Министерство образования и науки Российской Федерации
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ (НИ ТГУ)
Факультет информатики
Кафедра прикладной информатики

ДОПУСТИТЬ К ЗАЩИТЕ В ГЭК

Руководитель ООП
д-р техн. наук, профессор
_____ С.П.Сущенко
«_____»_____2016 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

РЕАЛИЗАЦИЯ СИСТЕМЫ АВТОМАТИЗИРОВАННОГО УПРАВЛЕНИЯ СЕРВИСНЫМИ ФУНКЦИЯМИ ЖИЛОГО ПОМЕЩЕНИЯ

по основной образовательной программе подготовки бакалавров
направление подготовки 09.03.03 - Прикладная информатика

Мочалов Валентин Сергеевич

Руководитель ВКР,
канд. физ.-мат. наук
_____ А. В. Ченцов
подпись
«_____»_____ 2016 г.

Автор работы
студент группы №1422
_____ В. С. Мочалов
подпись

Томск-2016

РЕФЕРАТ

Выпускная квалификационная работа, 49с., 23 рис., 12 источников, 1 приложение.

АВТОМАТИЗАЦИЯ УПРАВЛЕНИЯ ПОМЕЩЕНИЯМИ, УМНЫЙ ДОМ, RASPBERRY PI, ARDUINO.

Цель работы – разработать систему управления «умным домом».

Результат работы – реализован базовый функционал системы автоматизированного управления сервисными функциями жилого помещения с возможностями дальнейшего расширения, определено направление дальнейшего развития.

СОДЕРЖАНИЕ

Введение

В современном мире с каждым годом все увеличивается тенденция к упрощению и автоматизации повседневных задач. В обиход современного человека плотно вошли технологии удаленного и бесконтактного управления. Эти технологии помогают не только экономить время, но и позволяют не зависеть от местонахождения. Например, вы хотите всегда быть уверенным в сохранности своего имущества в свое отсутствие, или не хотите вставать с дивана, чтобы включить или выключить свет. В этом Вам помогут автоматизированные системы. Рост популярности автоматизированных систем управления сервисными функциями жилых помещений, таких как «умный дом», обусловлен стремлением человека к комфорту и удобству. Дополнительной привлекательностью является безопасность, будь то противопожарная система или сигнализация с дистанционным оповещением.

«Умный дом» является современным инструментом повышения комфорта и уровня жизни, так как часть процессов происходит автоматически, а остальной частью можно управлять удаленно, что делает ее актуальной для изучения и совершенствования[1][2].

Целью данной работы является разработка системы управления “умным домом”, которая позволит автоматизировать повседневные задачи, управляя как устройствами и датчиками своей разработки, так и устройствами сторонних производителей в режиме «черного ящика».

Задачи, которые были решены в рамках данного проекта:

- анализ существующих систем и готовых решений, выявление их достоинств и недостатков;

- разработка универсального защищенного протокола управления устройствами «умного дома»;
- реализация программного комплекса, управляющего собственными устройствами и устройствами сторонних производителей.

Глава 1. Теоретические основы

1.1. Обзорно-аналитическая часть

В этом разделе будут рассмотрены существующие на рынке реализации идеи системы «умный дом». В проанализированных системах выявим достоинства и недостатки.

«Умный дом» технически является совокупностью систем устройств и датчиков, объединенных в общую структуру некоторой общей шиной и управляемой центром управления. Эти системы бывают следующих видов:

- система микроклимата (отопление, вентиляция, кондиционирование, увлажнение и т.д.);
- система безопасности (охранная, пожарная сигнализации, система доступа, контроль утечек газа и воды, видеонаблюдение);
- система электропитания (резервные системы, контроль перегрузки электросети, система освещения, система управления нагрузочными элементами и т.д.);
- система связи (телефон, локальная сеть, интернет, SMS оповещения);
- система удаленного управления.

На данный момент наиболее широко используются следующие перспективные технологии объединения на общей шине систем и устройств «умного дома»:

- LanDriver – универсальная платформа построения шинных систем управления, используемая в автоматизации зданий. Предназначена для управления внутренними и внешними системами. Система LanDriver состоит из центрального

контроллера и модулей, подключенных между собой шиной (стандарт RS-485). К модулям подключается управляемое оборудование. Ориентирована на промышленное использование.

- EIB/KNX – Система EIB распределенная, управление осуществляется в пределах устройств. Устройства обмениваются информацией по шине EIB в соответствии с собственным протоколом. Система, построенная на EIB, автономна и не зависит от работоспособности центрального контроллера.
- AMX разрабатывает программно-аппаратные средства удаленного управления медиасистемой, системой видеонаблюдения и широкого спектра датчиков. Протоколы передачи данных закрыты. Изначально применялась собственная шина передачи данных, в новой линейке оборудования применяются стандартные протоколы Ethernet и Wi-Fi, имеются частично развитые технологии сопряжения с системами EIB, LON и других.
- Z-wave, технология беспроводной передачи данных, разработанная для домашней автоматизации. В технологии Z-wave применяются маломощные и миниатюрные радиомодули, встраиваемые в бытовую технику. В основе технологии лежит ячеистая технология, в которой каждый узел является приемником и передатчиком, т.е. при возникновении препятствия сигнал пойдет через соседние узлы сети, находящиеся в радиусе действия. Еще одним преимуществом является малое энергопотребление, что вместе с малыми размерами позволяет встраивать Z-wave в различные бытовые приборы.

Каждая из перечисленных выше систем использует для обмена данных свои или сторонние протоколы, в зависимости от поставленных к системе задач.

Стоит отметить, что большинство систем и технологий автоматизации помещений закрыты.

Рассмотрим некоторые варианты готовых систем, основанных на вышеперечисленных технологиях.

NetPing

Отечественная компания «Alentis Electronics» является разработчиком и производителем устройства мониторинга окружающей среды NetPing. Основная сфера применения – удаленный контроль и мониторинг устройств в доме и офисе. Использует в качестве базового протокола Ethernet.

Задачи, решаемые при помощи устройства NetPing:

- удаленное управление электропитанием;
- управление безопасностью и отслеживание чрезвычайных происшествий, используя датчики дыма, протечки воды, утечки газа, антивандальные системы, управление камерами видеонаблюдения;
- управление микроклиматом при помощи датчиков температуры, влажности и управление кондиционером через инфракрасный порт;
- управление АТС по порту RS-232;
- дистанционное изменение настроек в зависимости от ситуации;
- отправка уведомлений о неполадках или других важных событиях посредством SMS, электронной почты;
- доступ к системе в реальном времени через HTTP или SNMP;
- управление освещением и другими бытовыми приборами по расписанию;

Рисунок 1 - общая схема системы NetPing

Устройства NetPing позволяют подключить до 16 датчиков на одно устройство. Благодаря встроенному Web-серверу контроль и управление осуществляется через браузер.

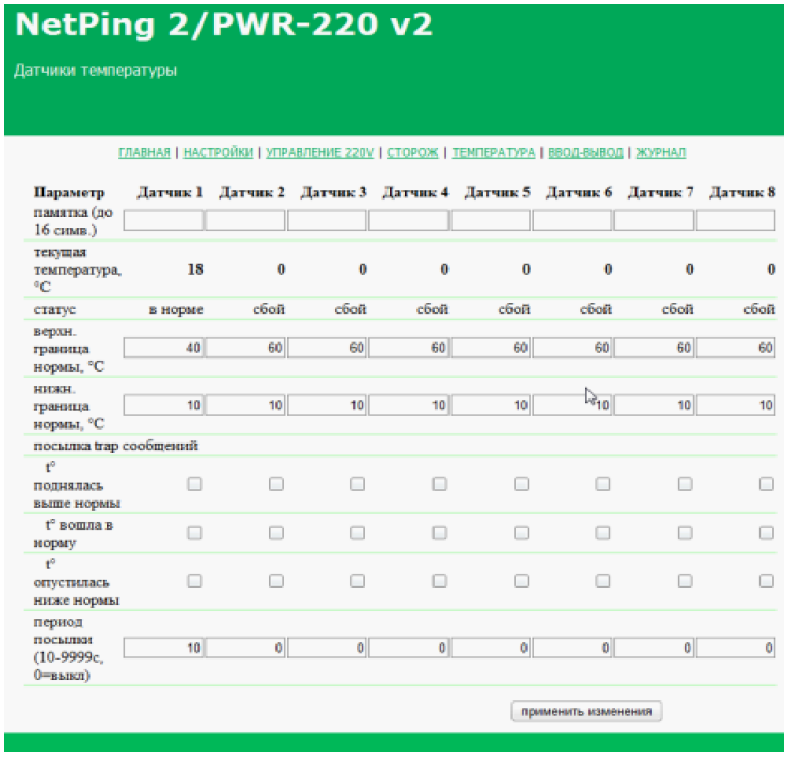

Рисунок 2 - контрольная панель датчиков температуры системы NetPing

Можно использовать сторонние программы мониторинга, например Zabbix, Nagios и **PRTG Network,** который рекомендует производитель NetPing. Преимущество PRTG Network заключается в более удобном интерфейсе программы, возможность вести подробную статистику и мобильную версию приложения (Android и iOS).

