчики выполнены на базе термистров.

Термистор — полупроводниковый резистор, электрическое сопротивление которого существенно зависит от температуры.

Для термистора характерны большой температурный коэффициент сопротивления (ТКС) (в десятки раз превышающий этот коэффициент у металлов), простота устройства, способность работать в различных климатических условиях при значительных механических нагрузках, стабильность характеристик во времени.

Терморезистор изготавливают в виде стержней, трубок, дисков, шайб, бусинок и тонких пластинок преимущественно методами порошковой металлургии. Их размеры могут варьироваться в пределах от 1–10 мкм до 1–2 см.

Основными параметрами терморезистора являются: номинальное сопротивление, температурный коэффициент сопротивления, интервал рабочих температур, максимально допустимая мощность рассеяния.

Различают терморезисторы с отрицательным (термисторы) и положительным (позисторы) ТКС. Их ещё называют NTC-термисторы и PTC-термисторы соответственно. У позисторов с ростом температуры растет и сопротивление, а у термисторов —- наоборот: при увеличении температуры, сопротивление падает.

Терморезисторы с отрицательным ТКС изготовляют из смеси поликристаллических оксидов переходных металлов (например, MnO, CoO, NiO, CuO), легированных Ge и Si, полупроводников типа AIII BV, стеклообразных полупроводников и других материалов.

Различают терморезисторы низкотемпературные (рассчитанные на работу при температурах ниже 170 К), среднетемпературные (170–510 К) и высокотемпературные (выше 570 К). Кроме того, существуют терморезисторы, предназначенные для работы при 4.2 К и ниже и при 900–1300 К. Наиболее широко используются среднетемпературные терморезисторы с ТКС от −2.4 до −8.4 %/К и номинальным сопротивлением 1–106 Ом. [18]

Для тестирования автоматизированной системы использовался температурный датчик LM335.

LM335 — это стабилитрон с нормированным температурным коэффициентом напряжения. Схема включения стабилитрона показана на рисунке 3.2.

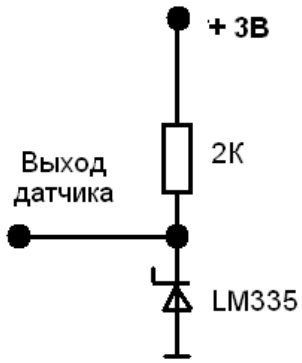

Рисунок 3.2 – Схема включения стабилитрона

Внешний вид датчика представлен на рисунке 3.3.

Рисунок 3.3 – Внешний вид датчика LM335

Технические характеристики температурного датчика LM335:

а) градуировка: 10 мВ/ K;
б) точность: 0.5 °C;
в) напряжение питания: 3 В;
г) рабочая температура: -40 – 100 °C ;
д) тип корпуса: TO-92-3pins;
е) Размер: 5x5.03 мм.

3.2 Микропроцессорная система регистрации данных Sun Spot

Система микроконтроллеров Sun Spot (Sun Small Programmable Object Technology) содержит базовую станцию, подключаемую к компьютеру, и два переносных модуля, связанных со станцией по радиоканалу стандарта IEEE 802.15.4 (2.4 ГГц). В состав модуля входят: 32-битный процессор ARM920T, блок памяти (512 КбSDRAM, 4 Мб Flash), плата датчиков и литий-ионная аккумуляторная батарея. Плата датчиков содержит трехосевой акселерометр, термочувствительный элемент, светочувствительный элемент, 8 RGB-светодиодов, 6 аналоговых входов, 2 быстродействующих выключателя, 5 контактов ввода/вывода общего назначения и 4 контакта для коммутации цепей повышенной мощности. Работает устройство под управлением виртуальной машины Squawk Java VM(Project Squawk) . Рабочая температура Sun Spot – от -20 до +60 градусов.

Разработка программного обеспечения осуществляется в среде NetBeans набором Ant-скриптов, обеспечивающих поддержку устройств Sun SPOT.[15]

3.3 Интегрированная среда разработки программ NetBeans

NetBeans — Бесплатная интегрированная среда разработки с открытым исходным кодом для разработчиков программного обеспечения. Среда предоставляет все средства, необходимые для создания профессиональных настольных, корпоративных, мобильных и веб-приложений на платформе Java, а также C/C++, PHP, JavaScript , Groovy и другие.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun

JDK или J2EE SDK подходящей версии. Среда разработки NetBeans по умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 Netbeans поддерживает разработку для мобильных платформ J2ME, C++ (только g++) и PHP без установки дополнительных компонентов.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

По качеству и возможностям последние версии NetBeans IDE не уступают лучшим коммерческим (платным) интегрированным средам разработки для языка Java, таким, как IntelliJ IDEA, поддерживая рефакторинг, профилирование, выделение синтаксических конструкций цветом, автодополнение набираемых конструкций на лету, множество предопределённых шаблонов кода и другое.

