МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
**Федеральное государственное автономное образовательное учреждение**
**высшего образования**
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Политехнический институт**
**Заочный факультет**
**Кафедра «Автоматизированный электропривод»**
**Направление подготовки 13.03.02 «Электроэнергетика и электротехника»**

**ДОПУСТИТЬ К ЗАЩИТЕ**

**Заведующий кафедрой**
**автоматизированного электропривода,**
**д.т.н., профессор**
_____ / М.А. Григорьев /
«_____» _____2020 г.

# РАЗРАБОТКА СИСТЕМЫ «УМНЫЙ ДОМ»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**
**ПО ПРОГРАММЕ БАКАЛАВРИАТА**
**«ЭЛЕКТРОПРИВОД И АВТОМАТИЗАЦИЯ ПРОМЫШЛЕННЫХ УСТАНОВОК**
**И ТЕХНОЛОГИЧЕСКИХ КОМПЛЕКСОВ»**
**ЮУрГУ–13.03.02.2020.858 ВКР**

**Руководитель**, должность
_____/ Р.З. Хусаинов /
«_____» _____2020 г.

**Автор работы,**
**бакалавр группы** ПЗз–576
_____/ В.В. Шмулов /
«_____» _____2020 г.

**Нормоконтролер**, должность
_____/ Т.А. Функ /
«_____» _____2020 г.

**Челябинск 2020**

# АННОТАЦИЯ

Шмулов В.В. Разработка системы «Умный дом». – Челябинск: ЮУрГУ, Э; 2020, 111 с., 24 ил., 15 табл., библиографический список – 9 наим., 2 листов чертежей ф. А3 и А2.

Целью выпускной квалификационной работы является разработка автоматизированной системы «Умный дом».

В работе, опираясь на требования и особенности проектируемой системы была разработана функциональная схема.

На основании нагрузочных диаграмм был произведен выбор и расчет двигателя методом среднеквадратичного момента. Совершён расчет и выбор редуктора. Сделаны проверки по производительности и нагреву. Построены статические и переходные характеристики.

Далее выполнен подбор реальных элементов, которые выполняют необходимые функции в составе общей схемы. В этой части работы учитывались доступность и простота оборудования.

Также была разработана система автоматизации для ворот и элементов внутри дома, построены две соответствующие принципиальные электрические схемы.

Были построены блок-схемы алгоритма событий, на основе которых написан код для трёх программируемых устройств на языке C++.

ОГЛАВЛЕНИЕ

# ВВЕДЕНИЕ

Что такое «Умный дом»? Такими словами принято описывать ряд технологий, которые в совокупности направлены на обеспечение средствами автоматического контроля и управления жилого пространства, с целью придания более комфортного и удобного окружения для человека.

Основой и первыми шагами для автоматизации дома стало появление в начале двадцатого столетия электрических приборов для упрощения выполнения бытовых задач (пылесос, холодильник, утюг с регулятором температуры, микроволновая печь и т.д.). А первые попытки создать домашнюю автоматизацию в современном понимании были проведены в середине двадцатого века. Самым ярким примером стал «Кнопочный дом», разработанный американским инженером Эмилем Матиасом в 1950 году. О нем и его доме в этом же году вышла статья в популярном журнале тех времен «Popular Mechanics». Его система была создана для конкретного 6-тикомнатного дома, где при помощи кнопок можно было управлять дверьми гаража, включать и выключать радио дистанционно и т.д. Кроме этого присутствовала система сигнализации. Вся проводка и исполнительные механизмы были расположены внутри стен дома.

Но вот официальным годом создания системы «Умного дома» и рождением самого термина является 1975 год. Событием, которое поспособствовало этому стала разработка первого стандартизированного решения для управления домашними устройствами от компании Pico Electronics. Этот стандарт разработчики именовали как X10. Для передачи и принятия сигналов в нём использовалась обычная электрическая сеть. Но также инженеры смогли предусмотреть и беспроводное управление при помощи радиочастоты. Инновация компании позволяла включать и выключать электрические приборы, менять яркость освещения, получать информацию о состоянии приборов. Для управления системой были сконструированы специальные пульты.

Данная система получила широкое распространение и множество положительных отзывов благодаря простоте установке и низкой цене.[1]

От начала двадцать первого века и по сегодняшний день технологии для создания «Умных домов» развиваются весьма быстрыми темпами. Уже существуют такие системы, как например: управление любой электрической нагрузкой в доме благодаря умным розеткам; управление освещением; домофонная система; охрана и сигнализация аварийных ситуаций (пожар, утечка газа, утечка воды и т. д.); системы с различными датчиками, которые могут отследить состав и температуру воздуха в комнате и на основе этих данных сформировать управляющее воздействие на нагреватель, вентиляцию, либо увлажнитель. Большинство готовых структур для дома имеют возможность управления с мобильного устройства, что делает их весьма удобными, так как современный человек привык к тому, что вся необходимая информация и функции находятся под рукой. А когда с помощью несложных команд от привычного всем устройства (смартфона) можно менять окружающую обстановку – это во много раз захватывающе, даже для привыкшего к чудесам техники, современного человека.

Существует множество готовых решений для задач разной сложности, но с возникновением идеи внедрения умных технологий в свой дом, каждый столкнётся с такой основной проблемой как цена и большое разнообразие. Разнообразие «умных» устройств проще рассматривать приведя их классификацию по основным явно выраженным признакам. Системы бывают:

- проводные,
- беспроводные,
- централизованные,
- децентрализованные.

В проводных системах все сигналы от устройства к устройству поступают по металлическим проводникам.

К достоинствам данного типа связи можно отнести: надежность (каждый сигнал подаётся отдельным проводом и возникновение ошибки исключено), быстрота отклика системы (не нуждается в затрачивании времени на обработку), долгий срок эксплуатации (не имеет аккумуляторов), интегрирование (в проводные системы легче интегрировать другие устройства). А недостатками могут стать: проектирование (чертежи и схемы для монтажа), прокладка проводов (прокладка должна осуществляться на стадии строительства или капитального ремонта, чтобы скрыть проводку), потребность в щите (в щите располагаются все устройства и жгут проводов), монтаж (расключение проводов по схеме).

В беспроводных системах устройства связаны между собой радиоканалом, а не проводами. На определенной частоте происходит обмен пакетами данных, которые посылаются ведущими и считываются ведомыми устройствами. Основные достоинства: уменьшение количества проводов, не обязательно внедрение на стадии ремонта (можно устанавливать в готовый интерьер), не нужен проект. Недостатки: радиоканал (связь может быть плохой из-за преград или расстояний), питание (для поддержания сигнала радиосвязи устройство должно иметь свой источник питания, и если это батарейки, то они нуждаются в регулярной замене), частота работы (беспроводные приборы в доме должны передавать пакеты данных на разных частотах, иначе на одной и той же частоте может возникнуть «зашумленность» и, как следствие, некорректная работа. В многоквартирных домах этот фактор значительно выше, т.к. в эфире радиочастот могут быть соседние аналогичные устройства умного дома или другая техника).

Централизованный умный дом имеет один центральный программируемый модуль с множеством входов и выходов. Централизованными могут быть как проводные, так и беспроводные системы. К достоинствам относятся: управление всеми приборами и мониторинг с одного интерфейса, возможность создавать сложные сценарии (как правило центральные модули имеет большой ресурс памяти и вычислительной способности).

Недостатки: программный код (имеет большой объём и могут быть ошибки из-за человеческого фактора), надежность (если контроллер выйдет из строя, то перестанет работать вся система).

В децентрализованных конфигурациях все элементы имеют свой микроконтроллер и соответствующую программу. Достоинства: надежность (при выходе из строя одного модуля, все остальные остаются работать), «гибкость» (распределенные элементы могут легко добавляться и удаляться из общей структуры), большой выбор интерфейсов для управления, популярность. Недостатки: громоздкость (из-за большого числа устройств). [2]

Для сравнения доступных функции, технологий и цен возьмём несколько производителей современного оборудования для дома.

Например, неизвестный производитель предлагает стартовый комплект системы удаленного управления двумя розетками под названием Switching Lite. В эту систему входят: центральный контроллер и два накладных реле, которые подключаются в евророзетку. Управление осуществляется со смартфона. За этот комплект оборудования придется заплатить 15 тыс. рублей. Для расширения у продавца есть достаточно большой выбор диммеров для розеток и освещения, различные датчики и т.д. Но за всё это необходимо доплачивать.

Следующий - центр управления Rubetek CC1. Очень продуманная система с множеством интерфейсов для связи. Головной контроллер собирает информацию с устройств и оповещает владельца обо всем, что происходит в доме с помощью смс и push уведомлений. Цена одного контроллера 8 тыс. руб. Средняя цена на различное дополнительное оборудование, датчики, реле и т.д. – 1300 руб. за один модуль.

Самым бюджетным и наиболее интересным станет Mi Smart Home Multifunction Gateway 2 от популярного китайского бренда Xiaomi. Как и во всех современных системах имеет контроллер для связи с остальным оборудованием, который стоит порядка 2500 руб. К этому контроллеру пользователь, как и в предыдущих вариантах подключается со смартфона.

Затем нужно приобрести стартовый набор датчиков и реле, который обойдется в 5000 руб. Итого мы имеем: контроллер, который подключается к домашнему роутеру, датчик движения, датчик положения входной двери, и пару выключателей нагрузки. А также хорошее исполнение, совмещающее качество сборки, функциональность и приятную цену.

В разрабатываемом проекте будут использованы схожие технологии как в готовых коммерческих предложениях от известных производителей. Но данный проект ориентирован на использование модульной электроники, предназначенной для широкого применения, с её помощью разработчики могут создавать любые проекты, связанные с автоматизацией, прототипы приборов различного уровня сложности. Модули не имеют готового корпуса, «прошивки» или источника питания, а лишь плату с распаянными элементами. Преимущество таких модулей, опять же – цена и общая доступность, как самих модулей, так и программного обеспечения к ним. Разработанное устройство сможет легко повторить даже начинающий изучение микроконтроллеров и электроники человек. Использование в платах микроконтроллеров известных производителей гарантирует надежность работы таких устройств. Ещё одним преимуществом данных модулей для свободной разработки является то, что они используют стандартные интерфейсы для общения между устройствами.

# 1 РАЗРАБОТКА ФУНКЦИОНАЛЬНОЙ СХЕМЫ

## 1.1 Описание систем и элементов входящих в схему

### 1.1.1 Элементы, находящиеся вне дома (автоматические ворота, средства управления, идентификация)

Для автоматического открытия ворот понадобится внедрение системы, способной на расстоянии идентифицировать владельца дома без какого-либо контакта при помощи радиочастотных меток и подать сигнал на открытие. Положение створки будет зависеть от того, въезжает во двор автомобиль или входит человек. Принятие решения в сторону того, как должны открыться ворота станет возможным благодаря считыванию разных меток с уникальным идентификатором. Если въезжает автомобиль (используется метка для автомобиля), то ворота полностью распахиваются, а если человек (используется метка для человека), то приоткрываются (положение – калитка). Пока метка находится вблизи ворот – ворота открыты, а когда удалена дальше, чем расстояние считывания и прошло определенной время – закрыты. Дополнительно управлять воротами можно с мобильного устройства или персонального компьютера посредством интерфейса.

Для перемещения створки понадобится электропривод с звеном, формирующим управление силовой частью и обеспечивающее реверс, датчик для отслеживания конечных положений ворот, а также датчик, сообщающий о препятствии на пути хода створки. Так как разрабатываемая схема предназначена для бытового, а не промышленного использования, необходимо учитывать, что питание электродвигательного аппарата будет производится от однофазной сети с номинальным напряжением 220 В. Звеном, которое сможет управлять электродвигателем привода ворот и обеспечит реверсивное движение, может стать какой-либо преобразователь электрических величин.