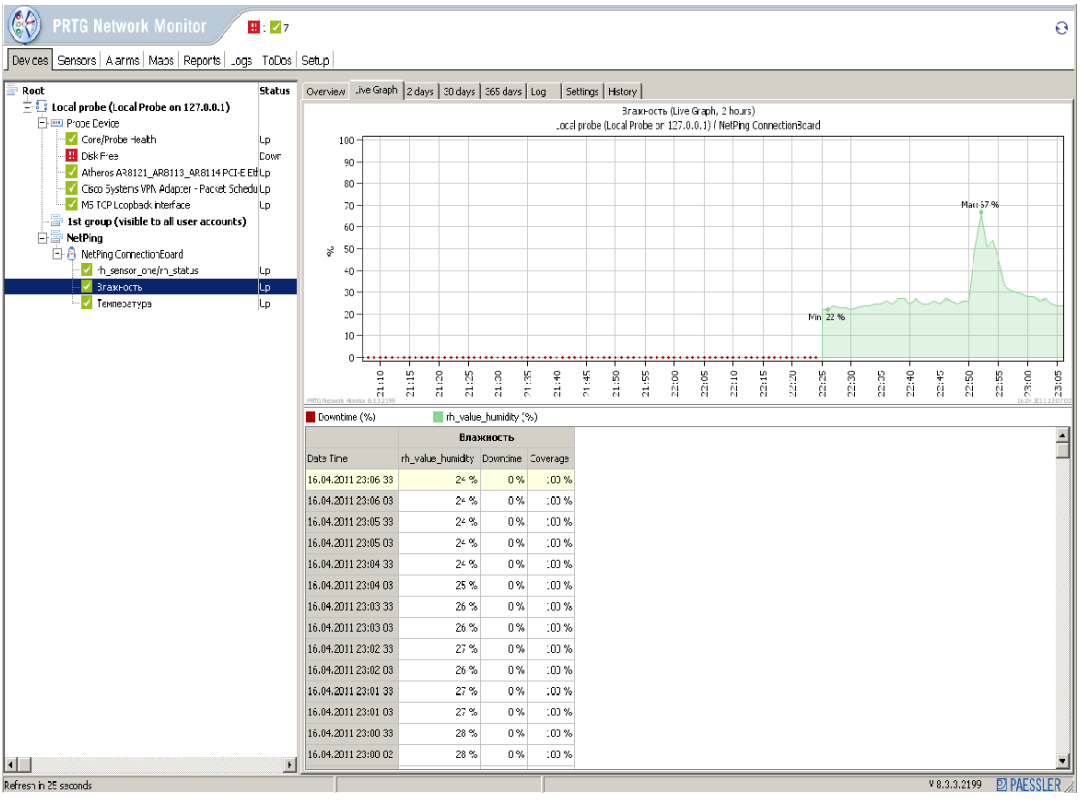

Рисунок 3 - мониториноговое приложение PRTG Network, подключенное к датчикам системы NetPing

OpenRemote

OpenRemote – программный комплекс, обеспечивающий автоматизацию жилых и коммерческих помещений. OpenRemote позволяет создать мобильное приложение для умного дома без программирования, при этом возможно использовать разные существующие протоколы связи - EIB/KNX, AMX, Z-wave. Простыми словами это кроссплатформенный конструктор, в котором можно создать интерфейс будущего мобильного приложения. Контроллеры, которые могут быть использованы: AMX, KNX, Beckhoff, Lutron, Z-Wave, 1-

Wire, MiCasaVerde Vera, EnOcean, xPL, Insteon, X10, Infrared, Russound, GlobalCache, IRTrans, XBMC, VLC, Samsung SmartTV, panStamps, Denon AVR, Marantz AVR, FreeBox, MythTV, RaZBerry и другие.

Рисунок 4 - процесс создания мобильного приложения системы OpenRemote

Home Sapiens

Интеллектуальная система с голосовым управлением, представляет собой программнный комплекс. В комплект не входит оборудование, но при этом обеспечена максимальная совместимость с компьютерным «железом». Обеспечена интеграция с системами Z-wave, Gira, ZigBee, x10, C-bus, что позволяет управлять освещением, бытовой электроникой, системой отопления и прочими подсистемами. Основной упор идет на голосовое управление и удобный интерфейс.

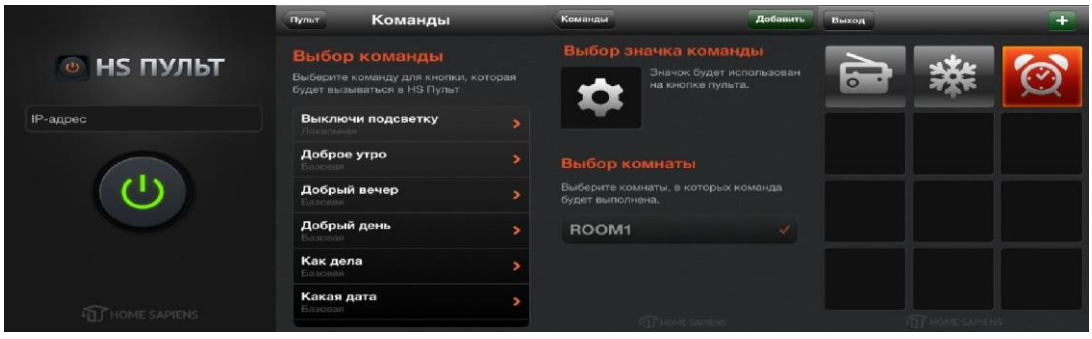

Рисунок 5 - Интерфейс управления Home Sapiens

MajorDoMo

MajorDoMo – это открытая программная платформа для автоматизации домашних процессов. Использует Ethernet. Данная система кроссплатформенная и не требовательна к ресурсам компьютера. Может быть использована без модулей (датчиков) в качестве персонального органайзера. Задачи, решаемые при помощи MajorDoMo:

- система безопасности;
- система микроклимата;
- медиасистема;
- органайзер.

Рисунок 6 - окно контроля системы MajorDoMo

Fibaro

Fibaro - система автоматизации зданий, основанная на беспроводной технологии передачи данных Z-Wave. Миниатюрные модули могут быть установлены за любым выключателем света или в бытовом приборе, то есть отпадает необходимость в монтаже кабельных каналов связи. Благодаря беспроводной технологии передачи данных устройства Fibaro можно демонтировать и переносить на новое место. Высокая интеграция с другими системами. Мозгом системы Fibaro является Home Center 2. Интерфейс предоставляет простой контроль над группами устройств отвечающие за функции – отопления, кондиционирования, освещения и т.д.

Рисунок 7 - интерфейс HomeCenter2 системы управления умным домом от Fibaro

Выводы

Проанализировав вышеперечисленные решения, мы можем выделить несколько основных проблем, присущих существующим системам[3]:

- в большинстве случаев - закрытость протоколов обмена данными, приводящая к невозможности использовать датчики и устройства одной системы в другой;
- почти все решения требуют прокладки кабельных сетей для интеграции в существующую структуру помещений и присоединения к проложенным на этапе строительства коммуникациям;
- рыночная стоимость всех без исключения решений начинается от 50000 рублей даже в базовой комплектации и без учета монтажных и пуско-наладочных работ.

1.2. Требования к разрабатываемому продукту[4]

Учитывая вышеизложенные выводы из анализа существующих на рынке систем, мы можем сформировать список требований к создаваемой системе «умного дома»:

- беспроводная связь между всеми устройствами и датчиками, требующая минимального вмешательства в конструкции помещения;
- простота и масштабируемость протокола связи для возможности использования устройств как своей, так и сторонней разработки;
- защищенность протокола связи от помех и попыток стороннего воздействия;
- максимальная дешевизна и надежность используемых компонентов.

1.3. Требования к защищенному каналу обмена данными

Поскольку программный комплекс «умного дома» напрямую управляет системами безопасности (охранная сигнализация, приводы замков и водопроводных кранов и другие устройства), одной из основных проблем при использовании беспроводных технологий в организации связи между устройствами и датчиками является защищенность канала связи. Любые данные, передающиеся «по воздуху», могут быть перехвачены и использованы злоумышленником в своих целях с помощью относительно недорогой аппаратуры. Соответственно, поскольку физически защититься от такого перехвата не представляется возможным, необходимо использовать методы защиты, делающие перехваченные данные бесполезными. Таким методом защиты является шифрование данных[6].

Также не стоит забывать про современную насыщенность радиоэфира всевозможными данными на разных частотах, которые, по сути, являются для нас помехами и их необходимо учитывать и фильтровать, поддерживать стабильный канал связи в условиях «загрязненного» эфира. Протокол обмена должен решать проблему нестабильной связи и поддерживать канал, а в случае невозможности – программа должна соответствующим образом среагировать на пропадание связи.

1.4. Выбор алгоритма шифрования

С помощью шифрования должны обеспечиваться три состояния безопасности информации: конфиденциальность (скрытие информации от посторонних лиц), целостность (предотвращение изменения данных во время их передачи) и идентифицируемость (аутентификация приемника и передатчика). Только при соблюдении данных условий мы можем считать, что соединение безопасно.

По методам шифрование информации делится на *симметричное* и *несимметричное*.

В *симметричных* криптосистемах для шифрования и расшифровки используется один и тот же ключ. Отсюда название — симметричные. Алгоритм и ключ выбирается заранее и известен обеим сторонам. Сохранение ключа в секретности является важной задачей для установления и поддержки защищённого канала связи. В связи с этим, возникает проблема начальной передачи ключа (синхронизации ключей). Кроме того, существуют методы криптоатак, позволяющие так или иначе дешифровать информацию не имея ключа или же с помощью его перехвата на этапе согласования. В целом эти моменты являются проблемой криптостойкости конкретного алгоритма шифрования и являются аргументом при выборе конкретного алгоритма.

Недостатками симметричного шифрования является проблема передачи ключа собеседнику и невозможность установить подлинность или авторство текста

В системах с *асимметричным* шифрованием используются два ключа — открытый и закрытый, связанные определенным математическим образом друг с другом. Открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для шифрования сообщения и для проверки ЭЦП. Для расшифровки сообщения и для генерации ЭЦП используется секретный ключ.

Данная схема решает проблему симметричных схем, связанную с начальной передачей ключа другой стороне. Если в симметричных схемах злоумышленник перехватит ключ, то он сможет как «слушать», так и вносить правки в передаваемую информацию. В асимметричных системах другой стороне передается открытый ключ, который позволяет шифровать, но не расшифровывать информацию. Таким образом решается проблема симметричных систем, связанная с синхронизацией ключей[6].

Поскольку в создаваемой системе используются микроконтроллеры Arduino Uno, не обладающие большими вычислительными мощностями, присутствует необходимость использования поточного шифрования, а также нет необходимости в постоянной смене ключа, для реализации защиты канала был выбран алгоритм симметричного шифрования **ГОСТ 28147-89** (Магма) в режиме простой замены. С более подробным описанием работы алгоритма мы познакомимся в главе 2.3

Глава 2. Практическая реализация

2.1. Разработка структуры программного комплекса

Проектирование системы строилось на перечне бизнес-процессов, которые должна будет реализовывать данная система. Обобщенно можно выделить следующие:

- посмотр список устройств и их состояния;
- взаимодействие с устройствами, имеющими 2 состояния - включено/выключено или открыто/закрыто;
- получение уведомлений от датчиков;
- управление ir-устройствами;
- добавление новых устройств через панель администрирования.

Реализованная система состоит из:

- веб-интерфейса, (хост-сервер на RaspberryPi);
- утилиты для отправки сообщений по беспроводному каналу (RaspberryPi);
- утилиты для отправки команд через ir-передатчик (RaspberryPi);
- сервиса, слушающего сообщения, полученные по беспроводному каналу (RaspberryPi);
- приложения, снимающего показания с датчика задымления (Arduino Uno);
- приложения, снимающего показания с датчика затопления (Arduino Uno);
- приложения, управляющего сервоприводами для открытия и закрытия штор (Arduino Uno);

Пользователь взаимодействует с системой только через веб-интерфейс.