В версии NetBeans IDE 6.1 декларируется поддержка UML, SOA, языка программирования Ruby (включая поддержку Ruby on Rails), а также средства для создания приложений на J2ME для мобильных телефонов. В версии 6.5 добавлена поддержка языка PHP. Также для тестирования выложен модуль поддержки Python.

NetBeans IDE поддерживает плагины, позволяя разработчикам расширять возможности среды. Одним из самых популярных плагинов является мощный дизайнер отчётов iReport (основанный на библиотеке JasperReports).

На идеях, технологиях и в значительной части на исходном коде NetBeans IDE базируются предлагаемые фирмой Sun коммерческие интегрированные среды разработки для Java — Sun Java Studio Creator, Sun Java Studio Enterprise и Sun Studio (для ведения разработки на C, C++ или Фортран). Сравнительно недавно Sun стала предлагать эти среды разработки

бесплатно для зарегистрировавшихся в Sun Developer Network (SDN) разработчиков, сама же регистрация на сайте бесплатна и не требует никаких предварительных условий, кроме согласия с лицензией CDDL.

NetBeans IDE доступна в виде готовых дистрибутивов (прекомпилированных бинарных файлов) для платформ Microsoft Windows, Linux, FreeBSD, Mac OS X, OpenSolaris и Solaris (как для SPARC, так и для x86 — Intel и AMD). Для всех остальных платформ доступна возможность скомпилировать NetBeans самостоятельно из исходных текстов.

В релизе NetBeans IDE 6.7 была добавлена интеграция с Project Kenai, поддержка языка Groovy и веб-фреймворка Grails. В версии 6.8 — поддержка PHP-фреймворка Symfony, а в 6.9 — Zend Framework. [16]

3.4 Программные средства автоматизированной системы

Специально для измерительного комплекса было разработано специализированное программное обеспечение на таких языках программирования как java, javascript и php.

Программное обеспечение делится на 3 части:

а) программное обеспечение Sun Spot’a – данный программный модуль позволяет получать данные с подключенных к микроконтроллеру датчиков и отправлять их на базовую станцию, подключенную к стационарному компьютеру. Программный код приведен в Приложении А.

б) программное обеспечение компьютера включает в себя:

1. набор драйверов для работы с Sun Spot.

2. Программный модуль решающий следующие задачи: приём данных с базовой станции, обработки полученных данных, запись данных в базу данных на сервере. Программный код приведён в Приложении Б.

в) набор серверных скриптов, позволяющий реализовать визуализацию данных на сайте. Программный код приведён в Приложении В.

3.5 Структура базы данных результатов лабораторных экспериментов

В качестве базы данных используется реляционная база данных MySQL.

База данных имеет табличный вид как показано в таблице 3.1

время	темп. датчика	темп. спота	напряжение
2012-05-29 15:39:11	26.7	24.9	2.9970703125
2012-05-29 15:39:05	26.7	25.2	2.9970703125
2012-05-29 15:38:47	26.7	24.9	2.9970703125

Таблица 3.1 – пример таблицы в базе данных.

4 Результаты эксплуатации автоматизированной системы

4.1 Результаты калибровки датчика температуры

Температурный датчик LM335 имеет диапазон напряжение от 0 до 3 Вольт, 0 градусов Цельсия соответствует напряжению 2,73 В. По этим данным была выведена формула (4.1) преобразования напряжения (в Вольтах) в температуру (в Цельсия).

$$A0t=(A0-2.73)/0.01$$

где A0t – температура, A0 – выходное напряжение датчика

Формула 4.1 – Формула преобразования напряжения в температуру

Преобразование напряжения в температуру выполняет программный модуль ПЭВМ.

4.2 Пример результатов регистрации температуры

Пример регистрации температуры приведён в таблице 4.1.