Преобразование параметров питающего напряжения даст доступ к большому выбору электродвигателей. Благодаря датчику, который будет отслеживать положение ворот в данный момент времени появится возможность сформировать соответствующий сигнал для движения или остановки. А вот датчик препятствия необходим, чтобы исключить аварийную ситуацию, когда при движении ворот в сторону закрытия на их пути окажется препятствие. Логику работы ворот обеспечит программируемое устройство, которое также будет связано с общей системой телемеханики.

### 1.1.2 Элементы, находящиеся внутри дома (управление светом, розетками, безопасность, идентификация, естественное освещение)

Чтобы осуществить дистанционное включение/отключение источников света, розеток, либо управлять перемещением ворот или жалюзи, нужно использовать технологии беспроводной связи, а именно WiFi сеть. Для этого понадобится программируемое устройство, на котором станет возможным создание визуального интерфейса, оно будет принимать и отправлять информацию по сети и объединит все остальные устройства схемы. Когда пользователь интерфейса подаст соответствующую команду (включить, отключить, регулировать, перемещать), она обработается и сформируется сигнал на нужное действие для объекта (лампочка, нагреватель, ворота и т.д).

Сенсорное управление будет выполнено при помощи датчика прикосновения и расположено в доступном месте, заменяя привычный клавишный выключатель. Замена клавишного выключателя является побочным решением, т.к. такое внедрение достаточно просто осуществить, а преимуществом этого будут – удобство, бесшумное нажатие и надежность (т.к. отсутствует механический узел).

Программирование времени включения/отключения для любого канала управление станет доступно дистанционно, будь то освещение или розетка.

Это очень полезная функция для умного дома. Программирование заключается в установке пользователем уставок времени в интерфейсе умного дома, а система в свою очередь запустит таймер, по истечении которого канал включится или отключится в соответствии с программой. Таймеры доступны для всех каналов и будут работать независимо. Применение такой функции, как таймеры может пригодится, когда необходимо, например включить обогрев во время глубокой фазы сна или наоборот выключить ночник и вентилятор, которые нужны только некоторое время до сна и т.д.

Назначение управляемого естественного освещения в том, чтобы препятствовать солнечным лучам попадать через окна в дом, и тем самым обеспечить рассеянное освещение и благоприятную температуру. Или наоборот, когда не хватает света от осветительных приборов и температура в доме низкая, излучение целесообразно пропустить внутрь дома. Для удобства управление естественным освещением разместится в интерфейсе умного дома. Осуществить данную функцию станет возможным оснастив окна дома специализированными жалюзи с электроприводом.

Функция охранной системы дома заключается в оповещении о взломе и включении отпугивающей сирены. Охранная система срабатывает, когда она находится в активном состоянии и получен сигнал от датчика движения или от датчика положения входной двери. Для активации системы необходимо подать соответствующую команду с интерфейса управления или пройти идентификацию метки, как владелец. Активность сигнализации может быть снята, также идентификацией владельца за отведенное время, либо дистанционно со смартфона. В случае срабатывания сигнализации, раздастся сирена и пройдет оповещение о взломе в форме смс сообщения или звонка на указанные номера телефонов. В охранной системе также предусмотрена работа от автономного источника питания, в тех случаях, если произошло умышленное или случайное обесточивание схемы.

Чтобы обеспечить соединение между системами внутри дома и вне его, не используя сигнальный кабель, в функциональную схему будет добавлен блок беспроводной связи, он объединит логические устройства, и управление будет доступным с одного интерфейса. Также необходимо обеспечить обе системы самостоятельными блоками питания.

## 1.2 Функциональная схема

Функциональная схема – это документ, на котором изображаются и разъясняются процессы протекающие в отдельных узлах устройства или установки, а также взаимосвязь между самими устройствами или установками.

Все элементы на схеме располагаются наиболее удобным способом, отличающимся от их фактического размещения. Каждая функциональная часть или деталь представляется в виде прямоугольника с сокращенным обозначением. Точно так же отображаются связи между ними в виде линий со стрелками, позволяющие проследить взаимодействие и влияние на общую работоспособность.

Перечислим процессы, которые войдут в проектирование:

- автоматическое открытие/закрытие ворот;
- дистанционное, сенсорное включение/выключение, а также регулирование освещения;
- дистанционное управление любой нагрузкой, подключенной к розетке;
- возможность программирования времени включения/отключения розеток и освещения.
- управление естественным освещением;
- охранная сигнализация.

На основе вышеперечисленных функций, заложенных в проект, построим функциональную схему (рисунок 1.1). [3]

На функциональной схеме изображены два блока питания (БП1, БП2), БП1 питает устройства, которые находятся за пределами дома и входят в систему ворот. БП2 питает всю автоматику, находящуюся в доме. Питание с БП1 поступает на первое управляющее устройство (УУ1). К УУ1 присоединены датчик препятствия (ДПР) и датчик положения (ДП). С ДПР и ДП УУ1 получит необходимые сигналы и опираясь на них запустит соответствующие алгоритмы: если препятствие, то открывать; если достигли нужного положения, то остановить. Устройство идентификации (УИ1) связано с УУ1, с помощью него радиочастотный сигнал воздействует на метку, идентифицирует её и передаёт по цифровому протоколу информацию о считанной метке УУ1. Выходные сигналы УУ1 передаются преобразователю (П), а преобразователь воздействует на двигатель привода ворот (Д1). С БП2 напряжение питания поступает на второе управляющее устройство УУ2. Схема резервного питания (РП) подключена параллельно основному питанию. По отдельными линиям связи на вход контроллера поступают сигналы с сенсорных датчиков прикосновения (СД1-3). Сигнальные линии датчиков движения (ДД1-3) объединены и уже по одной линии передаются на вход УУ2. На основе сигналов с СД1-3 УУ1 посылает выходной сигнал на модули управления нагрузкой (УН1-3), которые коммутируют напряжение сети 220 В на нагрузку. А при получении сигнала с хотя бы одного из ДД1-3 посылает сигнал на воспроизведение звука сирены сигнальному устройству (СУ) и сигнал на отправку уведомления модулю сотовой связи (СС). К УУ2 подключено собственное устройство идентификации (УИ2). Также с выходных каналов УУ2 сигнал идет в схему управления электроприводом (СУЭ), а СУЭ непосредственно управляет двигателем жалюзи (Д2). РС1 и РС2 подключены к УУ1 и УУ2 соответственно, чтобы обеспечить беспроводную радиосвязь.

## 1.3 Требования к воротам и элементам вне дома

Ворота в проекте откатного типа, которые подразумевают самостоятельное изготовление или изготовление по заказу.

Размеры створки ворот 2 на 5 м. Каркас створки и перекладины могут быть изготовлены из труб прямоугольного профиля, к перекладинам прикручивается профлист. Каркас имеет массу 130 кг. Для обшивки ворот необходимо 10 $м^2$ профнастила. Один квадратный метр листа профнастила весит 3,2 кг, значит общая масса обшивки составит 32 кг. Вся конструкция створки имеет массу162 кг. Для перемещения створки откатных ворот понадобятся: направляющий рельс, зубчатая рейка, роликовые опоры, верхняя пластина с роликами, нижний концевой ролик, нижний улавливатель, верхний улавливатель, резиновые заглушки.

Из всего перечисленного в общую массу, которую необходимо будет перемещать, при помощи электропривода войдут: створка, направляющий рельс, зубчатая рейка, нижний концевой ролик, резиновые заглушки. Масса створки 162 кг, направляющий рельс обладает массой 32 кг, а зубчатая рейка 10 кг. Массой остальной фурнитуры на створке можно пренебречь. Итого общая масса створки составит 204 кг.  Две роликовые опоры будут учтены, как источники трения.

Для механизма ворот в ходе проектирования необходимо подобрать редуктор, который входит в зацепление с зубчатой рейкой, а редуктор будет приводиться в движение асинхронным однофазным или трехфазным двигателем. Засчет редуктора будет проще добиться необходимой скорости перемещения и уменьшить требуемую мощность электродвигателя. Управлять двигателем будет частотный преобразователь, при помощи которого планируется осуществить плавный пуск и безрывковое перемещение створки ворот для долгосрочной службы, как самого электродвигателя, так и всех механизмов.

Удобство использования частотного преобразователя ещё и в том, что он имеет дискретные программируемые входы что упрощает согласование с другими цифровыми устройствами.

Распознавать личность владельцев и отличать их от транспорта решено при помощи RFID (англ. Radio Frequency IDentification, радиочастотная идентификация). Считывателю следует находится недалеко от ворот, а уникальная метка должна быть активной и иметь свой источник питания, чтобы радиус действия метки был от 5 до 20 м. Используя активные метки ворота успеют заблаговременно открыться и автомобиль или человек пересекут ворота комфортно без ожидания. При этом метку не нужно доставать, как ели бы использовался пульт управления, она лежит в машине или находится в сумке, кармане.

При дальнейшем проектировании необходимо те датчики, которые расположатся на улице выбрать с соответствующей степенью защиты от погодных условий и загрязнений.

Программируемый логический контроллер для ворот обеспечит согласование между всеми электрическими компонентами схемы, поэтому он должен иметь достаточное количество входов/выходов и возможность подключения к нему модуля радиосвязи и считывателя RFID меток.

## 1. 4 Требования к остальным элементам внутри дома

Управляемое естественное освещение будет выполнено при помощи жалюзи с электроприводом постоянного тока и редуктором в одном корпусе.

Мощность электродвигателя жалюзи 11 Вт. На выходном вале редуктора при питании 12 В скорость вращения будет составлять 30 оборотов в минуту. Привод будет связан своим валом с валиком, на который наматывается ткань.

Валик, на который наматывается ткань имеет диаметр 25 мм, а сама ткань имеет длину 1500 мм, значит без учёта утолщения валика от уже намотанной ткани полное открывание, либо закрывание жалюзи произойдет примерно за 19 оборотов, что соответствует 38-ми секундам по времени. Движение вверх и вниз обеспечит схема из двух пар транзисторов, либо специальная драйверная микросхема, переключением которых будет управлять логический контроллер. Любой из вариантов должен производить смену полярности подаваемого на электродвигатель жалюзи напряжения и тем самым менять направление вращения. Остановка движения произойдет по окончании заложенного времени на закрытие/открытие.

Чтобы питать осветительные приборы в структуре должны присутствовать приборы способные коммутировать нагрузку, рассчитанную на сетевое напряжение 220 В тогда, когда получат низковольтный сигнал от управляющего ими модуля. Для этой задачи подойдут твердотельные реле.

Чтобы управлять напряжением, подаваемым на розетки, также понадобятся модули, принимающие низковольтный сигнал, и в соответствии с сигналом они включат или отключат подключенную к розетке нагрузку.

Работа охранной сигнализации не должна зависеть от каких-либо других одновременно протекающих процессов, чтобы выполнять свою задачу без задержек. Чтобы воспроизводить звук сирены нужно устройство воспроизведения звуков. Отправка смс или звонка на указанные номера подводит к дальнейшему выбору модуля способного работать с современными GSM сетями. Так как охранная система будет иметь бесперебойное питание, то переключение от основного источника на дополнительный должно обеспечиваться мгновенно, чтобы не допустить перезагрузки системы. Идентификацию, как и в случае со схемой ворот надлежит выполнить используя RFID. Отличие лишь в том, что в доме не нужно применять активные метки.

Элементы, обеспечивающие выполнение всех изложенных функций и задач для «Умного дома» должны иметь низкую цену, чтобы оправдать затраты времени на разработку. Выбранные устройства составляющие общую схему должны быть современными в используемых технологиях, иметь компактную архитектуру, быть энергоэффективными.

При разработке должно быть учтено свободное добавление и удаление объектов управления, т.е. чтобы разработанное в итоге устройство можно было легко изменять под необходимые условия.