2.2. Реализация протокола взаимодействия между устройствами системы

Для обмена данными между различными устройствами системы необходимо разработать протокол.

Требования к протоколу:

- Необходимо, чтобы передача данных происходила не отдельными значениями, а набором величин, которые будут описывать, от какого модуля пришли данные, какому модулю предназначаются, что с ними делать и, собственно, сами данные (формировать пакет данных).
- При всем вышеописанном желательно, чтобы размер одного пакета был минимальным. Этим мы уменьшим количество поврежденных пакетов при передаче.
- Также, используя пакеты мы можем указывать их контрольную сумму, что позволит легко вычислить поврежденный пакет и попросить удаленное устройство повторить его передачу.
- Кроме того, можно шифровать данные при передаче для повышения безопасности.

Структура пакета:

Название	Тип поля	Описание
device_id	unsigned int8	Номер устройства, с которого отправляется пакет. Максимальное количество устройств в системе – 256.
destination_id	unsigned int8	Номер устройства-получателя.
packet_id	unsigned int16	Идентификатор пакета. В него записывается случайное число, которое идентифицирует пакет как уникальный.
command	unsigned int8	Номер команды. Инструкция, что делать с данными этого пакета.

Для каждого типа устройств используется своя система команд. Таки образом один и тот же номер команды для разных устройств может иметь разное значение.

Протокол взаимодействия RaspberryPi с выключателями:

1) Включить свет:
    - command = 1
2) Выключить свет:
    - command = 0

Протокол взаимодействия RaspberryPi с шторами/жалюзи:

1) Открыть шторы:
    - command = 1
2) Закрыть шторы:
    - command = 0

Протокол взаимодействия датчиков протечки/задымления с RaspberryPi:

3) Произошла исключительная ситуация (задымление или протечка):
    - command = 1
4) Исключительная ситуация более не наблюдается:
    - command = 0

2.3. Описание алгоритма шифрования

Рассмотрим подробнее алгоритм шифрования.

ГОСТ 28147-89 (“Магма”)[7] — блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками. Основа алгоритма шифра — сеть Фейстеля.

Для зашифровывания в этом режиме 64-битный блок открытого текста сначала разбивается на две половины (младшие биты — A, старшие биты — B). На i-ом цикле используется подключ K_i:

$$A_{i+1} = B_i \oplus f(A_i, K_i)$$

$$B_{i+1} = A_i$$

Для генерации подключей исходный 256-битный ключ разбивается на восемь 32-битных блоков: K1…K8.

Ключи K9…K24 являются циклическим повторением ключей. K1…K8 (нумеруются от младших битов к старшим). Ключи K25…K32 являются ключами K8…K1.

После выполнения всех 32 раундов алгоритма, блоки A33 и B33 склеиваются (старшим блоком становится A33, а младшим — B33) — результат есть результат работы алгоритма.

Расшифровывание выполняется так же, как и зашифровывание, но инвертируется порядок подключей Ki.

Функция $f(A_i, K_i)$ вычисляется следующим образом:

Ai и Ki складываются по модулю 232. Результат разбивается на восемь 4-битовых подпоследовательностей, каждая из которых поступает на вход своего узла таблицы замен (в порядке возрастания старшинства битов), называемого ниже S-блоком. Общее количество S-блоков алгоритма — восемь, то есть столько же, сколько и подпоследовательностей. Каждый S-блок представляет собой перестановку чисел от 0 до 15 (конкретный вид S-блоков в стандарте не определен). Первая 4-битная подпоследовательность попадает на вход первого S-блока, вторая — на вход второго и т. д.

Если узел S-блока выглядит так:

1, 15, 13, 0, 5, 7, 10, 4, 9, 2, 3, 14, 6, 11, 8, 12

и на входе S-блока 0, то на выходе будет 1, если 4, то на выходе будет 5, если на входе 12, то на выходе 6 и т. д.

Выходы всех восьми S-блоков объединяются в 32-битное слово, затем всё слово циклически сдвигается влево (к старшим разрядам) на 11 битов.

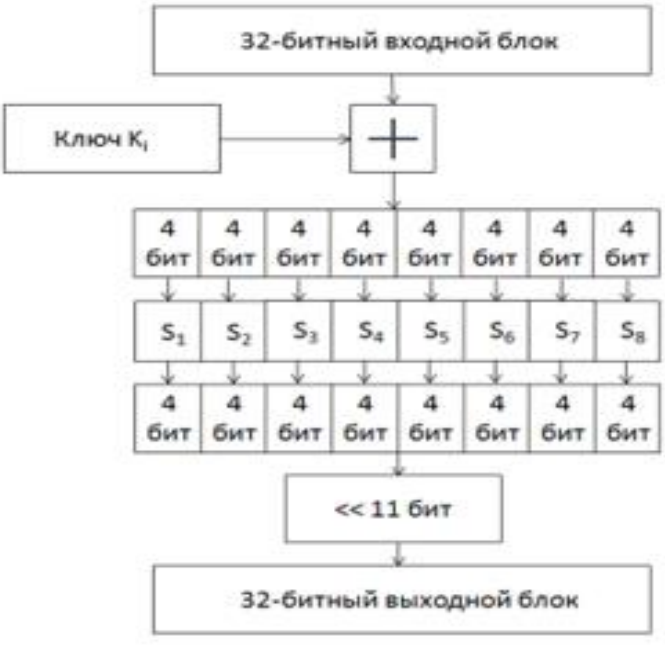

Рисунок 8 - Принцип работы шифрующего алгоритма

2.4 Реализация для RaspberryPi[9][11]

Описание веб-части системы

Веб-часть системы, осуществляющая взаимодействие пользователя и системы, реализована на принципах концепции MVC - модель-вид-контроллер.

MVC это схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные.

Все HTTP запросы обрабатываются классом RequestParser. Этот класс выполняет разбор запросов и передачу управления контроллерам для дальнейшего выполнения.

IndexController - контроллер, обеспечивающий работу главной страницы веб-интерфейса. Он генерирует страницу с описанием всех устройств подключенных к систем и с возможностью управления ими.

BlindsController - контроллер, отвечает за обработку запросов, связанных со шторами (открыть/закрыть и получить список устройств - шторы или жалюзи). Этот контроллер обрабатывает только ajax-запросы и возвращает результаты в виде json-строк.

SensorController - контроллер, отвечает за обработку запросов, связанных с датчиками задымления и протечки. Возвращает json-строку со списком всех датчиков и их состояниями для дальнейшего отображения в веб-интерфейсе.

LightController - контроллер, отвечает за обработку запросов, связанных с управлением световыми приборами. Принимает ajax-запросы на включение/выключение световых приборов. Позволяет вернуть json-строку со списком всех световых приборов и их состояниями - включен или выключен.

IRController - контроллер, отвечает за обработку запросов, связанных с IR-устройствами. Принимает ajax-запросы с кодами кнопок ir-пультов для управления телевизорами, акустическими системами и прочей техникой с возможностью управления через ir-приемник.

Отдельно стоит выделить контроллер AdminController, в нем реализовано администрирование системы. Он позволяет совершать авторизацию пользователей и редактировать устройства, подключенные к системе.

За генерацию страниц из html-шаблонов отвечает класс View. Он получает на вход название шаблона и список параметров необходимых для создания страницы[12].

Классы-модели системы: Blinds, Light, Sensor и User реализуют шаблон проектирования ActiveRecord. Эти классы используются для получения и сохранения данных в таблицах с соответствующими названиями.

Для отправки сообщений беспроводным устройствам реализован класс WirelessRemote. Он представляет из себя обертку над утилитой send, описанной ниже.

Для отправки сообщений ir-устройствам реализован класс IRRemote. Он представляет из себя обертку над утилитой irsend из пакета LIRC.

Описание базы данных

На рисунке 9 продемонстрирована ER-модель базы данных системы автоматизации жилых помещений. База данных используется для хранения аккаунтов пользователей, имеющих доступ к системе и информацию о текущих состояниях устройств подключенных к системе.

С большинством устройств нет обратной связи, таким образом для того, чтобы всегда знать, в каком состоянии они сейчас находятся (включено/выключено), необходимо запоминать это состояние сразу после его изменения. Также таблица sensor используется для обратной связи между датчиками затопления/задымления и веб-интерфейсом. Так фоновый сервис, слушающий сообщения от датчиков, обновляет их состояния в таблице sensor, а php-скрипт периодически читает эти значения и отображает в веб-интерфейсе.

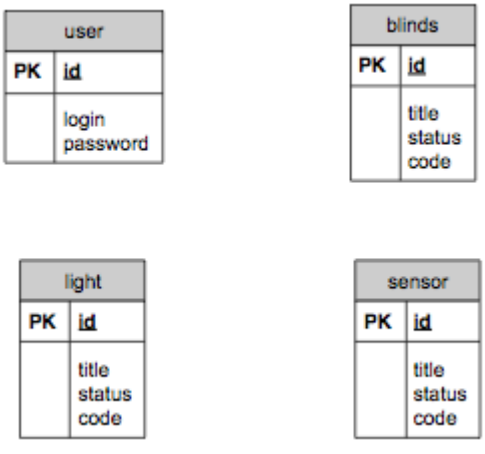

Рисунок 9 - Модель сущность-связь базы данных системы автоматизации жилых помещений

Утилита Send

Для отправки сообщений на выключатели и реле (шторы/жалюзи, приводы водопроводных кранов и замков, прочие сервоприводы) используется консольная утилита send, написанная на языке C++.

У RaspberryPi есть аппаратный интерфейс GPIO, к которому осуществляется подключение беспроводного контроллера 433MHz.

Акроним GPIO расшифровывается как General Purpose Input Output. Это низкоуровневый интерфейс ввода-вывода прямого управления. Через этот интерфейс Raspberry может слушать и отдавать команды любому внешнему устройству.

Для того, чтобы можно было из программы работать с пинами GPIO, используется сторонняя библиотека WiringPi.