время	темп. датчика	темп. спота	напряжение
2012-05-29 15:39:11	26.7	24.9	2.9970703125
2012-05-29 15:39:05	26.6	25.2	2.9970703125
2012-05-29 15:38:47	26.7	24.9	2.9970703125
2012-05-29 15:38:41	26.7	24.9	2.9970703125
2012-05-29 15:38:35	26.5	25.5	2.9970703125
2012-05-29 15:38:29	26.7	24.9	2.9970703125
2012-05-29 15:38:24	26.6	25.2	2.9970703125
2012-05-29 15:38:17	26.8	24.9	2.9970703125
2012-05-29 15:38:11	26.7	25.2	2.9970703125
2012-05-29 15:38:05	26.7	24.9	2.9970703125

Таблица 4.1 - Пример регистрации температуры

В столбце «время» приведена дата и время регистрации данных.

В столбце «темп. датчика» приведена температура воздуха, полученная с помощью температурного датчика.

В столбце «темп. спота» приведена температура воздуха полученная с помощью встроенного датчика микроконтроллера Sun Spot.

В столбце «напряжение» приведено напряжение датчика температуры в вольтах, соответствующее температуре указанной в столбце «темп. датчика».

4.3 Интернет-сайт для доступа к данным лабораторных экспериментов

Для визуализации данных создана страница на Internet-сайте meteolab.ru.

Сайт позволяет удалённо проводить наблюдения процесса измерений, а так же выбирать и анализировать данные за нужный период. Данные представлены как в графическом виде, так и в табличном. Так же на сайте приведены статистические данные: максимальные и минимальные значения, среднеквадратическое отклонение.

Изображение графического представления данных приведено на рисунке 4.1.

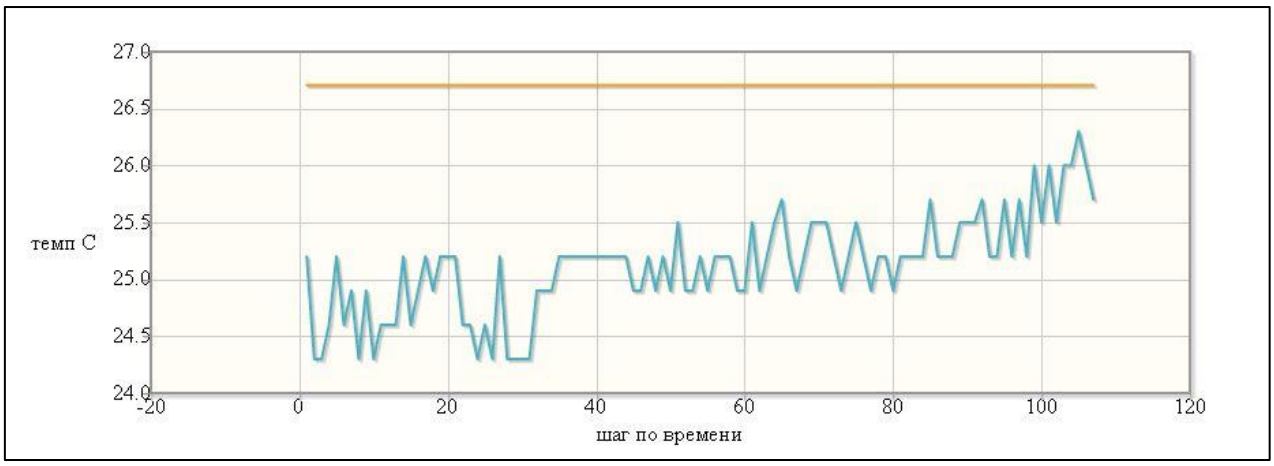

Рисунок 4.1 – Графическое представление данных

По оси Y расположена температурная шкала, диапазон подстраивается автоматически, в зависимости от разброса значений температуры. Ось X – шаг по времени, а каждый шаг по времени соответствует одному шагу регистрации температуры.

В зависимости от задачи, график можно модернизировать. Например, добавить функцию масштабирования, которая позволяет приближать

определённый участок графика, что упрощает работу при большом объеме данных.

Данные к графику добавляются в процессе измерений, и проходят в «фоновом» режиме, что облегчает работу пользователя, так как не нужно обновлять internet-страницу что бы увидеть обновлённые данные.

Изображение табличного представления данных на Internet странице приведено на рисунке 4.2.

время	темп. датчика	темп. спота	напряжение
2012-06-12 15:42:27	26.7	25.7	2.9970703125
2012-06-12 15:42:23	26.7	26	2.9970703125
2012-06-12 15:42:15	26.7	26.3	2.9970703125
2012-06-12 15:42:09	26.7	26	2.9970703125
2012-06-12 15:42:03	26.7	26	2.9970703125
2012-06-12 15:41:57	26.7	25.5	2.9970703125
2012-06-12 15:41:51	26.7	26	2.9970703125
2012-06-12 15:41:45	26.7	25.5	2.9970703125
2012-06-12 15:41:39	26.7	26	2.9970703125
2012-06-12 15:41:33	26.7	25.2	2.9970703125

Рисунок 4.2 – Табличное представление данных

В таблице отображается дата и время измерения, температура полученная с помощью температурного датчика LM335, температура на встроенном в микроконтроллер датчике, напряжение соответствующее температуре температурного датчика LM335.