# 2 РАСЧЕТ ЭЛЕКТРОДВИГАТЕЛЯ ДЛЯ ОТКАТНЫХ ВОРОТ

## 2.1 Исходные данные для расчета

Рисунок 2.1 – Кинематическая схема откатных ворот

1 – створка откатных ворот, 2 – роликовые опоры, 3 – зубчатая рейка, 4 – зубчатое колесо на выходе редуктора.

Таблица 2.1 – Исходные данные

| **Наименование показателя** | **Обозначение** | **Размерность** | **Величина** |
|---|---|---|---|
| Масса ворот | $\boldsymbol{m}$ | кг | 204 |
| Диаметр зубчатого колеса | $\boldsymbol{D_К}$ | м | 0,1 |
| Ширина колеса | $\boldsymbol{b}$ | м | 0,04 |
| Диаметр шейки вала | $\boldsymbol{d_Ш}$ | м | 0,03 |

Продолжение таблицы 2.1

| **Наименование показателя** | **Обозначение** | **Размерность** | **Величина** |
|---|---|---|---|
| Крутящий момент на валу | $M_K$ | Нм | 10 |
| Путь | $L$ | м | 5 |
| Рабочая скорость | $V_р$ | м/с | 0,5 |
| Скорость возврата | $V_в$ | м/с | 0,5 |
| Среднее ускорение | $a_{доп}$ | м/с$^2$ | 0,5 |
| Время работы | $t_р$ | с | 24 |
| Число циклов | $z$ | 1/ч | 10 |
| Момент инерции | $J$ | кгм$^2$ | 0,3 |

2.2 Расчет моментов статических сопротивлений и предварительный расчет мощности электродвигателя

2.2.1 Нагрузочные диаграммы скорости РО

На основе заданного пути $L$, установившейся скорости $V_y$,и среднего допустимого ускорения $a_{ДОП}$, определяем время пуска $t_П$ до рабочей скорости $V_P$ с $a_{ДОП}$ и время торможения $t_T$ от $V_P$ до остановки:

$$t_T = t_П = \frac{V_P}{a_{ДОП}} = \frac{0,5}{0,5} = 1 \text{ с}; \quad (1)$$

путь пройдены за время пуска (торможения)

$$L_T = L_П = \frac{V_P{}^2}{2 \cdot a_{ДОП}} = \frac{0,5^2}{2 \cdot 0,5} = 0,25 \text{ м}; \quad (2)$$

время установившегося режима движения со скоростью $V_P$

$$t_У = \frac{L - (L_T + L_П)}{V_P} = \frac{5 - (0,25 + 0,25)}{0,5} = 9 \text{ с}. \quad (3)$$

Так как по условию рабочая скорость равна скорости возврата $V_P = V_B$, то время и путь тоже будут одинаковы.

Тогда полное время работы:

а) рабочий ход $t_P = t_T + t_П + t_У = 1 + 1 + 9 = 11$ с; (4)

б) обратный ход $t_{Об} = t_T + t_П + t_У = 1 + 1 + 9 = 11$ с; (5)

в) полное время работы $T = t_P + t_{Об} = 11 + 11 = 22$; (6)

г) если число циклов в час 10, следовательно, 1 цикл по времени занимает 360 секунд, отсюда время паузы:

$$t_{пауз.} = 360 - 22 = 338 \text{ с}, \quad (7)$$

$$t_{пауз.Р} = \frac{t_{пауз.}}{2} = \frac{338}{2} = 169 \text{ с}, \quad (8)$$

$$t_{пауз.Об} = \frac{t_{пауз.}}{2} = \frac{338}{2} = 169 \text{ с.} \quad (9)$$

Результаты расчета вводятся в таблицу 2 и строятся нагрузочные диаграммы v(t) на рисунке 1

Рисунок 2.2 – Нагрузочные диаграммы скорости

2.2.2 Нагрузочные диаграммы моментов РО

Для механизма откатных ворот по известной кинематической схеме следует определить составляющие моментов, основные рабочие усилия, места приложения сил трения.

Результаты расчета составляющих статических моментов на валу рабочего органа на каждом из участков работы механизма вводятся в таблицу 2. Статический момент на каждом участке определяется суммой всех составляющих

$$M_{РОСТ} = \Sigma M \, . \quad (10)$$

Таблица 2.2– Данные рабочего органа РО по участкам движения

| Участок движения | | Рабочий ход | | | Обратный ход | | |
|---|---|---|---|---|---|---|---|
| Расчетные данные | Обозначения | Пуск | Уст. | Торможен. | Пуск | Уст. режим | Торможен. |
| Скорость, м/с | $v_{РО}$ | 0 | 0,5 | 0 | 0 | -0,5 | 0 |
| Время работы, с | $t_{РО}$ | 1 | 9 | 1 | 1 | 9 | 1 |
| Путь, м | $L_{РО}$ | 0,25 | 4,5 | 0,25 | 0,25 | 4,5 | 0,25 |
| Моменты РО, Нм: | | | | | | | |
| -трение скольжения в подшипниках | $M_{ТП}$ | 0,45 | | | -0,45 | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -сила трения качения | $M_{TK}$ | 20,01 | | | -20,01 | | |
| – момент трения на валу зубчатого колеса | $M_{ЗК}$ | 20,05 | | | -20,05 | | |
| Статический момент, Нм | $M_{РОСТ}$ | 40,1 | | | -40,1 | | |
| Момент инерции, $кгм^2$ | $J_{РОСТ}$ | 0,8 | | | 0,8 | | |
| Динамический момент, Нм | $M_{РОДИН}$ | 8 | 0 | -8 | -8 | 0 | 8 |
| Суммарный момент, Нм | $M_{РО}$ | 48,1 | 40,1 | 32,1 | -48,1 | -40,1 | -32,1 |

Составляющие статических моментов состоят из:

- статического момента сил трения в подшипниках:

$$M_{ТП} = \frac{m_1 \cdot d_{Ш} \cdot \mu_{П} \cdot g}{2}; \quad (11)$$

где $m_1$ – масса деталей и узлов, опирающихся на подшипники, кг;

$d_{Ш}$ – диаметр шейки вала или оси, м;

$\mu_{П}$ – коэффициент трения скольжения в подшипниках (0,015...0,02);

$g$ = 9,81 м/с$^2$ – ускорение силы тяжести.

- момент сил трения качения:

$$M_{ТК} = m \cdot f \cdot g; \quad (12)$$

где $f$ – коэффициент трения качения (для роликоподшипника 0,0025… 0,01);

m– масса движущегося тела, масса деталей, опирающихся на узел качения, кг.

- момент трения на валу зубчатого колеса

$$M_{ЗК} = \frac{D_K}{2} + f_K k P; \quad (13)$$

где $D_K$ – диаметр зубчатого колеса (звѐздочки), находящегося на выходном валу редуктора и преобразующего вращение в поступательное движение тела, м;

$f_K$ - условный коэффициент трения (для шлифованых зубьев 0,08…0,1);

$k$ - коэффициент, мм (для нормального прямозубого зацепления 0,919…1,21);

$P$ - окружное усилие $P = 2 \cdot M_K / D_K = 20/0{,}1 = 200$ Н. [4] (14)

Найдем по соответствующим формулам статические моменты при движении механизма ворот:

$$M_{ТП} = \frac{204 \cdot 0,03 \cdot 0,015 \cdot 9,81}{2} = 0,45 \text{Нм};$$

$$M_{ТК} = 204 \cdot 0,01 \cdot 9,81 = 20,01 \text{ Нм};$$

$$M_{ЗК} = \frac{0,1}{2} + 0,1 \cdot 1 \cdot 200 = 20,05 \text{Нм};$$

$$M_{РОСТ} = 0,45 + 20,01 + 20,05 = 40,1 \text{ Нм}.$$

Результаты расчета вводяться в таблицу 2.2 и строятся нагрузочные диаграммы $M_{РО}$ на рисунке 2.3.

Рисунок 2.3 – Нагрузочные диаграммы $M_{РО}$

Для определения динамических моментов рассчитываются моменты инерции рабочего органа:

$$J_{РО} = J + m \cdot \frac{D_К}{4}, \quad (15)$$

где $J$ – момент инерции вращающегося элемента рабочей машины, кгм$^2$;

$m$ – масса движущегося тела, кг.

Найдем момент инерции для РО по формуле:

$$J_{РО} = 0,3 + 204 \cdot \frac{0,1^2}{4} = 0,8 \text{ кг/м}^2.$$

При заданной величине допустимого ускорения $а_{доп}$ определяется динамический момент:

$$M_{РОдин} = J_{РО} \cdot \frac{2 \cdot a_{доп}}{D_К} = 0,8 \cdot \frac{2 \cdot 0,5}{0,1} = 8 \text{ Нм}. \quad (16)$$

Полный момент для пуска:

$$M_{РОп} = M_{РОСТ} + M_{РОдин} = 40{,}1 + 8 = 48{,}1 \text{ Нм}. \quad (17)$$

Полный момент для торможения:

$$M_{РОт} = M_{РОСТ} - M_{РОдин} = 40{,}1 - 8 = 32{,}1 \text{ Нм}. \quad (18)$$

Результаты расчетов скоростей и моментов для каждого участка движения с учётом направления приводим в таблицу 2. По результатам расчетов с учетом времени пуска, торможения, установившегося движения на рисунке 5 строим нагрузочную диаграмму моментов рабочего органа для каждого режима работы $M_{РО}(t)$.

2.2.3 Расчет мощности двигателя

На основании построенной нагрузочной диаграммы момента рабочей машины можно рассчитать:

– среднеквадратичное значение момента, в котором учтены статические нагрузки и часть динамических нагрузок:

$$M_{\text{сркв}} = \sqrt{\frac{M_{РОп1}{}^2 \cdot t_{П1} + M_{РОСТ1}{}^2 \cdot t_{У1} + M_{РОт1}{}^2 \cdot t_{Т1} + M_{РОп2}{}^2 \cdot t_{П2} + M_{РОСТ2}{}^2 \cdot t_{У2} + M_{РОт2}{}^2 \cdot t_{Т2}}{t_{П1} + t_{У1} + t_{Т1} + t_{П2} + t_{У2} + t_{Т2}}} \quad , (19)$$

где $M_{РОп1}$ – полный момент на участке пуска в прямом направлении;

$M_{РОСТ1}$ – статический момент на установившемся участке в прямом направлении;

$M_{РОт1}$ – полный момент на участке торможения в прямом направлении;

$M_{РОп2}$ – полный момент на участке пуска в обратном направлении;

$M_{РОСТ2}$ – статический момент на установившемся участке в обратн. направлении;

$M_{РОт2}$ – полный момент на участке торможения в обратном направлении;

$t_{П1}$ – время пуска в прямом направлении;

$t_{У1}$ – время усстановившегося режима в прямом направлении;

$t_{Т1}$ – время торможения в прямом направлении;

$t_{П2}$ – время пуска в обратном направлении;

$t_{У2}$ – время усстановившегося режима в обратном направлении;

$t_{Т2}$ – время торможения в обратном направлении.

$$M_{\text{срКВ}} = \sqrt{\frac{48{,}1^2 \cdot 1 + 40{,}1^2 \cdot 9 + 32{,}1^2 \cdot 1 + 48{,}1^2 \cdot 1 + 40{,}1^2 \cdot 9 + 32{,}1^2 \cdot 1}{1+9+1+1+9+1}} = 40{,}25\,\text{Нм}$$

– время цикла при заданном z – числе циклов работы машины в час:

$$t_{Ц} = \frac{3600}{z} = \frac{3600}{10} = 360\,\text{с}. \quad (20)$$

– продолжительность включения $\text{ПВ}_{\text{ФАКТ}}$ по времени работы t на всех участках движения времени:

$$\text{ПВ}_{\text{ФАКТ}} = \frac{1}{t_{Ц}} \cdot \sum (t_{П1} + t_{У1} + t_{Т1} + t_{П2} + t_{У2} + t_{Т2}) \cdot 100\% \quad (21)$$

$$\text{ПВ}_{\text{ФАКТ}} = \frac{1+9+1+1+9+1}{360} \cdot 100 = 6.$$

Расчетная мощность двигателя может быть определена по соотношению:

$$P_{ДВ} = k_1 \cdot M_{\text{срКВ}} \cdot \frac{2 \cdot V_P}{D_K} \cdot \sqrt{\frac{\text{ПВ}_{\text{ФАКТ}}}{\text{ПВ}_{\text{КАТ}}}} = 1{,}5 \cdot 40{,}25 \cdot \frac{2 \cdot 0{,}5}{0{,}1} \cdot \sqrt{\frac{6}{40}} = 236{,}5\,\text{Вт}. \quad (22)$$

где $k_1$ – коэффициент, учитывающий динамические нагрузки, обусловленные вращающимися элементами электропривода (двигатель, редуктор), а также потери мощности в редукторе (1,3…1,5);

$\text{ПВ}_{\text{КАТ}}$ – ближайшее к $\text{ПВ}_{\text{ФАКТ}}$ каталожное значение относительной продолжительности включения для электродвигателей выбранной серии.