Эта библиотека выполняет привязку пинов GPIO к переменным типа int языка C++. Это позволяет получать сообщения из сети простой проверкой значений переменных. Отправка сообщений в сеть выполняется так же с помощью подачи сигналов на пины GPIO через изменение значений соответствующих переменных через определенные промежутки времени (модуляция). Модуляция данных осуществляется через временные задержки в изменении уровня напряжения на пинах GPIO

Выдержка из кода:

1) подать напряжение:

digitalWrite(this->nTransmitterPin, HIGH);

2) сделать задержку на nHighPulses микросекунд:

delayMicroseconds(nHighPulses);

3) снять напряжение:

digitalWrite(this->nTransmitterPin, LOW);

2) сделать задержку на nLowPulses микросекунд:

delayMicroseconds(nLowPulses);

Утилита send принимает следующие входные параметры:

--code - код устройства, для которого предназначено сообщение;

--command - сообщение-команда, которое будет передано на устройство.

Общая логика работы утилиты send заключается в следующем:

1. Получаем на вход код устройства и команду предназначенную для этого устройства.
2. Передаем эти параметры методу send() библиотеки RCSwitch, которая, в свою очередь, кодирует это в сообщения для передачи по широковещательному беспроводному протоколу.

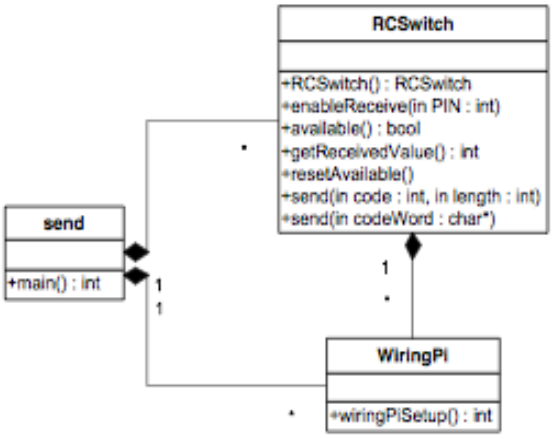

Рисунок 80 - диаграмма классов утилиты send

IRSend

Для управления устройствами с IR-приемником используется утилита irsend, реализованная в GNU проекте LIRC. Irsend работает с инфракрасным передатчиком, используя который отправляет команды для управления электроприборами, такими как: телевизоры, акустические системы и т.п.

Слушающий фоновый сервис

Для получения сообщений от датчиков задымления и протечек используется фоновый сервис, слушающий изменения на GPIO интерфейсе RaspberryPi.

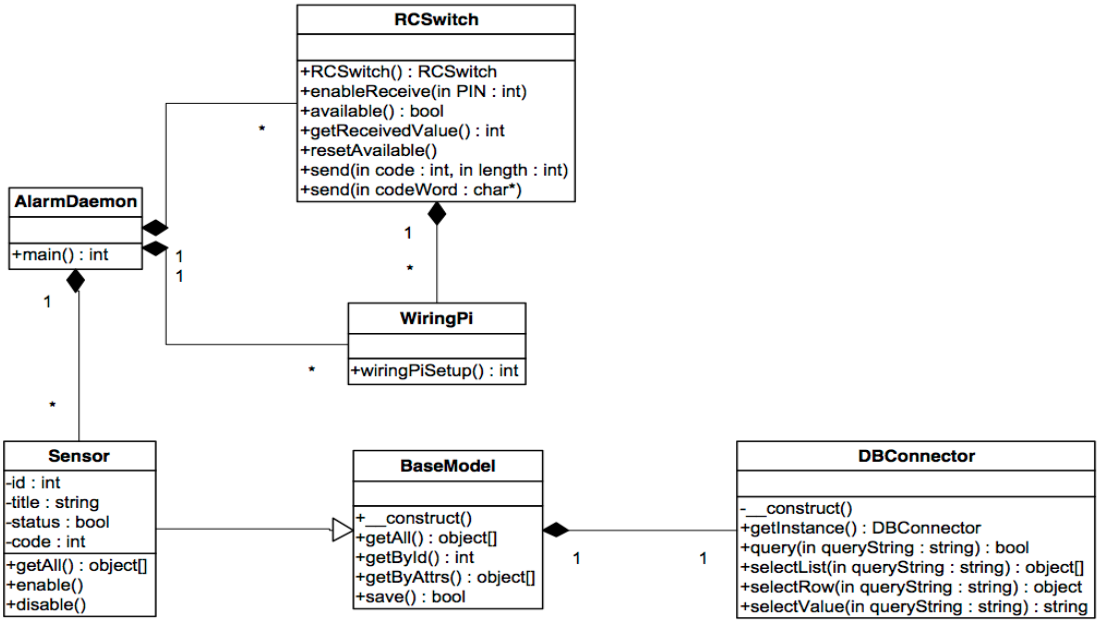

Рисунок 9 - диаграмма классов фонового сервиса

На рисунке 11 продемонстрирована диаграмма классов фоновой службы, работающей на аппаратной платформе RaspberryPi. Класс RCSwitch отвечает за прием и первичную обработку сообщений, полученных приемником беспроводной связи. Класс WiringPi организует работу со сторонней библиотекой WiringPi и выполняет первичную настройку оборудования [10]. Он выполняет привязку пинов GPIO к переменным типа int. Это позволяет получать сообщения из сети простой проверкой значений переменных. Классы Sensor, BaseModel и обеспечивают сохранение полученных данных в базу данных, реализуя шаблон проектирования active record. DBConnector - класс-одиночка, выполняющий непосредственное подключение к базе данных.

Алгоритм работы сервиса можно описать следующей цепочкой действий:

1. Датчик протечки или задымления присылает команду о том, что произошла исключительная ситуация.
2. В функции main файла AlarmDaemon.cpp выполняется бесконечный цикл, в котором проверяется наличие новых поступивших сообщений с GPIO.
3. Если пришло сообщение об аварийной ситуации с какого-либо из датчиков, это состояние датчика сохраняется в базу данных в таблице sensor.
4. Если позже поступило сообщение от датчика о прекращении аварийной ситуации, то статус этого датчика также обновится в таблице sensor.
5. Веб-интерфейс работает с этой же базой данных и выполняет периодическую проверку статусов датчиков. Если какой-то из датчиков имеет статус “аварийная ситуация”, это отобразится в веб-интерфейсе для информирования пользователя.

2.5 Реализация для микроконтроллера[8]

Передатчик для датчиков задымления/протечки

В качестве платформы для датчиков используется Arduino Uno, с подключенным по serial port сенсором. В бесконечном цикле выполняется опрос сенсора, если произошла аварийная ситуация то формируется сообщение для

утилиты RCSwitch, которая выполняет отправку этого сообщения по беспроводному протоколу для RaspberryPi.

Приемник для сервопривода

Так же в бесконечном цикле получаем сообщения от RCSwitch, полученные команды передаем на сервоприводы.

Приемник для выключателя

Так же в бесконечном цикле получаем сообщения от RCSwitch, полученные команды передаем выключателю.

Описание реализации программы оповещения о задымлении и протечке для микроконтроллера

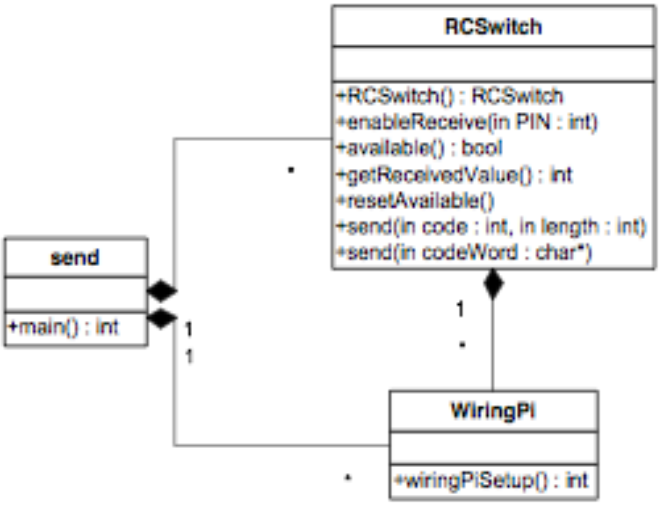

Рисунок 10 - диаграмма классов программы оповещения о задымлении и протечке для Arduino Uno

На рисунке 12 приведена диаграмма основных классов приложения, работающего на микроконтроллере Arduino Uno. Класс RCSwitch отвечает непосредственно за прием и передачу данных. Класс WiringPi организует работу со сторонней библиотекой WiringPi и выполняет первичную настройку оборудования.

send - точка входа приложения, единственный метод main выполняет инициализацию приложения и запуск бесконечного цикла опроса датчиков протечек и задымления.

RCSwitch - реализует прием и дешифрацию сообщений.

- enableReceive - метод, переключающий библиотеку RCSwitch на прием данных;
- available - метод, проверяющий наличие нового необработанного сообщения;
- getReceivedValue - метод, возвращающий полученное сообщение;
- resetAvailable - метод, переключающий приложение в состояние ожидания получения нового сообщения;
- send - метод, обеспечивающий отправку сообщения с RaspberryPi. Сообщение может быть закодировано как одной переменной типа long, так и массивом символов char[]

WiringPi - выполняет привязку пинов GPIO к переменным типа int. Это позволяет получать сообщения из сети простой проверкой значений переменных.

- wiringPiSetup - метод, выполняющий настройку интерфейса GPIO.