Для того, что бы запросить данные из базы данных следует выбрать нужную дату и время и нажать кнопку «запросить данные» как показано на рисунке 4.3.

Рисунок 4.3 – Пример запроса данных из базы данных на internet-странице

Общий вид internet страницы представлен на рисунке 4.4.

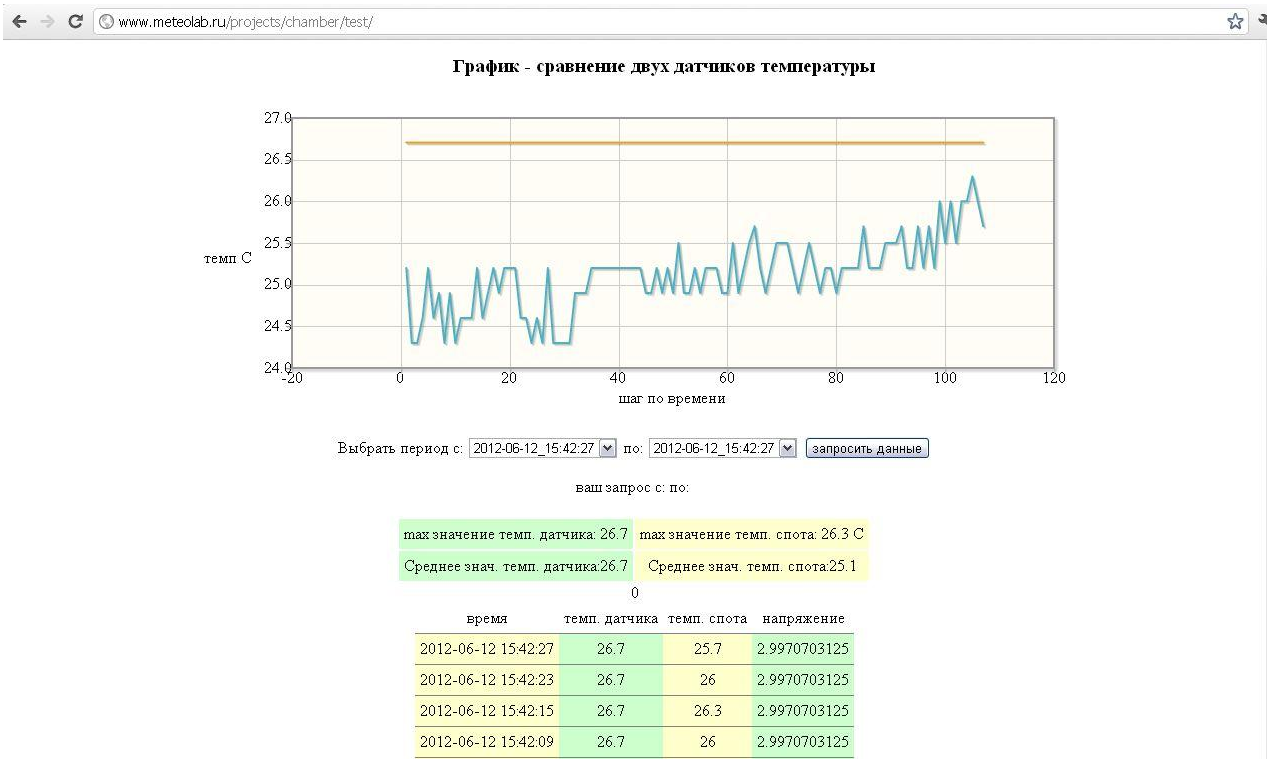

Рисунок 4.4 - Общий вид internet страницы представления данных

Заключение

В заключение можно сказать, что поставленные задачи решены. В процессе выполнения дипломной работы получены следующие результаты:

1. рассмотрены принципы построения автоматизированных измерительных систем;
2. осуществлен выбор информационных технологий, позволяющих реализовать систему автоматизации измерений;
3. создана система автоматической регистрации данных измерений в виде Java-приложения для микропроцессорного устройства SunSpot;
4. создана система обработки данных измерений в виде Java-приложения для ПЭВМ;
5. создана система хранения данных измерений с использованием СУБД MySQL;
6. создана система представления данных измерений в виде Интернет-сайта с графическим и табличным представлением данных.