## 2.3 Выбор двигателя

Выбор двигателя произведем по каталогу электродвигателей серии А, АИР. Выберем двигатель так, чтобы значение мощности было равно или несколько больше мощности, которую рассчитали в предыдущем пункте. [5]

Таблица 2.3 – Технические характеристики двигателя

| Марка | $P_Н$, кВт | $n$, Об/мин | $U_Н$, В | $I_Н$, А | $КПД$, % | $\cos\varphi$ | $I_П$, А | $J$, кгм$^2$ | $M_П$, Нм | $M_{КРИТ}$, Нм | $M_Н$, Нм |
|---|---|---|---|---|---|---|---|---|---|---|---|
| А71А2 | 0,75 | 2835 | 380 | 1,8 | 78 | 0,74 | 5,3 | 0,0006 | 2,8 | 2,8 | 2,5 |

2.4 Выбор редуктора

Передаточное число редуктора определим по известной номинальной скорости вращения выбранного электродвигателя и по основной скорости рабочего органа:

$$j_P = \frac{\omega_Н \cdot D_К}{2 \cdot V_P}, \tag{23}$$

где $\omega_Н$ - номинальная скорость вращения двигателя, $\frac{1}{c}$.

$$\omega_Н = \frac{\pi \cdot n}{30} = \frac{3{,}14 \cdot 2835}{30} = 296{,}8 \ \frac{1}{\text{с}}. \tag{24}$$

$$j_P = \frac{296{,}8 \cdot 0{,}1}{2 \cdot 0{,}5} = 29{,}7.$$

Выбранный редуктор должен иметь передаточное число, равное или несколько меньшее расчетного значения.

По каталогу был выбран планетарный двухступенчатый редуктор ЗП-25, который в паре с двигателем образуют мотор-редуктор. Редукторы с планетарной передачей обладают таким преимуществом как малые габариты и масса, что очень подходит для проекта. Высота, на которой расположен выходной вал выбранного редуктора тоже небольшая, что обеспечит удобное сочленение узлов механизма ворот.

Рисунок 2.4 – Внешний вид редуктора с двигателем

Таблица 2.4 – Технические характеристики редуктора

| Тип передачи | Передаточное число | Максимальный крутящий момент на выходном валу, Нм | Мощность на входном валу редуктора, кВт | КПД, % |
|---|---|---|---|---|
| Планетарная | 25 | 95 | 0,55 | 95 |

[6]

2.5 Приведение статических моментов и моментов инерции к валу двигателя

При составлении расчетной схемы механической части электропривода моменты сопротивления движению РО (статические моменты) и моменты инерции приводятся к валу двигателя.

Статические моменты рабочей машины, приведенные к валу двигателя, без учета потерь в редукторе ($\eta_P = 1$), рассчитываются по формуле:

$$M_{ПР} = \frac{M_{РОСТ}}{j_P} = \frac{40,1}{25} = 1,6\,\text{Нм}. \quad (25)$$

Статические моменты на валу с учетом потерь в редукторе ($\eta_P < 1$) рассчитывают в зависимости от режима работы электропривода. Статический момент на валу в двигательном режиме:

$$M_{ДВС} = \frac{M_{ПР}}{\eta_P} = \frac{1,6}{0,95} = 1,68\,\text{Нм}. \quad (26)$$

При работе электропривода в тормозных режимах потери в редукторе вызывают уменьшение нагрузки двигателя, при этом момент на валу определяют по формуле:

$$M_{ТВС} = M_{ПР} \cdot \eta_P = 1,6 \cdot 0,95 = 1,52\,\text{Нм}. \quad (27)$$

Приведенные статические моменты системы «электропривод – рабочая машина» рассчитывают для каждого участка с учетом режима работы электропривода по формуле:

$$M_C = M_{BC} \pm M_{XX}, \quad (28)$$

где $M_{XX}$ - момент холостого хода двигателя, Нм.

Момент холостого хода электродвигателя определим по формуле:

$$M_{XX} = M_H - M_B, \quad (29)$$

где $M_H$ – номинальный момент электродвигателя, Нм;

$M_B$ –момент на валу электродвигателя, Нм.

Момент на валу электродвигателя равен:

$$M_B = \frac{P_н}{\omega_н}, \quad (30)$$

где $P_н$ – номинальная мощность электродвигателя, Вт;

$\omega_н$ = 314,16 – синхронная скорость электродвигателя, зависит от частоты питающей сети $f$ и числа пар полюсов $p$, $\frac{рад}{с}$.

Отсюда момент на валу электродвигателя согласно формуле (30) равен:

$$M_B = \frac{750}{314{,}16} = 2{,}38 \text{ Нм}.$$

Момент холостого хода электродвигателя:

$$M_{XX} = M_H - M_B = 2{,}5 - 2{,}38 = 0{,}12 \text{ Нм}.$$

Статический момент системы «электропривод – рабочая машина» в двигательном режиме определим по формуле:

$$M_{ДС} = M_{ДВС} + M_{XX} = 1{,}68 + 0{,}12 = 1{,}8 \text{ Нм}. \quad (31)$$

Статический момент системы «электропривод – рабочая машина» в тормозном режиме определим по формуле:

$$M_{TC} = M_{TBC} - M_{XX} = 1,52 - 0,12 = 1,4 \text{ Нм}. \quad (32)$$

При этом в статическом моменте учитываются не только силы сопротивления движению в рабочей машине, но также и потери в редукторе, и механические потери в двигателе. Суммарный приведенный к валу двигателя момент инерции системы может быть рассчитан по соотношению:

$$J = \delta J_{Д} + J_{ПР}, \quad (33)$$

где $J_{Д}$ - момент инерции ротора двигателя;

$\delta$ - коэффициент, учитывающий момент инерции остальных элементов электропривода: муфты, тормозного шкива, редуктора и др. (примем в формуле коэффициент $\delta$ = 1,5);

$J_{ПР}$ - приведенный к валу двигателя суммарный момент инерции движущихся исполнительных органов рабочей машины и связанных с ними движущихся масс (грузов, заготовки и т.п.).

Приведенный к валу двигателя момент инерции рабочей машины определим по формуле:

$$J_{ПР} = \frac{J_{PO}}{j_P^2} = \frac{0,8}{25^2} = 0,001 \text{ кгм}^2. \quad (34)$$

Тогда суммарный, приведенный к валу двигателя, момент инерции системы электропривода равен:

$$J = 1,5 \cdot 0,0006 + 0,001 = 0,0019 \text{ кгм}^2.$$

Пусковой момент определим по формуле:

$$M_{П} = M_{C} + M_{ДИН}, \quad (35)$$

где $M_{C}$ - статический момент сопротивления движению, Нм;

$M_{ДИН}$ - динамический момент, Нм.

Динамический момент движения ворот:

$$M_{ДИН} = J\frac{2 \cdot a_{ДОП} \cdot j_P}{D_K} = \frac{0,0019 \cdot 2 \cdot 0,5 \cdot 25}{0,1} = 0,475\,\text{Нм}.$$

Пусковой момент при движении ворот согласно формуле (34) составит:

$$M_{ПС} = M_{ДС} + M_{ДИН} = 1,8 + 0,475 = 2,275\,\text{Нм}. \quad (36)$$

Тормозной момент при движении ворот:

$$M_{ТРС} = M_{ДИН} - M_{ТС} = 0,475 - 1,4 = -0,925\,\text{Нм}. \quad (37)$$

Средний момент двигателя примем равным моменту, допустимому по ускорению.

При пуске:

- в прямом направлеии

$$M_{СР1} = M_{ПС} = 2,275;$$

- в обратном направлении

$$M_{СР2} = M_{ПС} = 2,275.$$

При торможении:

- в прямом направлении

$$M_{СР3} = M_{ТРС} = 0,925;$$

- в обратном направлении

$$M_{СР4} = M_{ТРС} = 0,925.$$

Установившуюся рабочую скорость двигателя находим по формуле:

$$\omega_p = \frac{2 \cdot V_p}{D_K} \cdot J_p = \frac{2 \cdot 0,5 \cdot 25}{0,1} = 250\frac{\text{рад}}{\text{с}}. \quad (38)$$

Таблица 2.6 – Приведение статических моментов и моментов инерции к валу двигателя

| Участок движения | Прямой ход | Обратный ход |
|---|---|---|

| Расчетные данные | Обозначен. | Пуск | Устан. | Торможен. | Пуск | Устан. | Торможен. |
|---|---|---|---|---|---|---|---|
| По данным таблицы 0 | | | | | | | |
| Скорость РО, м/с | $v_{PO}$ | - | 0,5 | - | - | -0,5 | - |
| Момент статический РО, кНм | $M_{POCT}$ | 40,1 | | | -40,1 | | |
| Момент инерции РО, $кгм^2$ | $J_{POCT}$ | 0,8 | | | 0,8 | | |
| Приведение к валу двигателя ($J_p = 25$, $\eta_p = 0.95$) | | | | | | | |
| Скорость двигателя, рад/с | $\omega_p$ | - | 250 | - | - | 250 | - |
| Момент статический на валу, кНм: | | | | | | | |
| – без учета потерь в передаче | $M_{пр}(\eta = 1)$ | 1.6 | | | -1,6 | | |
| – с учетом $\eta_p$ | $M_{вс}(\eta \neq 1)$ | 1,68 | | | -1,68 | | |
| – с учетом механических потерь ±ΔМХ двигателя, кНм | $M_c$ | 1,8 | 1,8 | 1,4 | -1,8 | -1,8 | -1,4 |
| Приведенный момент инерции, $кгм^2$ | $J_{ПР}$ | 0,001 | | | 0,001 | | |
| Момент инерции электропривода, $кгм^2$ | $J$ | 0,0019 | | | 0,0019 | | |
| Динамический момент, Нм | $M_{ДИН}$ | 0,475 | - | 0,475 | 0,475 | - | 0,475 |
| Момент двигателя, допускаемый по ускорению, кНм | $M_{ДОПУСК}$ | 2,275 | 2,275 | -0,925 | 2,275 | 2,275 | -0,925 |

Продолжение таблицы 2.6

| Участок движения | | Прямой ход | | | Обратный ход | | |
|---|---|---|---|---|---|---|---|
| Расчетные данные | Обознач. | Пуск | Устан. | Торможен. | Пуск | Устан. | Торможен. |
| Данные предварительного расчета | | | | | | | |
| Момент двигателя средний на участке, кНм | $M_{ср}$ | 2,275 | 2,275 | -0,925 | 2,275 | 2,275 | -0,925 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Время работы, с | $t_в$ | 1 | 9 | 1 | 1 | 9 | 1 |
| Угол поворота вала двигателя, рад | $\alpha_в$ | 125 | 2500 | 125 | 125 | 2500 | 125 |

2.6 Предварительная проверка двигателя по нагреву и производительности

Используя выбранные выше значения пусковых и тормозных моментов, скоростей установившихся режимов и возможности выбранной схемы управления двигателем найдем:

1) время переходного процесса:

-при пуске

$$t_П = J\frac{\omega_Р}{M_{СР1} - M_{ДС1}} = \frac{0,0019 \cdot 250}{2,275 - 1,8} = 1\text{с}; \quad (39)$$

-при торможении

$$t_Т = J\frac{\omega_Р}{M_{СР3} - M_{ТС1}} = \frac{0,0019 \cdot 250}{0,925 - 1,4} = 1\text{с}. \quad (40)$$

2) угол поворота вала двигателя за время переходного процесса:

-при пуске

$$\alpha_П = \frac{t_П \cdot \omega_Р}{2} = \frac{1 \cdot 250}{2} = 125\,\text{рад}; \quad (41)$$

-при торможении

$$\alpha_Т = \frac{t_Т \cdot \omega_Р}{2} = \frac{1 \cdot 250}{2} = 125\,\text{рад}; \quad (42)$$

-установившаяся скорость

$$\alpha_У = \frac{2 \cdot L \cdot j_Р}{D_К} = \frac{2 \cdot 5 \cdot 25}{0,1} = 2500\,\text{рад}; \quad (43)$$

где $\alpha_П$ - угол поворота вала двигателя за время пуска;

$\alpha_Т$ - угол поворота вала двигателя за время торможения;

$\alpha_У$ - угол поворота вала двигателя с установившимся значением скорости.