2.6. Общее функционирование системы

Описание реализации программы, обрабатывающей сообщения, полученные от датчиков задымления и протечек

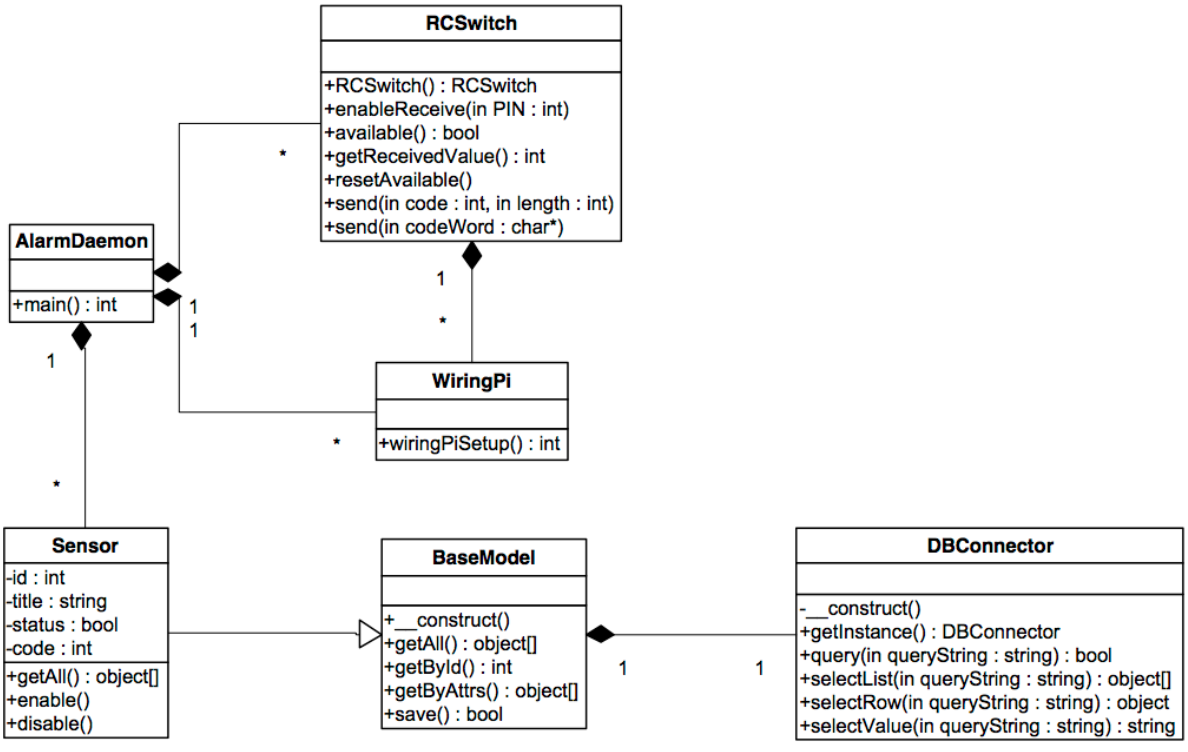

Рисунок 11 - диаграмма классов программы, обрабатывающей сообщения полученные от датчиков задымления и протечек

На рисунке 13 продемонстрирована диаграмма классов фоновой службы, работающей на аппаратной платформе RaspberryPi. Класс RCSwitch отвечает за прием и первичную обработку сообщений, полученных приемником беспроводной связи. Класс WiringPi организует работу со сторонней библиотекой WiringPi и выполняет первичную настройку оборудования. Классы Sensor, BaseModel и DBConnector обеспечивают сохранение полученных данных в базу данных.

AlarmDaemon - точка входа приложения, единственный метод main выполняет инициализацию приложения и запуск бесконечного цикла прослушивания сообщений от датчиков протечек и задымления.

WiringPi - выполняет привязку пинов GPIO к переменным типа int. Это позволяет получать сообщения из сети простой проверкой значений переменных.

- wiringPiSetup - метод, выполняющий настройку интерфейса GPIO

RCSwitch - реализует прием и дешифрацию сообщений.

- enableReceive - метод, переключающий библиотеку RCSwitch на прием данных;
- available - метод, проверяющий наличие нового необработанного сообщения;
- getReceivedValue - метод, возвращающий полученное сообщение;
- resetAvailable - метод, переключающий приложение в состояние ожидания получения нового сообщения;
- send - метод, обеспечивающий отправку сообщения с RaspberryPi.

BaseModel - базовый класс для классов-моделей, реализующих работу с базой данных через шаблон проектирования - ActiveRecord;

- __construct - метод, инициализирующий подключение к базе данных через класс DBConnector;
- getAll - метод, возвращающий список всех записей;
- getById - метод, возвращающий запись для указанного ID;
- getByAttrs - метод, возвращающий список записей, удовлетворяющих параметрам поиска;
- save - метод, реализующий сохранение объекта в базу данных. Если у объекта задан первичный ключ, то запись обновится, если первичный ключ не задан, то будет создана новая запись.

Sensor - дочерний класс BaseModel, обеспечивает работу с таблицей sensor.

- id - атрибут, содержит идентификатор записи из таблицы sensor;
- title - атрибут, содержит название датчика;
- status - атрибут, содержит статус датчика (в состоянии тревоги или спокойствия);
- code - атрибут, содержит код датчика, используется для идентификации устройства в протоколе;
- getAll - метод, возвращает список всех датчиков из таблицы sensor;

- enable - метод, помечает датчик как включенный (в состоянии тревоги);
- disable - метод, помечает датчик как выключенный (в состоянии спокойствия).

DBConnector - класс, обеспечивающий подключение к базе данных, реализует шаблон проектирования - одиночка.

- __construct - метод, инициализирует подключение к базе данных;
- getInstance - метод, возвращает экземпляр данного класса;
- query - метод, выполняет SQL запрос на вставку или обновление;
- selectList - метод, возвращает список записей, удовлетворяющих параметрам поиска;
- selectRow - метод, возвращает запись, удовлетворяющую параметрам поиска;
- selectValue - метод, возвращает единственное значение, удовлетворяющую параметрам поиска и указанному столбцу.

Диаграмма классов системы автоматизации жилых помещений

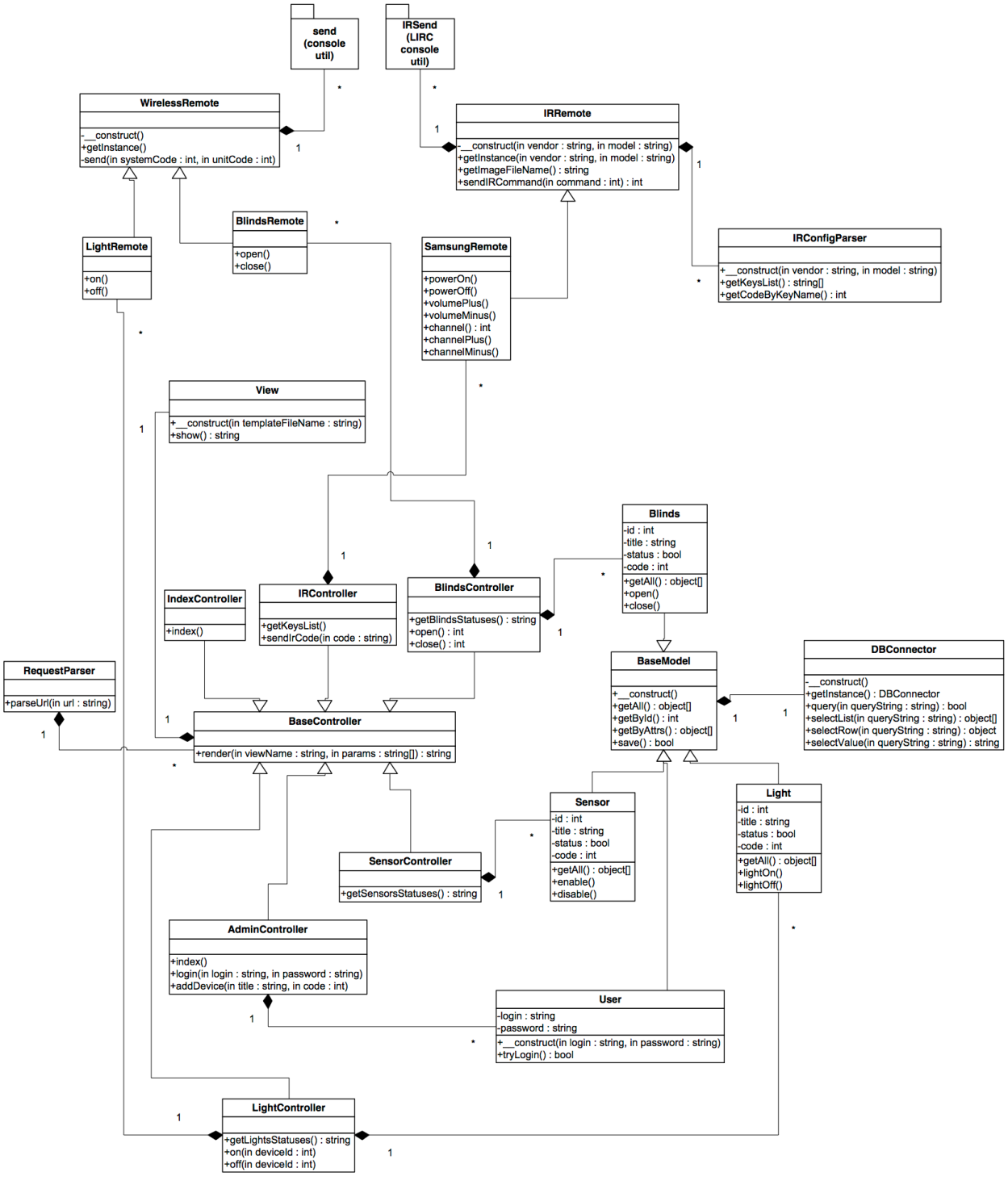

Рисунок 12 - диаграмма классов системы автоматизации жилых помещений

На рисунке 14 продемонстрирована диаграмма классов системы автоматизации жилых помещений.

Систему можно разделить на 3 основные составляющие части: веб-интерфейс, СУБД и интерфейсы взаимодействия с беспроводными устройствами.

Работу веб-интерфейса обеспечивают следующие классы:

RequestParser - класс, выполняющий обработку и маршрутизацию http-запросов.

- parseUrl - метод, выполняет разбор http-запроса и вызывает соответствующий метод указанного в запросе контроллера.

BaseController - базовый класс для контроллеров.

- render - метод, выполняет рендеринг страницы с передачей всех необходимых переменных в html-шаблон;

IndexController - класс, реализующий работу главной страницы веб-интерфейса.

- index - метод, выполняющий генерацию главной страницы.

IRController - класс, отвечает за обработку http запросов, связанных с IR-устройствами.

- getKeysList - метод, возвращает список всех допустимых кнопок пульта дистанционного управления;
- sendIrCode - метод, получает на вход код нажатой кнопки пульта ДУ, с дальнейшей передачей его ir-приемнику.

BlindsController - класс, отвечает за обработку http запросов, связанных со шторами.

- getBlindsStatuses - метод, возвращает список штор и жалюзи, управляемых системой, с их текущим статусом - открыт/закрыт;
- open - метод, обрабатывает запрос на открытие штор;
- close - метод, обрабатывает запрос на закрытие штор;

SensorController - класс, отвечает за обработку http-запросов, связанных с датчиками задымления и протечки.

- getSensorsStatuses - метод, возвращает список датчиков с их текущим статусом - активен/неактивен.

LightController - класс, отвечает за обработку http-запросов, связанных с управлением световыми приборами.

- getLightsStatuses - метод, возвращает список световых приборов с их текущим статусом - включен/выключен;
- on - метод, обрабатывает запрос на включение прибора освещения;
- off - метод, обрабатывает запрос на выключение прибора освещения.