Данная система автоматизации процессов лабораторных измерений является работоспособной. Измерения, обработка и представление данных происходят полностью автоматически, а доступ к данным можно получать из любого места, где есть доступ к сети Internet. Данные представляются пользователю в графическом и табличном виде, выборку данных возможно осуществлять по запросу, а так же получать статистические данные, что существенно облегчает работу с данными.

В дальнейшем планируется использовать систему для автоматизации измерений в климатической камере и лабораторной установке для

выращивания кристаллов льда. Эти лабораторные установки создаются в Российском государственном гидрометеорологическом университете на кафедре экспериментальной физики атмосферы в лаборатории метеотехнологий.

Данную систему можно в короткие сроки адаптировать под решение нужной задачи, таким образом, она может быть использована не только в упомянутых выше лабораторных установках, но и при решении других задач, которые нуждаются в автоматизации измерений.

Список используемых источников

1) Хоровиц П., Хилл У. Искусство схемотехники[текст]. В 3-х т: Т. 2. Пер. с англ. — 4-е изд., перераб. и доп.— М.: Мир, 1993. — 371 с.
2) Компания softelectro / [электронный ресурс]. – Режим доступа: http://www.softelectro.ru .
3) Терри Оглтри Модернизация и ремонт сетей [текст] . — 4-е изд. — М.: Вильямс, 2005. — С. 1328.
4) Дуглас Камер Сети TCP/IP, том 1. Принципы, протоколы и структура [текст]. — М.: Вильямс, 2003. — С. 880.
5) Компания IEE Standards Association / [электронный ресурс]. – Режим доступа: http://standards.ieee.org .
6) В. Васвани. MySQL: использование и администрирование [текст]. — М.: Питер, 2011. — 368 с
7) Кузнецов Максим, Симдянов Игорь. MySQL 5. В подлиннике [текст]. — Спб.: БХВ-Петербург, 2006. — С. 1024.
8) Питер Лабберс, Брайан Олберс, Фрэнк Салим. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений [текст]. — М.: Вильямс, 2011. — 272 с.
9) Фримен Эрик, Фримен Элизабет. Изучаем HTML, XHTML и CSS [текст]. — 1-е изд. — М.: Питер, 2010. — С. 656.
10) Эрик А. Мейер. CSS-каскадные таблицы стилей: подробное руководство [текст]. — М.: Символ, 2006. — 576 с.
11) Дейв Крейн, Бер Бибо, Джордон Сонневельд. Ajax на практике [текст]. — М.: Вильямс, 2007.

12) Дейв Крейн, Эрик Паскарелло, Даррен Джеймс. AJAX в действии: технология — Asynchronous JavaScript and XML [текст]. — М.: Вильямс, 2006.

13) Дэвид Хантер, Джефф Рафтер и др. XML. Базовый курс [текст]. — М.: Вильямс, 2009. — 1344 с.

14) Компания jqplot / [электронный ресурс]. – Режим доступа: http://www.jqplot.com/ .

15) Компания Oracle Labs / [электронный ресурс]. – Режим доступа: http://www.sunspotworld.com/ .

16) Монахов Вадим Язык программирования Java и среда NetBeans [текст]. — СПб.: «БХВ-Петербург», 2008. — С. 640

17) Кузнецов С. Д. Основы баз данных [текст]. — 2-е изд. — М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с

18) Пасынков В. В., Чиркин Л. К. Полупроводниковые приборы: Учебник для вузов [текст]. — 4-е перераб. и доп. изд. — М.: Высшая школа, 1987. — С. 401-407. — 479 с.