3) время работы с установившейся скоростью:

-при прямом ходе

$$t_y = \frac{\alpha_y - (\alpha_П + \alpha_T)}{\omega_P} = \frac{2500 - (167{,}5 + 155)}{250} = 9. \quad (44)$$

Время с установившейся скоростью обратного хода равно прямому ходу.

Примем, что момент двигателя за время переходного процесса от начального до конечного значения скорости остается постоянным.

Проверка двигателя по производительности заключается в сравнении суммарного фактического времени работы электропривода в цикле $t_ф$, с заданным значением времени работы $t_р$ в исходных данных для проектирования. Задание по производительности должно быть безусловно выполнено, т.е. $t_Ф \le t_P$.

$$t_Ф = (|t_П| + |t_T| + |t_y|) \cdot 2 = (1 + 1 + 9) \cdot 2 = 22\,\text{с}. \quad (45)$$

Предварительная проверка двигателя по нагреву осуществляется сравнением среднеквадратичного момента с допускаемым моментом двигателя при его работе с фактической продолжительностью включения $ПВ_{ФАКТ}$.

Условие выполнятся: $t_Ф = 22 \le t_P = 22$.

Предварительная проверка двигателя по нагреву осуществляется сравнением среднеквадратичного момента с допускаемым моментом двигателя при его работе с фактической продолжительностью включения $ПВ_{ФАКТ}$.

$$M_{\text{сркв}} = \sqrt{\frac{M_{CP1}{}^2 \cdot t_П + M_{CP1}{}^2 \cdot t_y + M_{CP3}{}^2 \cdot t_T + M_{CP2}{}^2 \cdot t_П + M_{CP2}{}^2 \cdot t_y + M_{CP4}{}^2 \cdot t_T}{t_Ф}}, \quad (46)$$

$$M_{\text{сркв}} = \sqrt{\frac{2{,}275^2 \cdot 1 + 2{,}275^2 \cdot 9 + 0{,}925^2 \cdot 1 + 2{,}275^2 \cdot 1 + 2{,}275^2 \cdot 9 + 0{,}925^2 \cdot 1}{22}} = 2{,}187$$

Момент двигателя при $ПВ_{КАТ}$ , ближайшем к $ПВ_{ФАКТ}$:

$$M_{КАТ} = \frac{P_Н}{\omega_Н} = \frac{750}{296{,}7} = 2{,}52\,\text{Нм}. \quad (47)$$

Продолжительность включения ПВФАКТ:

$$ПВ_Ф = \frac{t_Ф}{t_Ц} \cdot 100 = \frac{22}{360} \cdot 100 = 6{,}1. \quad (48)$$

Момент двигателя, допускаемый по нагреву:

$$M_{ДОП} = M_{КАТ} \cdot \sqrt{\frac{ПВ_{КАТ}}{ПВ_{Ф}}} = 2,53 \cdot \sqrt{\frac{40}{6,1}} = 5,16 \text{ Нм}. \quad (49)$$

Двигатель подходит по условиям нагрева, т. к. $M_{\text{сркв}} = 2,187$ Нм $\leq M_{ДОП} = 5,16$ Нм. Значит можно переходить к дальнейшему расчету.

## 2.7 Расчёт и построение графиков статических характеристик

Синхронная скорость вращения $\omega_{0Н}$:

$$\omega_{ОН} = \frac{2 \cdot \pi \cdot f_{1Н}}{p} = \frac{2 \cdot \pi \cdot 50}{1} = 314,16 \frac{\text{рад}}{\text{с}}. \quad (50)$$

где $p$ – число пар полюсов; $f_{1Н}$ – номинальная частота напряжения статора, Гц.

Номинальный момент на валу $M_Н$:

$$M_Н = \frac{P_Н}{\omega_Н} = \frac{750}{296,86} = 2,53 \text{ Нм}. \quad (51)$$

Номинальный электромагнитный момент:

$$M_{ЭМН} = \frac{3 \cdot U_{1Н} \cdot I_{1Н} \cdot \cos\varphi - 3 \cdot I_{1Н}^2 \cdot r_1}{\omega_{ОН}}, \quad (52)$$

$$M_{ЭМН} = \frac{3 \cdot 380 \cdot 1,8 \cdot 0,71 - 3 \cdot 1,8^2 \cdot 91,9}{314,16} = 1,51 \text{ Нм}.$$

Номинальное относительное скольжение:

$$S_Н = \frac{\omega_{ОН} - \omega_Н}{\omega_{ОН}}, \quad (53)$$

$$S_Н = \frac{314,16 - 296,86}{314,16} = 0,055.$$

Критическое скольжение:

$$S_k = S_Н \cdot (\mu_к + \sqrt{\mu_к^2 - 1}), \quad (54)$$

где $\mu_к$ – перегрузочная способность асинхронного двигателя.

$$\mu_к = \frac{M_К}{M_Н} = \frac{2,8}{2,53} = 1,1. \quad (55)$$

$$S_k = 0,055 \cdot (1,1 + \sqrt{1,1^2 - 1}) = 0,99.$$

Таблица 2.7 – Расчетные параметры для работы в заданных точках

| **Расчетные параметры** | | **Прямой ход** | **Обратный ход** |
|---|---|---|---|
| | | **Заданные точки** | |
| $\omega_{зад}$ | Рад/с | 250 | -250 |
| | О.е. | 0,79 | 0,79 |
| $M_с$ | Нм | 1,8 | 1,8 |
| | О.е. | 0,71 | 0,71 |
| | | **Расчетные данные** | |
| $\omega_{0зад}$ | О.е. | 0,83 | 0,83 |
| | Рад/с | 260,75 | 260,75 |
| $f_1$ | О.е. | 0,83 | 0,83 |
| | Гц | 41,5 | 41,5 |
| $U_1$ | О.е. | 0,83 | 0,83 |
| | В | 182,6 | 182,6 |

Синхронная скорость:

$$\overline{\omega_{0зад}} = \overline{\omega_{зад}} + \overline{\omega_{ест}} = \overline{\omega_{зад}} + \overline{M_{зад}} \cdot S_Н, \quad (56)$$

$\omega_{0зад}$ =0,79+0,71·0,055=0,83,

$\omega_{0зад}$ =0,79+0,71·0,055=0,83.

$$\omega_{0зад} = \omega_{0зад} \cdot \omega_{0Н}, \quad (57)$$

$\omega_{0зад}$ =0,83·314,16=260,75 рад/с,

$$\omega_{0зад} = 0{,}83 \cdot 314{,}16 = 260{,}75 \text{ рад/с}.$$

Частота и напряжение на статоре при U1/f1=const:

$$\alpha = \frac{f_1}{f_{1H}} = \omega_{0зад} = 0{,}83; \quad (58)$$

$$f_1 = \alpha \cdot f_{1H} = 0{,}83 \cdot 50 = 41{,}5 \text{ Гц}; \quad (59)$$

$$U1 = \alpha \cdot U1_H = 0{,}83 \cdot 220 = 182{,}6 \text{ В}. \quad (60)$$

Рисунок 2.5 – Естественная характеристика при номинальных значениях

Рисунок 2.6 – Искусственные характеристики для работы в точке

2.8 Построение временных диаграм переходных процессов

Переходные характеристики строятся с помощью программы zipchad. В случае с перемещением ворот характеристика для прямого и обратного хода будут одинаковыми.

Рисунок 2.7 – Переходные процессы пуска и торможения прям. и обр. ход

2.9 Расчет интегральных показателей

2.9.1 Проверка по нагреву двигателя

Эквивалентный ток:

$$I_Э = \sqrt{\frac{\sum_{i=1}^{n} I_i^2 \cdot \Delta t_i}{\sum_{i=1}^{n} \beta_i \cdot \Delta t_i}} \le I_{доп}, \quad (61)$$

где $I_i$ - среднеквадратичное значение тока на i-ом участке; $\Delta t_i$ - длительность $i$-го участка работы; $\beta_i$ - коэффициент ухудшения теплоотдачи двигателя; $I_{доп}$ - допустимый по нагреву ток.

На рисунке 2.7 приведены значения среднеквадратичного тока $I_{kv}$ за время пуска $t_k$, за время торможения $t_t$ для одного участка движения.

Коэффициент ухудшения теплоотдачи остановленного двигателя $\beta_0$ зависит от его конструктивного исполнения и условий вентиляции. Примерные значения $\beta_0$ приведены в таблице 2.8.

Таблица 2.8 – Значения коэффициента ухудшения теплоотдачи $\beta_0$

| Исполнение двигателя | $\beta_0$ |
|---|---|
| Закрытый с независимой вентиляцией | 1 |
| Закрытый без принудительного охлаждения | От 0,95 до 0,98 |
| Закрытый самовентилируемый | От 0,45 до 0,55 |
| Защищенный самовентилируемый | От 0,25 до 0,35 |

Ухудшение условий охлаждения двигателя в переходных режимах учитывается коэффициентом ухудшения теплоотдачи $\beta_i$, которому, в зависимости от скорости вращения, можно присвоить значения из таблицы 2.9.

Таблица 2.9 – Рекомендуемые значения коэффициента ухудшения условий охлаждения двигателя

| $\omega$ | $\omega \le 0{,}2 \cdot \omega_н$ | $0{,}2 \cdot \omega_н \le \omega \le 0{,}8 \cdot \omega_н$ | $\omega \ge 0{,}8 \cdot \omega_н$ |
|---|---|---|---|
| $\beta_i$ | $\beta_0$ | $(1+\beta_0)/2$ | 1 |

Рассчитаем эквивалентный ток:

$$I_Э = \sqrt{\frac{0{,}8^2 \cdot 1 + 0{,}46^2 \cdot 1 + 0{,}8^2 \cdot 1 + 0{,}46^2 \cdot 1}{1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1}} = 0{,}425 \text{ А}. \quad (62)$$

Допустимый по нагреву ток двигателя:

$$I_{доп} = I_{кат} \cdot \sqrt{\frac{ПВ_{КАТ}}{ПВ_{ФАКТ}}} = 1{,}8 \cdot \sqrt{\frac{40}{6}} = 4{,}6 \text{ А}. \quad (63)$$

$I_Э < (0{,}85…0{,}9)I_{ДОП}$, следовательно, двигатель проходит проверку по нагреву.