AdminController - класс, обеспечивает настройку системы.

- index - метод, выполняющий генерацию главной страницы администрирования;
- login - метод, выполняющий генерацию страницы с формой для логина;
- addDevice - метод, реализующий возможность добавления световых приборов.

View - класс, отвечающий за генерацию web-страниц из html-шаблонов.

- __construct - конструктор, получет на вход название шаблона;
- show - метод, выполняющий генерацию страницы из шаблона и вывод на экран.

За работу с базой данных отвечают следующие классы:

DBConnector - класс, выполняющий подключение и низкоуровневые операции с базой данных. Реализует шаблон проектирования - одиночка.

- __construct - конструктор, выполняет подключение к СУБД;
- getInstance - статический метод, возвращающий экземпляр данного класса;
- query - метод, выполняет SQL запрос к базе;
- selectList - метод, выполняет SQL запрос на SELECT, возвращает список строк;
- selectRow - метод, выполняет SQL запрос на SELECT, возвращает одну строку;
- selectValue - метод, выполняет SQL запрос на SELECT, возвращает одно значение;

BaseModel - базовый класс для ActiveRecord моделей.

- __construct - конструктор;
- getAll - метод, возвращает список ActiveRecord объектов, соответсвующих всем записям из таблицы;
- getById - метод, возвращает ActiveRecord объект для указанного первичного ключа;
- getByAttrs - метод, возвращает список ActiveRecord объектов, удовлетворяющих параметрам выборки;
- save - метод, выполняет сохранение ActiveRecord объекта в базе. Если первичный ключ был указан, то запись в таблице обновляется, в противном случае - создается новая запись.

Blinds - класс, обеспечивающий работу с таблицей blinds.

- id - атрибут, уникальный идентификатор устройства, суррогатный первичный ключ;
- title - атрибут, название устройства;
- status - атрибут, состояние устройства - открыто или закрыто;
- code - атрибут, код устройства, используется в протоколе для идентификации устройства;
- getAll - метод, возвращает все строки из таблицы blinds;
- open - метод, выставляет поле status = 1 для строки с указанным id;
- close - метод, выставляет поле status = 0 для строки с указанным id.

Light - класс, обеспечивающий работу с таблицей light.

- id - атрибут, уникальный идентификатор устройства, суррогатный первичный ключ;
- title - атрибут, название устройства;
- status - атрибут, состояние устройства - включено или выключено;
- code - атрибут, код устройства, используется в протоколе для идентификации устройства;
- getAll - метод, возвращает все строки из таблицы light;
- lightOn - метод, выставляет поле status = 1 для строки с указанным id;
- lightOff - метод, выставляет поле status = 0 для строки с указанным id.

Sensor - класс, обеспечивающий работу с таблицей sensor.

- id - атрибут, уникальный идентификатор устройства, суррогатный первичный ключ;
- title - атрибут, название устройства;
- status - атрибут, состояние устройства - активировано или в нет;
- code - атрибут, код устройства, используется в протоколе для идентификации устройства;
- getAll - метод, возвращает все строки из таблицы sensor;
- enable - метод, выставляет поле status = 1 для строки с указанным id;
- disable - метод, выставляет поле status = 0 для строки с указанным id;

User - класс, обеспечивающий работу с таблицей user.

- login - атрибут, логин (имя) пользователя;
- password - атрибут, пароль пользователя;
- __construct - конструктор, создает объект с указанным логином и паролем;
- tryLogin - метод, выполняет проверку корректности указанного логина и пароля путем поиска соответствующей строки в базе;

Работа с беспроводными устройствами обеспечивается классами:

WirelessRemote - базовый класс, реализующий работу беспроводными устройствами через консольную утилиту send. Реализует шаблон проектирования - одиночка.

- __construct - конструктор;
- getInstance - статический метод, возвращающий экземпляр данного класса;
- send - метод, вызывает утилиту send с параметрами, которые будут переданы беспроводным устройствам;

LightRemote - класс, реализующий управление световыми приборами.

- on - вызвать утилиту send с параметрами, необходимыми для включения устройства;
- off - вызвать утилиту send с параметрами, необходимыми для выключения устройства;

BlindsRemote - класс, реализующий управление шторами или жалюзи.

- open - вызвать утилиту send с параметрами, необходимыми для открытия штор;
- close - вызвать утилиту send с параметрами, необходимыми для закрытия штор;

send - консольная утилита, выполняет ретрансляцию запросов из класса WirelessRemote беспроводным устройствам через прямую работу с беспроводным передатчиком 433MHz.

IRRemote - базовый класс, обеспечивает работу с устройствами, принимающими команды в инфракрасном диапазоне. Реализует шаблон проектирования - одиночка.

- __construct - конструктор;
- getInstance - статический метод, возвращающий экземпляр данного класса;
- getImageFileName - метод, возвращает имя файла изображения пульта дистанционного управления, для использования в веб-интерфейсе;
- sendIRCommand - метод, отправляет IR-команду, используя консольную утилиту IRSend из пакета LIRC.

IRConfigParser - класс, выполняющий разбор конфигурационных файлов для различных устройств из состава проекта LIRC.

- __construct - конструктор;
- getKeysList - метод, возвращает список кнопок пульта ДУ для заданного устройства;
- getCodeByKeyName - метод, возвращает код, идентифицирующий кнопку пульта ДУ по ее названию.

SamsungRemote - пример реализации класса, работающего с устройствами конкретного производителя.

- powerOn - метод, выполняет включение телевизора;
- powerOff - метод, выполняет выключение телевизора;
- volumePlus - метод, выполняет увеличение громкости;

- volumeMinus - метод, выполняет уменьшение громкости;
- channel - метод, переключает канал на телевизоре на заданный;
- channelPlus - метод, переключает канал на следующий;
- channelMinus - метод, переключает канал на предыдущий;

IRSend - консольная утилита, выполняет ретрансляцию запросов из класса IRRemote через IR-передатчик.

Модель сущность-связь базы данных системы автоматизации жилых помещений

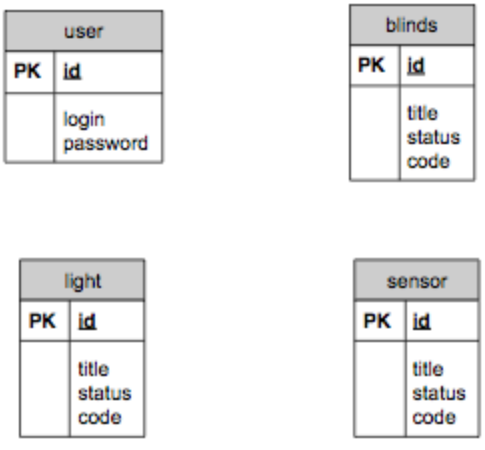

Рисунок 13 - модель сущность-связь базы данных системы автоматизации жилых помещений

На рисунке 15 продемонстрирована ER-модель базы данных системы автоматизации жилых помещений. База данных содержит следующие таблицы:

user - хранит информацию о профилях пользователей.

- id - уникальный идентификатор пользователя, суррогатный первичный ключ;
- login - логин (имя) пользователя;
- password - пароль пользователя.

blinds - содержит информацию о шторах и жалюзи, открытие и закрытие которых автоматизировано с использованием сервоприводов.

- id - уникальный идентификатор устройства, суррогатный первичный ключ;
- title - название устройства;
- status - состояние устройства - открыто или закрыто;
- code - код устройства, используется в протоколе для идентификации устройства.

light - содержит информацию о световых приборах, доступных для автоматического управления.

- id - уникальный идентификатор устройства, суррогатный первичный ключ;
- title - название устройства;
- status - состояние устройства - включено или выключено;
- code - код устройства, используется в протоколе для идентификации устройства.

sensor - содержит информацию о датчиках задымления и протечек и их состоянии.

- id - уникальный идентификатор устройства, суррогатный первичный ключ;
- title - название устройства;
- status - состояние устройства - активировано или нет;
- code - код устройства, используется в протоколе для идентификации устройства.

Схема взаимодействия пользователя с системой для управления устройствами через IR-порт

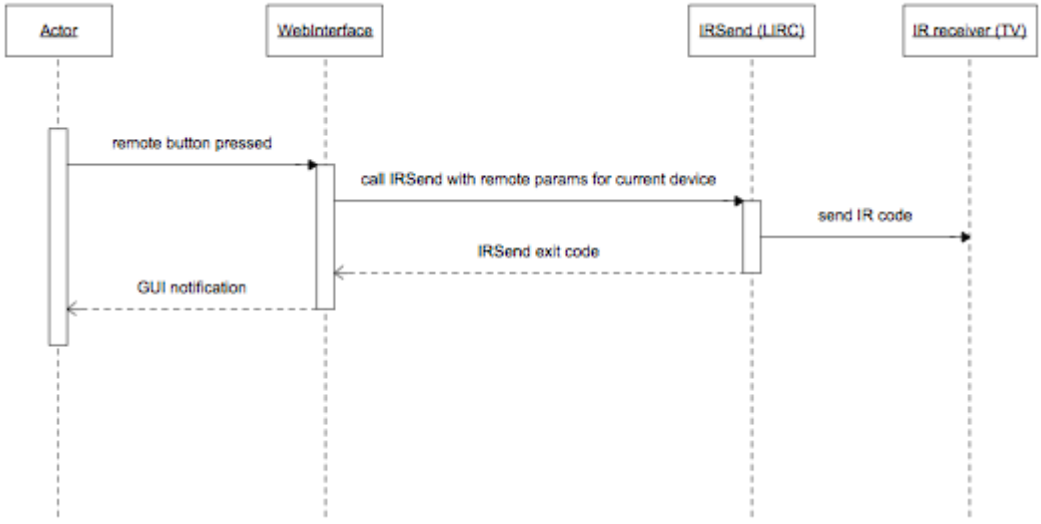

Рисунок 14 - диаграмма последовательностей процесса взаимодействия пользователя с системой для управления устройствами через IR-порт

На рисунке 16 представлена диаграмма последовательностей, демонстрирующая процесс работы с IR-устройствами.

Объекты системы:

- Actor - пользователь системы.
- WebInterface - интерфейс взаимодействия пользователя с системой, реализованный в виде совокупности web-страниц.
- IRSender(LIRC) - модуль системы, обеспечивающий отправку команд через инфракрасный интерфейс, используется из состава проекта с открытым исходным кодом - LIRC.
- IR receiver - приемник команд, отправленных в инфракрасном диапазоне.