Приложение А

Программное обеспечение микроконтроллера Sun Spot.

```
package org.sunspotworld.demo;
import com.sun.spot.resources.transducers.ITemperatureInput;
import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.resources.Resources;
import com.sun.spot.resources.transducers.IAnalogInput;
import com.sun.spot.resources.transducers.ITriColorLED;
import com.sun.spot.service.BootloaderListenerService;
import com.sun.spot.util.Utils;
import java.util.Calendar;
import java.util.Date;
import javax.microedition.io.*;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;


public class SensorSampler extends MIDlet {
    private static final int DATA_SINK_PORT = 67;

    protected void startApp() throws MIDletStateChangeException {
        RadiogramConnection rCon = null; //radiogramma
        Datagram dg = null; // datagramma
        String ourAddress = System.getProperty("IEEE_ADDRESS"); //our adress
```

```
        long now = 0L;
        double A0=0.0;
        double A1=0.0;
        double A2=0.0;
        double A3=0.0;
        double temp=0.0;
        Calendar cal = Calendar.getInstance();//calendar
        String ts = null;
        ITemperatureInput tempSensor = (ITemperatureInput)
Resources.lookup(ITemperatureInput.class);
        ITriColorLED led = (ITriColorLED)Resources.lookup(ITriColorLED.class, "LED1");
        IAnalogInput iai0 = (IAnalogInput) Resources.lookup(IAnalogInput.class, "A0");
         IAnalogInput iai1 = (IAnalogInput) Resources.lookup(IAnalogInput.class, "A1");
          IAnalogInput iai2 = (IAnalogInput) Resources.lookup(IAnalogInput.class, "A2");
           IAnalogInput iai3 = (IAnalogInput) Resources.lookup(IAnalogInput.class, "A3");
        System.out.println("Start application. Our Adress is: " +
                ourAddress + " ...");

           // Listen for downloads/commands over USB connection
           BootloaderListenerService.getInstance().start();

        try {
           // Open up a broadcast connection at the data sink port
           // where the 'on Desktop' portion of this demo is listening
           rCon = (RadiogramConnection) Connector.open(
                   "radiogram://broadcast:" + DATA_SINK_PORT);
           dg = rCon.newDatagram(rCon.getMaximumLength());
        } catch (Exception e) {
           System.err.println("Caught " + e +
                   " in connection initialization.");
           notifyDestroyed();
        }

        while (true) {
```

```
try {
   // Get the current time and sensor reading
   now = System.currentTimeMillis(); //Set time
   A0 = iai0.getVoltage();
   A1 = iai1.getVoltage();
   A2 = iai2.getVoltage();
   A3 = iai3.getVoltage();
   temp = tempSensor.getCelsius();

   // Flash an LED to indicate a sampling event
   led.setRGB(0, 255, 0);// befor print data, LED light green color
   led.setOn();// us LED
   Utils.sleep(50);//sleep LED
   led.setOff(); // LED off
   // write in ts month,day, hour i tak dalee
   cal.setTime(new Date(now));
   ts = cal.get(Calendar.YEAR) + "-" +
         (1 + cal.get(Calendar.MONTH)) + "-" +
         cal.get(Calendar.DAY_OF_MONTH) + " " +
         cal.get(Calendar.HOUR_OF_DAY) + ":" +
         cal.get(Calendar.MINUTE) + ":" +
         cal.get(Calendar.SECOND);
   //print ts and sensor light
   System.out.println("\n\ntime: "+ts+"\n");
   System.out.println("A0: "+A0+"V\n");
   System.out.println("A1: "+A1+"V\n");
   System.out.println("A2: "+A2+"V\n");
   System.out.println("A3: "+A3+"V\n");


   // Package our identifier, timestamp and sensor reading
   // into a radio datagram and send it.
   dg.reset();
   dg.writeUTF(ourAddress);
```

```
                dg.writeUTF(ts);
                dg.writeDouble(A0);
                dg.writeDouble(A1);
                dg.writeDouble(A2);
                dg.writeDouble(A3);
                dg.writeDouble(temp);
                rCon.send(dg);

                // Go to sleep to conserve battery
                Utils.sleep(6000 - (System.currentTimeMillis() - now));
            } catch (Exception e) {
                System.err.println("Caught " + e +
                        " while collecting/sending sensor sample.");
            }
        }
    }

    protected void pauseApp() {
        // This will never be called by the Squawk VM
    }

    protected void destroyApp(boolean arg0) throws MIDletStateChangeException {
        // Only called if startApp throws any exception other than MIDletStateChangeException
    }
}
```