2.9.2 Проверка по нагреву и условие выбора преобразователя

Проверка выполняется сравнением среднеквадратичного тока двигателя за время работы с номинальным выходным током преобразователя:

$$I_{сркв} = \sqrt{\frac{1}{t_p} \cdot \sum_{i=1}^{n} I_i^2 \cdot \Delta t_i}. \quad (64)$$

$$I_{сркв} = \sqrt{\frac{0{,}8^2 \cdot 1 + 0{,}46^2 \cdot 1 + 0{,}8^2 \cdot 1 + 0{,}46^2 \cdot 1}{22}} = 0{,}077 \text{ А}. \quad (65)$$

Следовательно, выбираемый преобразователь должен соответствовать условию $I_{сркв} < I_{НПР}$.

[7][8]

# 3 ВЫБОР И ОПИСАНИЕ ЭЛЕМЕНТОВ СХЕМЫ

На основе данных о технологическом процессе можно определить какие элементы автоматизации необходимы для осуществления проекта.

В предыдущих пунктах были выбраны и проверены двигатель и редуктор для перемещения ворот. Далее целесообразно начать подбор элементов с преобразователя частоты.

## 3.1 Выбор преобразователя частоты

Основным параметром для выбора преобразователя является мощность, но также для проекта важно, чтобы выбранный преобразователь имел небольшую цену, т.к будет находится в домашнем, а не промышленном пользовании.

Выбор преобразователя частоты осуществляется, исходя из условия:

$$\begin{cases} U_{ПЧ} \geq U_{НЛ} \\ I_{ПЧ} \geq I_{1Н} \end{cases}, \quad (66)$$

где $U_{ПЧ}$, $I_{ПЧ}$ - номинальное линейное напряжение и ток нагрузки преобразователя частоты;

$U_{НЛ}$, $I_{1Н}$ - номинальное линейное напряжение и фазный ток статора двигателя.

Для выбранного асинхронного двигателя А71А2 при подключении по схеме «треугольник» номинальное линейное напряжение равно напряжению сети 220В, а фазный ток 1,8 А. Исходя из всего был выбран преобразователь бренда CoolClassic модели ZW-AT1-1500W.

Рисунок 3.1 – Внешний вид преобразователя частоты

Таблица 3.1 – Технические характеристики преобразователя

| Входное напряжение | Выходное напряжение | Входная частота | Выходная частота | Фаз на входе | Фаз на выходе | Ток | Мощность адаптера питания |
|---|---|---|---|---|---|---|---|
| 220 В | 220 В | 50/60 Гц | 0-400 Гц | 1/3 | 3 | 8 А | 1,5 кВт |

3.2 Выбор автоматического выключателя

Автоматический выключатель (АВ) должен выбираться по условию:

$$\begin{cases} U_{Н.ВЫКЛ} \geq U_{Н.СЕТИ} \\ I_{Н.ВЫКЛ} \geq I_{1Н} \end{cases}, \quad (67)$$

где $U_{Н.ВЫКЛ}$ - номинальное напряжение АВ;

$U_{Н.СЕТИ}$ - напряжение питающей сети;

$I_{Н.ВЫКЛ}$ - номинальный ток АВ.

Данному условию выбора подходит автоматический выключатель LPN-2C-2, производителя OEZ. Данный выключатель подходит для защиты проводок электрических цепей с оборудованием, которое вызывает импульсы тока (группы ламп, двигатели и т.п.).

Таблица 3.2 – Технический характеристики LPN-2C-2

| Номинальный ток, А | Номинальное напряжение (АС), В | Характеристика отключения | Количество полюсов, шт | Максимальная отключающая способность, кА |
|---|---|---|---|---|
| 2 | 230 | C | 2 | 10 |

## 3.3 Выбор основного контроллера

Для проектируемой системы необходима модульная плата с многокаальным программируемым контроллером, работающим совместно с WiFi контроллером. Таким модулем является Arduino Mega на микроконтроллере ATmega2560, совмещенная с контроллером ESP8266. Данная плата является китайской разработкой компании «RobotDyn». С её помощью будет увязана и централизована вся схема автоматики дома. Благодаря интеграции ESP8266 в плату, появилась возможность управлять платой, а следовательно и всеми контурами подключенными к Arduino посредством команд отправленых по WiFi сети. Arduino Mega - это новое семейство микроконтроллеров Ардуино для решения самых разных задач автоматизации. Плата имеет 54 цифровых входа/выхода, 15 из которых используются как ШИМ, 16 аналоговых входов, 4 аппаратных последовательных входа для реализации последовательных

интерфейсов UART, кварцевый резонатор с частотой 16МГц, USB порт, разъём питания, разъём micro-USB.

Для начала работы с устройством достаточно просто подать питание от AC/DC-адаптера или батарейки, либо подключить его к компьютеру посредством USB-кабеля. Плата имеет компактный размер: длиной 10,2 см и шириной 5,4 см с учетом USB разъёма и разъёма питания, выступающих из корпуса платы. Плата может работать от источника питания в диапазоне от 6 до 20 вольт. При меньшем напряжении плата работает нестабильно. При более высоком напряжении плата начинает нагреваться и может выйти из строя.

Выводы питания Arduino Mega 2560 перечислены ниже:

-VIN. Напряжение, поступающее в Arduino непосредственно от внешнего источника питания (при отсутствии подачи напряжения на USB порт или иного сточника). Через этот вывод можно как подавать внешнее питание, так и потреблять ток, когда устройство запитано от внешнего адаптера.

-5V. На этот вывод поступает напряжение 5В от стабилизатора напряжения на плате, вне независимости от того, как запитано устройство: от адаптера (7 - 12В), от USB (5В) или через вывод VIN (7 - 12В) Максимальный ток, потребляемый от этого вывода, составляет 50 мА.

-GND. Выводы земли.

-IOREF. Этот вывод предоставляет платам расширения информацию о рабочем напряжении микроконтроллера Ардуино. В зависимости от напряжения, считанного с вывода IOREF, плата расширения может переключиться на соответствующий источник питания либо задействовать преобразователи уровней, что позволит ей работать как с 5В, так и с 3.3В устройствами.

Arduino Mega 2560 имеет 225 Кб флэш-память, которая используется для хранения программного кода, 8Кб из которых используются для загрузки и 4 Кб энергонезависимой памяти служащей для работы с библиотекой EEPROM.

Каждый из 54 цифровых пинов на Arduino Mega может работать в режиме входа или выхода, используя функции pinMode, digitalWrite и digitalRead. Выходы работают на 5 В.

Каждый пин может отдать или принять максимум 40 мА и имеет внутренний подтягивающий резистор 20-50 кОм (отключен по умолчанию). Плюс к этому, некоторые выводы имеют специальные функции:

- Последовательный интерфейс Serial: 0 (RX) и 1 (TX); Serial 1: 19 (RX) и 18 (TX); Serial 2: 17 (RX) и 16 (TX); Serial 3: 15 (RX) и 14 (TX). Данные выводы используются для получения (RX) и передачи (TX) данных по последовательному интерфейсу. 0 и 1 подключены к выводам микросхемы CH340G, которая производит преобразование данных USB-UART.

-ШИМ выводы 2-13 и 44-46 с помощью функции analogWrite могут выводить 8-битные аналоговые значения в виде ШИМ -сигнала.

- Интерфейс SPI: выводы 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). Выводы осуществляют передачу связи по интерфейсу SPI, используя библиотеку SPI. Пины SPI могут быть выведены на ISCP, совместимый с Arduino Uno, Duemilanove и Diecimila.

-LED 13. Это встроенный в плату светодиод, который включен в 13 вывод. При подаче сигнала HIGH, светодиод загорается, при подаче сигнала LOW выключается.

-Выводы 20(SDA) и 21(SCL) позволяют осуществлять передачу связи по интерфейсу TWI. 10 В Mega2560 имеются 16 аналоговых входов, каждый из которых можно представить в аналоговое напряжение в виде 10-битного числа.
-Вывод AREF опорного напряжения для аналоговых входов

-Вывод RESET подаёт низкий уровень сигнала для перезагрузки микроконтроллера.

Arduino Mega 2560 предоставляет ряд возможностей для осуществления связи с компьютером, еще одним Ардуино или другими микроконтроллерами. В

ATmega2560 есть четыре аппаратных приёмопередатчика UART для реализации последовательных интерфейсов (с логическим уровнем TTL 5В).

Микроконтроллер CH340G обеспечивает связь одного из приемопередатчиков с USB-портом компьютера, и при подключении к ПК позволяет Ардуино определяться как виртуальный COMпорт.

Особенность выбранной для проекта платы Arduino Mega в том что в неё интегрирован WiFi контроллер ESP8266. Контроллер размещен на той же плате, обеспечив ему своё стабилизированное питание. Программирование микроконтроллера ESP8266, как и основного Atmega2560 возможно через встроенный USB-UART конвертер на базе микросхемы CH340G.

Рисунок 3.2 – Внешний вид контроллера Arduino Mega+WiFi

3.4 Выбор дополнительного контроллера

Для управления откатными воротами необходим дополнительный контроллер, чтобы прописать в нем логику открывания, подключить датчики препятствия и считыватель рфид меток. Также этот контроллер будет взаимодействовать с основным контроллером при помощи радиосвязи. Связь между контроллерами необходима для того, чтобы можно было управлять воротами со смартфона подключившись к интерфейсу, который расположен в главном контроллере.

Для исполнения всех этих задач подойдет плата контроллера Arduino Nano того же производителя, что и предыдущий контроллер.

Платформа Nano, построенная на микроконтроллере ATmega328, имеет маленькие габаритные размеры и может использоваться в местах с ограниченным пространством. Она имеет схожую с Arduino Mega функциональность, однако отличается сборкой и количеством входных и выходных каналов. Плата Arduino Nano может питаться от постоянного

источника питания от 6 до 12 В (30-ый вывод на плате) или от 5 В (27-ой вывод), либо кабеля Micro-USB. Автоматически выбирается источник с самым высоким напряжением.

Микроконтроллер ATmega328 имеет 32 кБ флеш-памяти, 2 кБ ОЗУ и 1 Кб EEPROM. Nano имеет 14 цифровых входов/выходов работа каждого из них задаётся используя функции pinMode(), digitalWrite(), и digitalRead. Цифровые входы/выходы работают при напряжении 5 В. Каждый вывод имеет нагрузочный резистор (стандартно отключен) 20-50 кОм и может пропускать до 40 мА.

Выводы 0 (RX) и 1 (TX) используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы CH340G, которая производит преобразование данных USB-UART. Любой из выводов 3, 5, 6, 9, 10, и 11 обеспечивает ШИМ с разрешением 8 бит при помощи функции analogWrite(). Посредством выводов 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK) осуществляется связь SPI, которая, хотя и поддерживается аппаратной частью, не включена в язык Arduino. Встроенный светодиод LED, подключенный к цифровому выводу 13. На платформе Nano установлены 8 аналоговых входов, каждый разрешением 10 бит (т.е. может принимать 1024 различных значения). Стандартно выводы имеют диапазон измерения до 5 В относительно земли, тем не менее имеется возможность изменить верхний предел посредством функции analogReference(). С помощью выводов A4 (SDA) и A5 (SCL) осуществляется связь I2C (TWI). Для создания используется библиотека Wire. Более подробная информация находится в документации.

Рисунок 3.3 – Внешний вид платы Arduino Nano с двух сторон

## 3.5 Устройство связи

Для связи между двумя контроллерами будет использован радиомодуль NRF24L01. Радиосвязь избавит нас от прокладки сигнального кабеля от дома до ворот. А связь между контроллерами обеспечит управление воротами с дома и улицы от команд смартфона.

Размеры платы - 14,5 на 30 мм. Модуль работает на частоте 2,4 гГц и поддерживает скорость передачи данных до 2 Мбит/с. Модуль работает, как в режиме приемника, так и в режиме передатчика. Для наших условий будем использовать два модуля: один – приёмник (подключен к контроллеру ворот), другой – передатчик (подключен к контроллеру дома). Есть два исполнения данных модулей: с усилителем сигнала и без.