Схема взаимодействия пользователя с системой для управления устройствами по беспроводному протоколу на частоте 433MHz

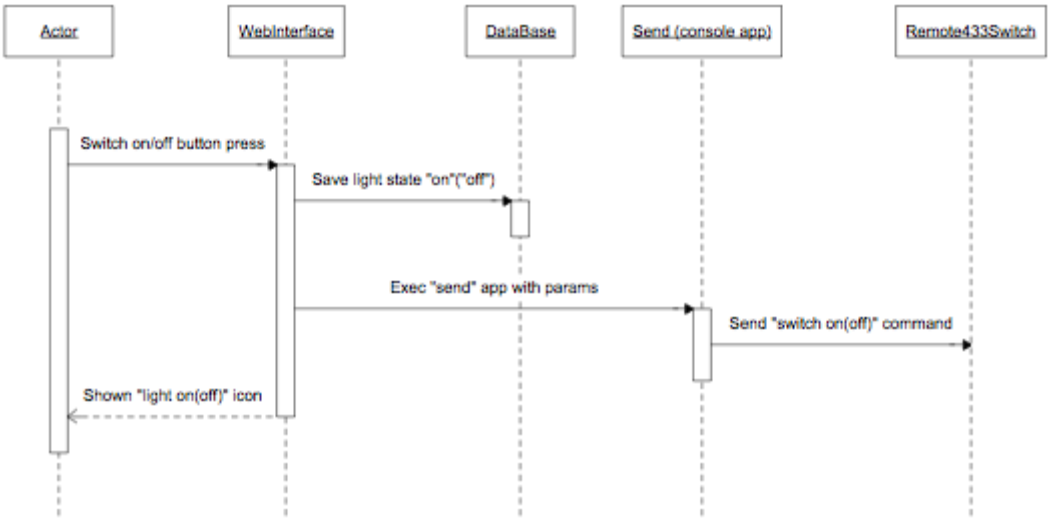

Рисунок 15 - диаграмма последовательностей процесса взаимодействия пользователя с системой для управления устройствами по беспроводному протоколу на частоте 433MHz

На рисунке 17 представлена диаграмма последовательностей, демонстрирующая процесс работы с беспроводными устройствами.

- Actor - пользователь системы.
- WebInterface - интерфейс взаимодействия пользователя с системой, реализованный в виде совокупности web-страниц.
- DataBase - СУБД, используемая для хранения информации о состояниях беспроводных устройств.
- Send - модуль системы, обеспечивающий отправку команд на частоте 433MHz.
- Remote433Switch - приемник команд.

Схема взаимодействия датчиков протечки и задымления с системой для информирования пользователя о нештатной ситуации

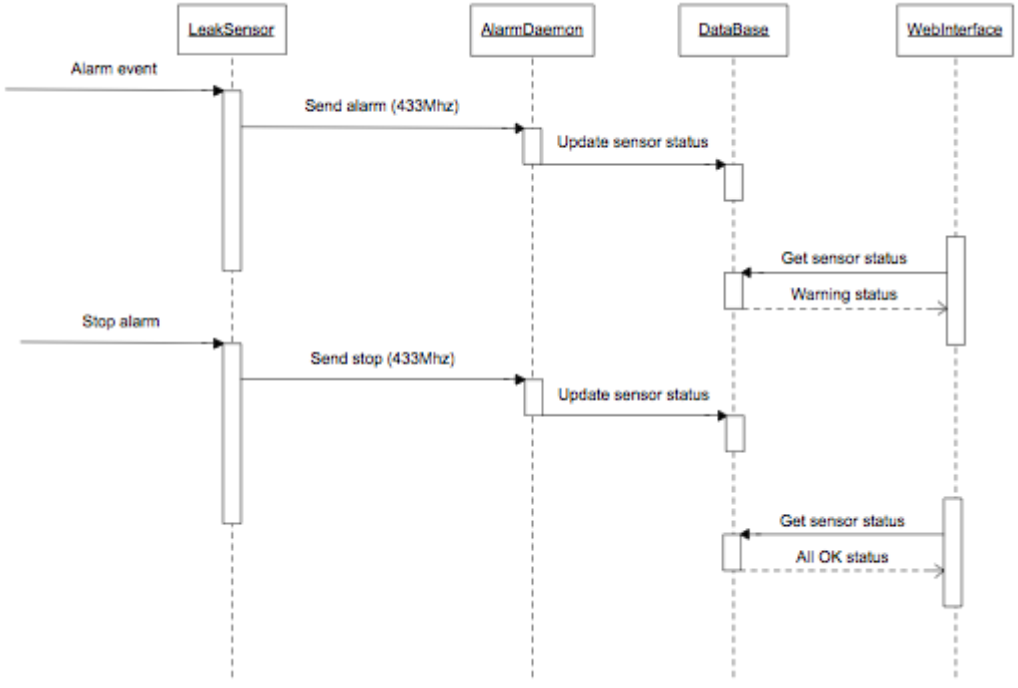

Рисунок 16 - диаграмма последовательностей взаимодействия датчиков протечки и задымления с системой

На рисунке 18 представлена диаграмма последовательностей, демонстрирующая процесс работы с датчиками задымления и протечки.

- LeakSensor - датчик протечки.
- AlarmDaemon - фоновый сервис, слушает сообщения о тревоге, полученные от датчиков протечек и задымления.
- DataBase - СУБД, используемая для хранения информации о состояниях беспроводных устройств.
- WebInterface - интерфейс взаимодействия пользователя с системой, реализованный в виде совокупности web-страниц.

Диаграмма вариантов использования системы автоматизации жилых помещений

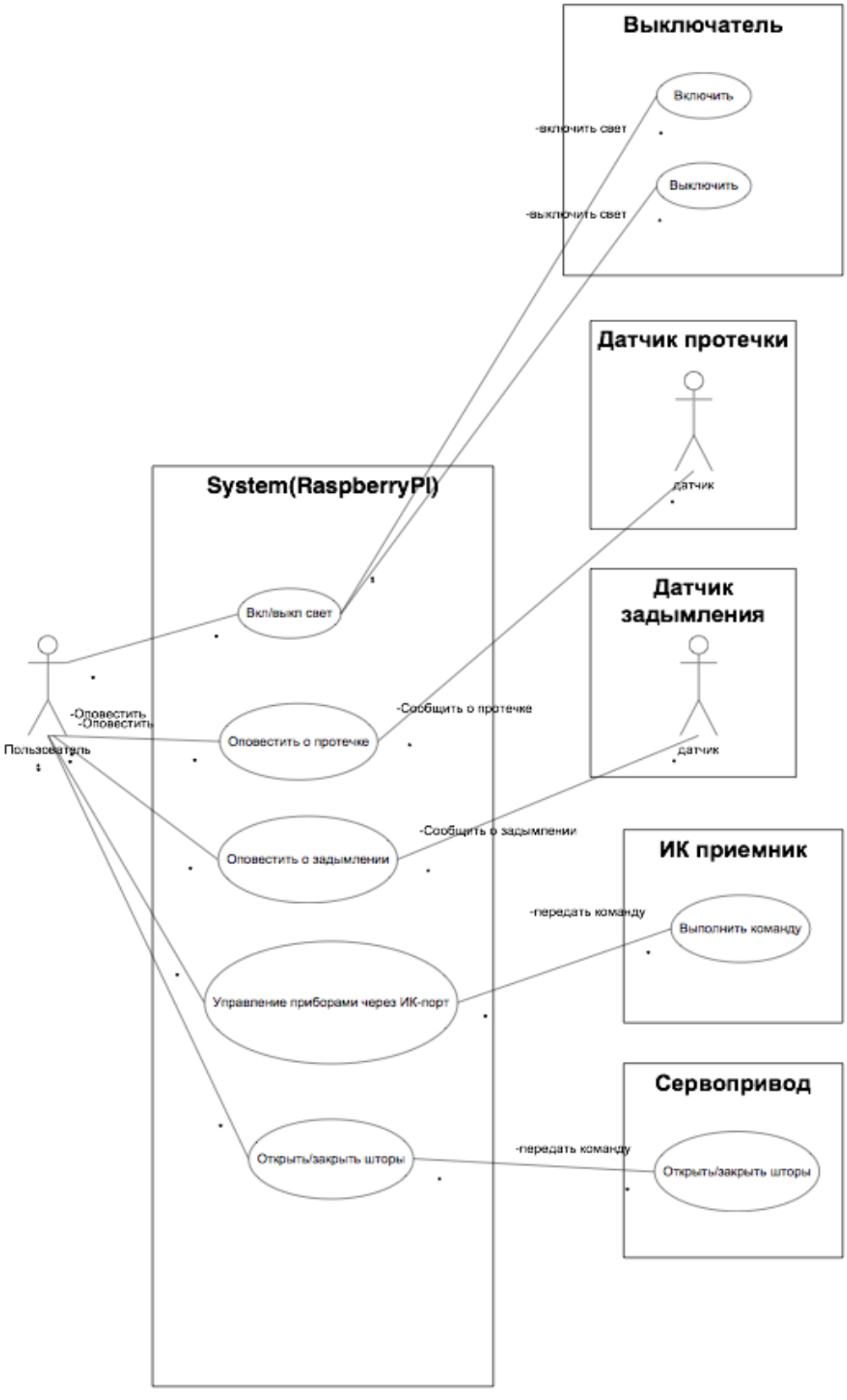

Рисунок 19 - диаграмма вариантов использования системы автоматизации жилых помещений

На рисунке 19 представлена диаграмма вариантов использования системы автоматизации жилых помещений.

Пользователь системы может выполнять следующие действия:

- включать/выключать свет;
- открывать/закрывать шторы или жалюзи;
- управлять приборами с IR-интерфейсом.

Система взаимодействует с пользователем через уведомления о событиях:

- задымление;
- протечка.

Система взаимодействует с устройствами (управляет ими):

- выключатели света;
- сервоприводы;
- IR-устройства.

Датчики, регистрирующие возникновение задымления или протечки, взаимодействуют с системой через посылку сообщений о:

- срабатывании датчика регистрации задымления;
- срабатывании датчика регистрации протечки.

Заключение

В рамках данной работы была решена поставленная задача, а именно – разработка системы управления «умным домом», обладающая базовым функционалом и возможностями к расширению.

Достоинствами разработанной системы является то, что она, во-первых, является универсальной для любых устройств, работающих в заданном частотном диапазоне, во-вторых, защищенной от помех и внешнего вмешательства, в третьих, достаточно дешева для успешного внедрения даже в сложной экономической обстановке.