Приложение Б

Программное обеспечение персонального компьютера.

```
package org.sunspotworld.demo;

import com.sun.spot.io.j2me.radiogram.*;
import com.sun.spot.peripheral.TimeoutException;

import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.text.DateFormat;
import java.util.Calendar;
import java.util.Date;
import javax.microedition.io.*;



public class DatabaseDemoHostApplication {
    // Broadcast port on which we listen for sensor samples
    private static final int DATA_SINK_PORT = 67;
   //DB connection options
    private static final String DATABASE_URL = "jdbc:mysql://meteolab.ru:3306/";
    private static final String DATABASE_NAME = "sunspot";
    private static final String DATABASE_USER = "drozdov";
```

```
    private static final String DATABASE_PASSWORD = "sun-spot-pass";
    private static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
    private static final String DATA_TABLE_NAME = "spottest";

    private static final int SAMPLING_DURATION = 840000;
    private static final int CONNECTION_TIMEOUT = 10000;

     Statement stmt;
    java.sql.Connection dbCon = null;
    private RadiogramConnection rCon = null;
    //main class.
    private void run() throws Exception {
        try {
            setUp();
           collectData();
            } catch (Exception e) {
            e.printStackTrace();
        } finally {
            tearDown();
        }
    }

    private void setUp() throws Exception {
        System.out.println("Database demo application starting ... ");
       try {
            Class.forName(JDBC_DRIVER).newInstance ();
            String url = DATABASE_URL + DATABASE_NAME;
            dbCon = DriverManager.getConnection(url,
DATABASE_USER,DATABASE_PASSWORD);
            System.out.println("адресс: " + url);
            System.out.println("соединение: " + dbCon);
        } catch (Exception e) {
            System.err.println("ошибка соединения с БД: " + e);
```

```
            throw e;
        }

        try {
            rCon = (RadiogramConnection) Connector.open("radiogram://:" +
                    DATA_SINK_PORT);
            // wait a finite time for data to arrive whenever we call receive()
            rCon.setTimeout(CONNECTION_TIMEOUT);
            System.out.println("Подключение по радио прошло успешно!");
        } catch (Exception e) {
             System.err.println("ошибка соедиение с SPOT " + e.getMessage());
             throw e;
        }
    }

    public void collectData() throws Exception {

        String id = null;
        String ts = null;
        Date time = null;
        Double A0 = 0.0;
        Double A1 = 0.0;
        Double A2 = 0.0;
        Double A3 = 0.0;
        Double temp = 0.0;
        int sampleCnt = 0;
        Datagram dg = null;
        long start = 0L;

        System.out.println("Сеанс будет длиться " + (SAMPLING_DURATION/1000) +"
секунд");
        start = System.currentTimeMillis();
        dg = rCon.newDatagram(rCon.getMaximumLength());
        // Main data collection loop
```

```
        while ((System.currentTimeMillis() - start) < SAMPLING_DURATION) {
          try {
            // Read sensor sample received over the radio
           // time = java.util.Calendar.getInstance().getTime();

            String day;
            Calendar cal = Calendar.getInstance();
            day = cal.get(Calendar.YEAR) + "-" +
                  (1 + cal.get(Calendar.MONTH)) + "-" +
                  cal.get(Calendar.DAY_OF_MONTH) + " " +
                  cal.get(Calendar.HOUR_OF_DAY) + ":" +
                  cal.get(Calendar.MINUTE) + ":" +
                  cal.get(Calendar.SECOND);
            rCon.receive(dg);
            id = dg.readUTF();  // read sender's Id
            ts = dg.readUTF();  // read time stamp for the reading
            A0 = dg.readDouble();
            A1 = dg.readDouble();
            A2 = dg.readDouble();
            A3 = dg.readDouble();
            temp = dg.readDouble();
            Double A0t;
            A0t=(A0-2.73)/0.01;
            System.out.println("Измерение №"+sampleCnt+" время: "+day+"\n");
            System.out.println("Температура с A0 C:     "+A0t+"\n");
            System.out.println("Температура на споте C: "+temp+"\n");
            System.out.println("напряжение на A0:       "+A0+"\n");
           try{
              stmt = (Statement) dbCon.createStatement();
              String query = "INSERT INTO temp (time,a0t,tsp,a0v) VALUES(\'" + day + "\'," 
+A0t + "," +temp + "," +A0 + ")";
              //System.out.println(query);
              stmt.executeUpdate(query);
            } catch (Exception exc) {
```

```
            System.err.println("ошибка записи в БД: "+exc);

            throw exc;
          }
/*
            try{
              stmt = (Statement) dbCon.createStatement();
              stmt.executeUpdate("INSERT INTO " + DATA_TABLE_NAME
+"(time,a0,a1,a2,a3) VALUES(\'" + day + "\'," +A0t + "," +A1 + "," +A2 + "," +A3 + ")");
            } catch (Exception exc) {
              System.err.println("ошибка записи в БД: "+exc);

              throw exc;
          }
 */
            sampleCnt++;
          } catch (TimeoutException e) {
            System.err.println("!");

            throw e;
          } catch (Exception e) {
            System.err.println("Ошибка вывода данных " + e);
            throw e;
          }
      }
      System.out.println("*****************************************************\n");
      System.out.println("\n Зарегестрировано " + sampleCnt + " измерения(ий)\n");
      System.out.println("*****************************************************\n");
    }


    public void tearDown() throws Exception {
      if (dbCon != null) dbCon.close();
      if (rCon != null) rCon.close();
```

```
        System.exit(0);
    }

    /**
     * Start up the host application.
     *
     * @param args any command line arguments
     */
    public static void main(String[] args) throws Exception {
        DatabaseDemoHostApplication app = new DatabaseDemoHostApplication();
        app.run();
    }
}
```