Без усилителя радиус действия устройств около 100 м, а при использовании усилителей с дополнительными антенами от 1,5 до 2 км. Плата подключается и работает с контроллером по интерфейсу SPI. Выводы платы поддерживают сигнал 5В. Назначение выводов:

SCK (Serial ClocK) — тактирование (синхронизация);

MOSI / MI (Master Out Slave In) — вход данных;

MISO / MO (Master In Slave Out) — выход данных;

CE/SS - Выбор ведомого на шине SPI из нескольких устройств;

SCN - выбор режима приема/передача , фактически тот же CE;

IRQ - выход прерывания,чаще всего не используется. Необходим для немедленной реакции микроконтроллера при приеме нового пакета данных.

GND - земля,масса,минусовая шина;

Vcc -питание модуля.

Питание платы – 3,3 В, но мощности выхода Arduino 3,3 В не хватает, поэтому необходим другой источник стабилизированного питания. Такие источники есть в готовом виде, которые сконструированы специально для данных модулей и удобно подключаются друг к другу с помощью контактной гребёнки.

Рисунок 3.4 – Внешний вид устройства NRF24L01 без усилителя сигнала

## 3.6 Выбор необходимых датчиков

### 3.6.1 Датчик положения ворот.

В качестве датчика положения ворот подойдет любое герконовое магнитоуправляемое реле. Такие датчики не требуют какого-либо обслуживания, герметичны и очень надежны. При помощи контроллера предоставляется возможность отслеживать положение ворот по одному геркону. Положение ворот будет обновляется каждый раз при приближении к нему любого из трех магнитов, установленных на воротах. Сам датчик будет находиться в неподвижном состоянии.

### 3.6.2 Датчик препятствия.

Для ворот в целях безопасности необходимы датчики препятствия, потому как после подачи сигнала на закрытие ворот или во время действия рфид метки – ворота будут автоматически закрываться, и если на пути движения возникнет какой-либо объект, ворота следует немедленно остановить или реверсировать.

Для такой роли подойдут специализированные датчики для ворот. Для проекта был выбран датчик производителя RYTERNA.

Устройство имеет приёмник и излучатель инфракрасного излучения. Ширина проёма, который может контролировать сенсор до 15 м. Питание датчика от 12-24 В. Выходом устройства является контакт реле.

Рисунок 3.5 – Схема подключения датчика препятствия

3.6.3 RFID считыватель

Напомним, что считыватель RFID меток необходим для идентификации владельца дома по активной RFID метке. Такая метка имеет источник питания, поэтому может действовать на расстоянии от 5 до 20 м. По задумке индивидуальный код активной метки считывается и добавляется в сформированный список контроллера, затем при использовании метки, когда она находится в зоне считывания, автоматические ворота открываются, а когда метка пропадет – закрываются. Данную метку можно положить в автомобиль или носить с собой в кармане или сумке. Так как передаваемый код меток индивидуален, то становится возможным создать условие, при котором контроллер сможет распознавать человека и автомобиль как разные объекты. Для автомобиля откатные ворота будут открываться на всю ширину проёма, а для человека на небольшое расстояние, имитируя калитку.

Устройство, которое может прочитать метку и передавать информацию в контроллер – модуль MFRC522, основанный на одноименной микросхеме. Данный модуль также имеется у производителя Robotdyn. Для связи с микроконтроллером модуль использует протокол SPI.

Считывание меток происходит при помощи радиосвязи частотой 13,56 МГц. Напряжение питания модуля 3,3 В. Скорость передачи данных 10 МБит/с.

Рисунок 3.6 – Внешний вид RFID считывателя

### 3.6.4 Датчики обнаружения движения в помещении

Датчики наличия движения будут формировать сигнал для охранной системы. Для проекта не подойдут датчики с механической коммутацией контактов. Реле будет постоянно издавать шум и потреблять значительную мощность, так как имеет катушку-электромагнит. Необходимо подобрать датчик с цифровым выходом, коммутируемый полупроводниковой схемой.

Выбран наиболее подходящий датчик производителя Robotdyn PIR Motion Sensor. Модуль предназначен для фиксирования движения объектов, излучающих тепло, поэтому его работа не зависит от освещения. Чтобы увеличить угол срабатывания, на чувствительный элемент датчика установлена линза Френеля, которая собирает и фокусирует излучения. С линзой рабочий угол составляет 110 градусов, а максимальное расстояние до источника излучения 8 м. Устройство имеет цифровой выход, который обретает значения логического нуля и единицы.

Большим достоинством является очень низкий ток потребление – 50 мкА. Для настройки чувствительности и длительности удержания выходного сигнала на плате имеются два переменных резистора.

Для подключения датчика к контроллеру достаточно на контакт VCC подать «+», на GND - «-», а OUT подключить к цифровому или аналоговому входу.

Рисунок 3.7 – Внешний вид датчика движения

### 3.6.5 Сенсорный датчик

Датчик используется для подачи логической единицы на вход контроллера в тот момент, когда к сенсорной панели кожным покровом руки прикоснётся человек. Человеческое тело становится равносильно присоединению антенны к чувствительному входу усилителя. Из каталога того же производителя был

выбран модуль сенсорного датчика, соответствующий требованиям проекта. А именно: датчик имеет малые габариты, и на выходе датчика формируется уровень логического сигнала 5В, что соответствует входному сигналу контроллера.

Рисунок 3.8 – Внешний вид сенсорного датчика

## 3.7 GSM модуль

GSM модуль в проекте необходим для реализации оповещения владельца дома о срабатывании охранной сигнализации в форме смс или звонка на мобильный телефон.

Для проекта подойдет мудуль SIM800L. Очень компактный и многофункциональный модуль, который имеет серийный порт для непосредственной связи с микроконтроллером. Напряжение питания модуля 3,3-4,2 В. Контакты питания (GND, VCC) и контакты для связи с контроллером (RX, TX) выведены по бокам платы. Дополнительно имеются контакты для перезагрузки модуля, вывод индикотора звонка, вывод для динамика и микрофона. Потребляемый ток в режиме ожидания 0,7 мА, а в пиковом режиме 2А. Для работы модуля нужна сим карта, которая вставляется в имеющийся на плате слот.

Рисунок 3.9 – Внешний вид модуля SIM800L

Для стабильной работы модуля SIM800L необходим источник питания с рабочим током 2А, для этого подойдет преобразователь напряжения LM2596 с предварительно настроенным выходным напряжением 4.3 В.

## 3.8 Музыкальный модуль для сигнализации

Модуль WTV020-SD необходим для реализации звуковой сигнализации и оповещений. Модуль может воспроизводить файлы формата Wav и ad4 с флеш накопителя, для которого на плате есть слот. Управление модулем возможно при помощи контроллера. Протокол общения – I2C. Напряжение питания 3,3 В. Плата имеет очень маленькие габаритные размеры. Благодаря тому, что считывание аудиофайлов происходит со съемного флеш накопителя, легко можно перезаписывать звуки на новые ничего не переподключая и не меняя в прошивке. Динамик сопротивлением от 8 до 16 Ом напрямую подключается к устройству. Для воспроизведения звуков на более мощных динамиках, к устройству может быть подключен усилитель.

Рисунок 3.10 – Модуль WTV020-SD

## 3.9 Выбор диммеров для регулирования яркости освещения

Регулируемая яркость освещения будет реализована при помощи ШИМ сигнала с вывода микроконтроллера. Для коммутации силовой части необходим полупроводниковый прибор типа симистор с подходящими параметрами по

току, напряжению и скорости открытия/закрытия полупроводникового перехода.

Из каталога производителя Robotdyn была выбрана модульная четырехканальная плата диммера, на которой расположены четыре симистора BTA-16-600B, RC-фильтры для каждого канала и четыре оптрона для защиты выхода котроллера и для определения нулевой точки. Для подключения нагрузки имеются клемные зажимы.

При подаче высокого уровня сигнала с микроконтроллера на управляющий контакт модуля, симистор открывается и находится в открытом состоянии, пока не будет снято управляющее воздействие. После снятия сигнала управления симитор самостоятельно закрывается, когда синусоидальная характеристика сетевого напряжения проходит через нулевую точку и ток прекращается.

Таблица 3.3 – Технические характеристики симистора

| | |
|---|---|
| Максимальное обратное напряжение, Uобр., В | 600 |
| Макс. импульсное напр. в закрытом состоянии Uзс.повт.макс.,В | 600 |
| Макс. среднее за период значение тока в открытом состоянии Iос.ср.макс.,А | 16 |
| Макс. кратковременный импульсный ток в открытом состоянии Iкр.макс.,А | 168 |
| Макс. напр. в открытом состоянии Uос.макс.,В | 1,5 |
| Наименьший постоянный ток управления, необходимый для включения тиристора Iу.от.мин.,А | 0,05 |
| Отпирающее напряжение управления,соответствующее минимальному постоянному отпирающему току Uу.от.,В | 1,3 |
| Время включения tвкл.,мкс | 2 |

Рисунок 3.11 – Внешний вид диммера

3.10 Выбор электронного реле

Для нерегулируемых каналов нагрузки тоже необходим коммутирующий прибор. В основе модуля лежит всё тот же симистр BTA-16-600B, также на плате имеется сопротивление в качестве фильтра и оптоизолятор в качестве детектора нуля и защиты выхода котроллера.

Рисунок 3.12 – Внешний вид модуля

## 3.11 Драйвер для управления приводом жалюзи

Привод жалюзи выполнен для питания постоянным током, поэтому для управления двигателем необходимо подобрать плату драйвера. Исходя из того, что напряжение питания двигателя 12 В, напряжение с контроллера 5 В и мощность мотора 11 Вт, была выбрана модульная плата L298N. На ней имеется одноименная микросхема, внутри которой выполнены два «H» моста. Также на плате есть схема стабилизации напряжения 5 В, клемные зажимы (для подключения напряжения питания двигателей, питания микросхемы и управлющих сигналов) и диодные группы для защиты микросхемы.

Рисунок 3.13 – Модуль драйвера для управления двигателем

Таблица 3.4 – Технические характеристики L298N

| Напряжение питания логики | Полярность | Напряжение сток-исток | Сопротивление открытого канала | Пороговое напряжение на затворе | Максимальный ток сток-исток |
|---|---|---|---|---|---|
| 5 В | N | 55 В | 8 мОм | 2-4 В | 110 А |

## 3.12 Выбор блоков питания

Для выбранного устройства SIM800L необходим преобразователь напряжения, способный питать нагрузку током 2 А и напряжением 4,2 В. Таким преобразователем является преобразователь LM2596.

Чтобы выбрать блок питания на напряжение 12 В для системы внутри дома, необходимо знать мощность главного контроллера и суммарный ток потребления всех подобранных устройств для проектируемой схемы.

А для того, чтобы выбрать блок питания на напряжение 12 В для системы вне дома, необходимо знать суммарный ток потребления контроллера ворот, датчиков и RFID считывателя.

Количество некоторых выбранных устройств, таких как симисторные модули, транзисторные схемы для жалюзи с приводом, зависит от требования количества каналов управления для них.

Для выбора блоков питания возьмем максимальное количество каналов с запасом: 20 – для освещения и розеток, 10 – для привода жалюзи.