В дальнейшем планируется расширение имеющегося функционала в сторону увеличения количества типов поддерживаемых устройств (в частности, работа с другими частотными диапазонами и протоколами связи).

Список использованной литературы

1. Велт, Т. Дж., Элсенпитер, Р. К. "Умный дом" строим сами / Т. Дж. Велт, Р. К. Элсенпитер. - СПб. : КУДИЦ-Образ, Питер, 2005. - 384 с.
2. Богданов, С. В. Умный дом: монография / С. В. Богданов. - 2-е изд., перераб. и доп. - СПб. : Наука и Техника, 2005. - 208 с.
3. Харке, В. Умный дом. Объединение в сеть бытовой техники и систем коммуникаций в жилищном строительстве: монография / В. Харке; пер. с нем. И. В. Рядченко. - М. : Техносфера, 2006 (Чебоксары). - 287 с. : ил.
4. Сопер, М. Э. Практические советы и решения по созданию "Умного дома": самоучитель / М. Э. Сопер; пер. с англ. А. Ю. Карцева. - М. : NT Press, 2007. - 421 с.
5. Тесля, Е. «Умный дом» своими руками. Строим интеллектуальную цифровую систему в своей квартире / Е. Тесля. – СПб: Питер, 2008. – 224 с.
6. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си/ 2-е издание — М.: Триумф, 2002. – 610 с.
7. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Введ. 01.07.90. М.: Государственный комитет СССР по стандартам: Изд-во стандартов. 1989. – 28 с.
8. Arduino language reference: Arduino project [Электронный ресурс] - 2016. – Режим доступа: https://www.arduino.cc/en/Reference/HomePage, свободный.
9. BCM2835 ARM Peripherals: Raspberry Pi Foundation [Электронный ресурс] - 2016. – Режим доступа: http://www.raspberrypi.org/wp-content/uploads/2012/02/https://www.arduino.cc/en/Reference/HomePage, свободный.
10. Bell C. Beginning Sensor Networks with Arduino and Raspberry Pi - Apress, 2013. — 372 p. — ISBN-10: 1430258241, ISBN-13: 978-1430258247
11. Dennis A.K. Raspberry Pi Home Automation with Arduino - Packt Publishing, 2013. — 176 p. — ISBN-10: 1230336241, ISBN-13: 679-1230258562
12. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Николай Прохоренок. – СПб: БХВ-Петербург, 2010. - 900 с.

Приложение А. Скриншоты интерфейса пользователя

Рисунок 17 - Главная страница интерфейса управления

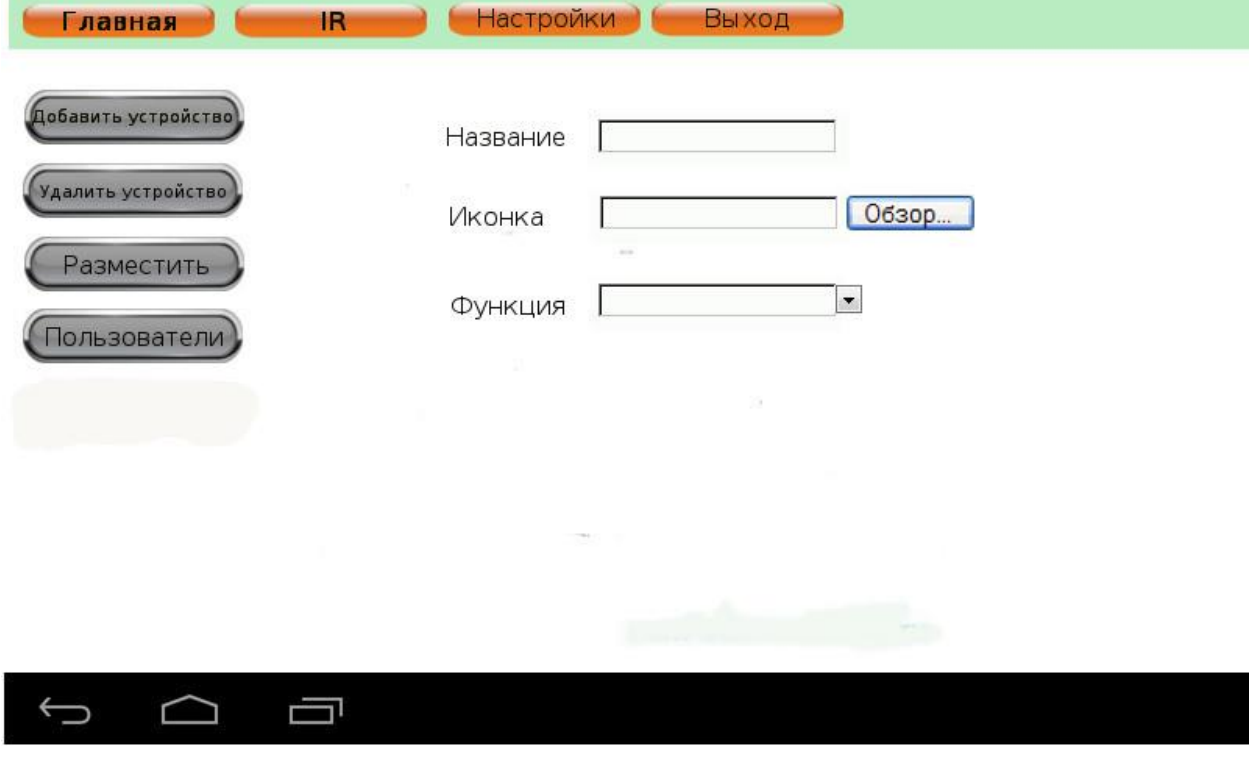

Рисунок 18 - добавление нового устройства

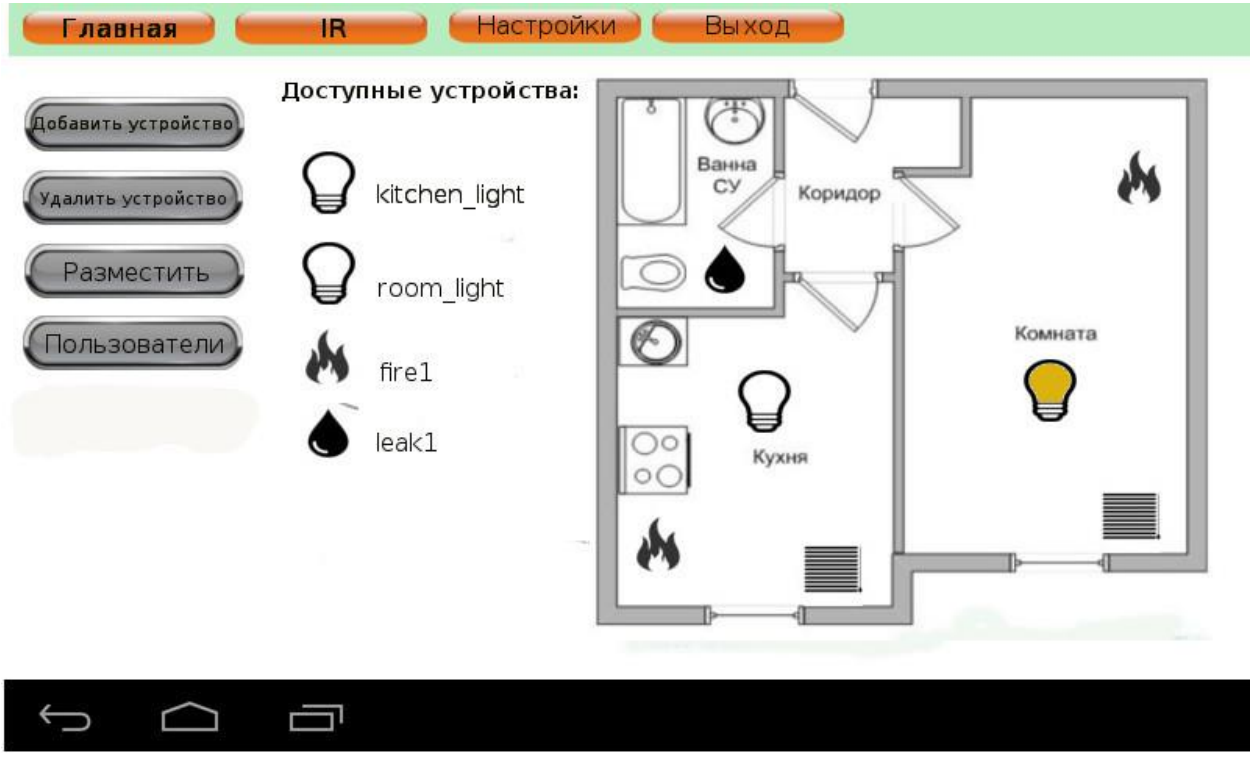

Рисунок 19 - добавление устройства на схему помещения

Рисунок 20 - настройка управления телевизором

Уважаемый пользователь! Обращаем ваше внимание, что система «Антиплагиат» отвечает на вопрос, является ли тот или иной фрагмент текста заимствованным или нет. Ответ на вопрос, является ли заимствованный фрагмент именно плагиатом, а не законной цитатой, система оставляет на ваше усмотрение.

Отчет о проверке № 1

дата выгрузки: 07.06.2016 17:30:45
пользователь: gam89@rambler.ru / ID: 3161422
отчет предоставлен сервисом «Антиплагиат»
на сайте http://www.antiplagiat.ru

Информация о документе

№ документа: 1
Имя исходного файла: diploma.docx
Размер текста: 2751 кБ
Тип документа: Прочее
Символов в тексте: 47160
Слов в тексте: 5605
Число предложений: 302

Информация об отчете

Дата: Отчет от 07.06.2016 17:30:45 - Последний готовый отчет
Комментарии: не указано
Оценка оригинальности: 91.23%
Заимствования: 8.77%
Цитирование: 0%

Оригинальность: 91.23%
Заимствования: 8.77%
Цитирование: 0%

Источники

Доля в тексте	Источник	Ссылка	Дата	Найдено в
3.9%	[1] ГОСТ 28147-89	http://ru.wikipedia.org	раньше 2011 года	Модуль поиска Интернет
3.77%	[2] ГОСТ 28147-89	http://dic.academic.ru	раньше 2011 года	Модуль поиска Интернет
3.55%	[3] Шифрование	http://ru.wikipedia.org	раньше 2011 года	Модуль поиска Интернет