Приложение В

Программный код асинхронного Ajax запроса графического представления данных.

```
<html>
<haed>
<title></title>
<script language="javascript" type="text/javascript" src="plugins/jquery.min.js"></script>
<script language="javascript" type="text/javascript" src="plugins/jqplot.json2.min.js"></script>
<script language="javascript" type="text/javascript" src="plugins/jquery.jqplot.min.js"></script>
<script language="javascript" type="text/javascript"
src="plugins/jqplot.canvasAxisLabelRenderer.min.js"></script>
<script language="javascript" type="text/javascript" src="plugins/jqplot.cursor.min.js"></script>
<script language="javascript" type="text/javascript"
src="plugins/jqplot.pointLabels.min.js"></script>
<link rel="stylesheet" type="text/css" href="plugins/jquery.jqplot.css" />
<link rel="stylesheet" type="text/css" href="style.css" />
<script type="text/javascript">
function timer() {
                    // Здесь вызов метода, отправляющего запрос на сервер

                    parsing();

                    // Устанавливаем время следующего обращения к этому же методу
                    через 1000 мс = 1 с

                    document.getElementById("chart1").innerHTML = "";

                    setTimeout("timer()", 3000);
```

```
}

function server() {

// Тут на самом деле должна быть отправка Ajax-запроса

alert("Ответ сервера - OK");

}

// начало парсинга

function parsing() {

if (window.XMLHttpRequest) {

xhr = new XMLHttpRequest();

} else {

if (window.ActiveXObject) {

xhr = new ActiveXObject("Microsoft.XMLHTTP");

}

}

if (xhr) {

xhr.onreadystatechange = setNamesArray;

xhr.open("GET", "data.php", true);

xhr.send(null);

} else {

alert("Ошибка создания запроса AJAX");
```

```
}

function setNamesArray() {

if (xhr.readyState == 4) {

//document.write("статус 4 ok!<br>");

if (xhr.status == 200) {

//document.write("статус 200 ok!<br>");

if (xhr.responseXML) { //проверяем xml ли это

//document.write("разбор идёт..<blabel: "X Axis",r>");

var line1=[[]];//temp

var line2=[[]];//volt

var line3=[[]];//spot temp

var alltemp = xhr.responseXML.getElementsByTagName("temp");

var allspot = xhr.responseXML.getElementsByTagName("spot");

var allvolt = xhr.responseXML.getElementsByTagName("volt");

var alltime = xhr.responseXML.getElementsByTagName("time");

//document.write("кол-во элементов temp: "+alltemp.length+"<br>");

for (var i=0; i<alltemp.length; i++) {

var tp = parseFloat(alltemp[i].getAttribute("t"));//temp

var vt = allvolt[i].getAttribute("v");//volt

var sp = parseFloat(allspot[i].getAttribute("sp"));//temp spot
```

```
var tm = alltime[i].getAttribute("tm");//time

line1[i] = [i+1, tp];

line3[i] = [i+1, sp];

line2[i] = [i, vt*1];

//document.write(line1[i]+"<br>");

}

$(document).ready(function(){

var plot1 = $.jqplot('chart1',[line3,line1],{

//title: 'График зависимости температуры от напряжения',

axes:{

xaxis:{

label: "шаг по времени",

//renderer:$.jqplot.DateAxisRenderer,

//tickOptions:{formatString:'%H:%M:%S'},

},

yaxis:{

tickOptions:{formatString:' %.1f '}

}

},

series:[
```

```
{
lineWidth:2,
showMarker:false
},
{
lineWidth:2,
showMarker:false
},
],
legend: {
show: true,
placement: 'outsideGrid'
},
cursor: {
show: true,
tooltipLocation:'sw'
}
});
});
/
```

```
                    }
                    else document.write("this is not XML<br>");
                    }
                    }
                    }
                    }
                    // конец парсинга
                    </script>
</head>
<body onload="timer()">
<div align="center">
<div id="chart1" style="height:300px;width:800px;"> </div>
</div>
</html>
```