Таблица 3.5 – Перечень устройств и токов

| Наименование | Ток потребления, А |
|---|---|
| В доме: | |
| Преобразователь LM2596 и модуль SIM800L | 2 |
| RFID модуль MFRC522 | 0,026 |
| Модуль радиосвязи NRF24L01 | 0,8 |
| Плата контроллера ATmega2560 | 0,5 |
| Симисторные модули для диммирования света | 0,1 |
| Симисторные модули для розеток | 0,1 |
| Драйвер L298N | 0,036 |
| Приводы жалюзи | 9 |
| Датчики движения | 0,0005 |

Продолжение таблицы 3.5

| | |
|---|---|
| Датчики прикосновения | пренебречь |
| Итого: | 12,5 |
| Вне дома: | |
| Плата контроллера Atmega328 | 0,25 |
| RFID модуль MFRC522 | 0,026 |
| Модуль радиосвязи NRF24L01 | 0,8 |
| Датчик препятствия | 0,045 |
| Итого: | 1,121 |

Ток на выходе преобразователя питания должен быть больше, чем суммарный ток из таблицы 0. В соответствии с этим условием были выбраны два импульсных блока питания открытого исполнения китайского производства, фирмы MJJC. Для схемы в доме выходной ток выбранного преобразователя составит – 15А, а для схемы ворот – 2 А. [8][9]

## 4 РАЗРАБОТКА АЛГОРИТМА РАБОТЫ СИСТЕМЫ

Прежде чем приступить к разработке схем и написанию кода программы для контроллеров, необходимо описать подробный алгоритм работы системы.

Алгоритм системы будет выполнен в виде блок-схемы. Блок-схема – это способ графического изображения последовательного алгоритма какого-либо процесса, программы и т.д. Блок-схемы представлены на рисунках 4.1 и 4.2.

Для удобства описания алгоритма обозначим входные и выходные параметры.

Таблица 4.1 – Входные и выходные параметры системы

| **Тип** | **Название параметра** |
|---|---|
| Состояния | Охранная система активирована |
| | Охранная система деактивирована |
| | Ворота открыты |
| | Ворота закрыты |
| | Среднее положение ворот |
| | Ворота открываются |
| | Ворота закрываются |
| | Канал включен |
| | Канал выключен |
| | Таймер присутствует |
| | Таймер отсутствует |
| Входные | Сигнал RFID метки для охранной системы |
| | Сигнал RFID метки для ворот |
| | Датчик препятствия ворот |
| | Датчик движения |
| | Датчик прикосновения |
| | Время отключения |

Продолжение таблицы 4.1

| Тип | Название параметра |
|---|---|
| Входные | Время включения |
| | Пропало основное напряжение |
| Выходные | Сигнал на открытие ворот |
| | Сигнал на закрытие ворот |
| | Сигнал на останов ворот |
| | Включение канала с нагрузкой |
| | Выключение канала |
| | Регулирование канала с нагрузкой |
| | Включить сигнализацию |
| | Выключить сигнализацию |
| | Отправить сообщение |
| | Активировать охранную систему |
| | Деактивировать охранную систему |
| | Сигнал на открытие жалюзи |
| | Сигнал на закрытие жалюзи |
| | Остановить жалюзи |
| Входные команды | Команда «Открыть» |
| | Команда «Закрыть» |
| | Команда «Включить» |
| | Команда «Выключить» |
| | Команда «Установить яркость» |
| | Команда «Установить таймер на включение» |
| | Команда «Установить таймер на выключение» |

Рисунок – 4.1 Блок-схема часть 1

Рисунок 4.2 – Блок-схема часть 2

Всем событиям проекта, которые описаны блок-схемами на рисунках 4.1 и 4.2 присвоены номера. Далее опишем словесно каждое событие:

Событие 1 – происходит считывание RFID метки, если её уникальный ID совпадает с базой, значит она верная. Далее идет условие: если охранная система активна, то деактивировать её и отключить сигнализацию, иначе наоборот активировать.

Событие 2 – При поступлении сигнала с датчика движения проверяется активность охранной системы. Если система активна, то включается звуковая сигнализация и отправляется тревожное сообщение.

Событие 3 – Когда поступает команда «Включить» для канала охранной системы, то система активируется.

Событие 4 – Когда поступает команда «Отключить» для канала охранной системы, то система деактивируется.

Событие 5 – Когда контроллер ворот считал RFID метку, проверяется совпадение с базой, если совпадает, то проверяем закрыты ли ворота и если да, то подается сигнал на открытие ворот.

Событие 6 – Если ворота не закрыты и прошло 40 секунд с момента открытия, то подается сигнал на закрытие ворот.

Событие 7 – Данное событие подает сигнал на останов, когада достигнуто нужное положение ворот.

Событие 8 – Когда поступает команда «Открыть» для канала ворот, то подается сигнал на открытие ворот.

Событие 9 – Когда поступает команда «Закрыть» для канала ворот, то подается сигнал на закрытие ворот.

Событие 10 – Если поступил сигнал от датчика препятствия, то подается сигнал на открытие ворот.

Событие 11 – Когда поступает команда «Включить» для канала управления нагрузкой, то подается сигнал на включение нагрузки.

Событие 12 – Когда поступает команда «Отключить» для канала управления нагрузкой, то подается сигнал на отключение нагрузки.

Событие 13 – Когда поступает команда «Установить таймер на включение» для канала, то записывается уставка.

Событие 14 – Когда поступает команда «Установить таймер на выключение» для канала, то записывается уставка.

Событие 15 – Когда поступает команда «Регулировать нагрузку» для канала, то подаётся сигнал регулирования.

Событие 16 – Проверка если канал включен и присутствует таймер и время истекло, то выключить нагрузку канала и обнулить уставку. Иначе если канал выключен и присутствует таймер и время истекло, то включить канал с нагрузкой.

Событие 17 – Если канал управления жалюзи включен, то запомнить время и подать сигнал на открытие, иначе если выключен, то запомнить время и подать сигнал на закрытие.

Событие 18 – Если время движения жалюзи больше 38 секунд, то подаем сигнал на останов жалюзи.

Событие 19 – Когда поступает команда «Открыть» для канала жалюзи, то включается канал управления жалюзи.

Событие 20 – Когда поступает команда «Закрыть» для канала жалюзи, то выключается канал управления жалюзи.

Событие 21 – Если пропало основное напряжение питания, то сохранить все переменные в энергонезависимую память.

# 5 РАЗРАБОТКА ПРИНЦИПИАЛЬНОЙ ЭЛЕКТРИЧЕСКОЙ СХЕМЫ

Принципиальная схема является основной документацией проекта. Данный документ оформляется согласно ГОСТ 2.702-2011. На этой схеме подробным образом показаны электрические соединения всех составляющих. Электрические приборы и модули изображаются в виде прямоугольников с присвоенными им соответствующими номером позиции и буквой. Буква определяет тип электрического устройства. Внутри этих прямоугольников изображаются контактные группы, клемники и т.д в виде таблицы. В таблице указывается позиция и цепь, к которой ведет данный контакт. Кнопки, лампочки, резисторы и т.д. тоже имеют свои условные графические и буквенные обозначения.

В проекте было разработано две принципиальные схемы: схема автоматизации откатных ворот вне дома и схема автоматизации системы внутри дома.

## 5.1 Принципиальная схема «ворота»

На данной схеме изображены: преобразователь напряжения (поз. U1), частотный преобразователь (поз. U2), плата контроллера (поз. A1), передатчик датчика препятствия (поз. A4), приемник датчика препятствия (поз. A5), устройство радиосвязи (поз. A2), RFID считыватель (поз. A3) и герконовое реле (поз. S1). Сетевое питание переменного напряжения 220 В подключено на вход преобразователя U1, а с выхода этого преобразователя напряжение 12 В поступает на контакты питания 29, 30 контроллера A1. С контроллера питание 12 В расходится на приемный и передающий модули датчика препятствия A4, A5. Сетевое однофазное питание также подается на преобразователь частоты U2. На выходе U2 получаем преобразованное трехфазное напряжение 220 В.

Для питания электронной схемы самого преобразователя частоты, в нем имеется своё понижающее и стабилизирующее устройство. К трехфазному выходу U2 подключен трехфазный асинхронный электродвигатель, обмотки которого соединены треугольником.

Модуль радиосвязи (A2) и считыватель меток (A3) работают по последовательному переферийному интерфейсу (SPI). Это последовательный синхроный протокол передачи данных, используемый микроконтроллерами для обмена данными с одним или несколькими периферийными устройствами на небольших растояниях. Для организации соединения и передачи данных в схеме необходимо одно ведущее устройство, обычно это микроконтроллер и одно или несколько ведомых устройств. Подключение осуществляется по четырем проводам. Модули подключены к контроллеру на соответствующие правилу их подключения контакты. Данные устройства имеют три линии данных, которые подключаются параллельно MISO, MOSI, SCK и линия для выбора устройства SS. Питание для модулей 3,3 В следует с 17-го контакта контроллера A1.

К контроллеру подключены входные сигналы с датчика положения и препятствия. Для надежной работы входа контроллера, необходимо чётко определить уровень входного сигнала, поэтому вход подтягивается резистором к плюсу питания, либо к «земле». В нашем случае используются внутренние подтягивающие резисторы контроллера, которые подтягивают его вход к «земле». Значит для подачи сигнала необходимо использовать контакт GND. На схеме данный контакт под номером 29 с контроллера поступает на контакт 2 приёмника A5 и в случае замыкания контакта реле приёмника коммутируется на контакт 3 (OUT) и поступает на 5-ый цифровой вход контроллера A1(контакт 8). С того же контакта 29(GND) линия следует на герконовое реле S1, где при замыкании его контакта поступает на 2-ой цифровой вход контроллера A1(контакт 5).

Цифровые выходы контроллера A1 3 и 4 (контакты 6 и 7) подключены к цифровым входам 5 и 6 преобразователя U2 (контакты 5 и 6), а так как преобразователь и контроллер имеют общий контакт GND, то подавая сигнал с контроллера входная цепь преобразователя замыкается и срабатывает соответствующая команда. Цифровые входы 5 и 6 преобразователя U2 отвечают за прямое и обратное вращение двигателя соответственно.

## 5.2 Принципиальная схема «внутри дома»

Сетевое питание 220 В для питания данной схемы берется с уже существующего домового выключателя (не выбирался) и поступает на вход преобразователя напряжения U1, а с него выходит напряжение 12 В. Данная схема содержит контур для бесперебойного питания контроллера от химического источника питания (аккумуляторной батареи GB1). Для этого используем самую простую схему, в которую включены два диода VD1, 2. Диод VD2 в нормальном режиме работы заперт более высоким потенциалом основного источника питания. Когда пропадают 12 В от основного источника, диод VD2 отпирается и начинает питать нагрузку от батареи GB1. Диод VD1 необходим в случае, если по каким-либо причинам напряжение основного источника опустилось ниже напряжения батареи. Тогда диод VD1 запирается, и схема начинает получать питание от батареи. Без диода VD1 батарея будет разряжаться на преобразователь напряжения. Для контроля напряжения 12 В в цепь включено реле KL1, которое подаёт сигнал в контроллер в случае перехода на резервный источник питания. Работа от дополнительного источника питания не рассчитана на длительный срок, а необходима, чтобы сохранить текущие конфигурации системы и успеть оповестить владельца о проникновении в дом, если сеть обесточена умышленно и сработала сигнализация.

С преобразователя напряжения U1 напряжение 12 В поступает на преобразователь для модуля связи U2, с которого выходит пониженное напряжение 4,2 В.

Также 12 В подаётся на питание главного контроллера A1, с помощью разъемного соединения XS1,2 на контакты 1, 2 и на питание схемы драйвера реверсивного пуска жалюзи по проводам 1, 2.

Модуль GSM связи SIM800L (позиция A5) получает 4,2 В с преобразователя U2 на контакты 1, 3. С модуля A5 по двум проводам линия связи TXD, RXD (контакты 4,5) следуют на вход серийного порта контроллера A1 на соответствующие контакты группы XP1.3 под номерами 2, 3. Вывод RX нельзя подключать напрямую, так как цифровой вывод контроллера использует 5В, а модуль SIM800L использует 3,3В. Необходимо сигнал RX, поступающий от A1, понизить до 3,3 В, чтобы не повредить модуль SIM800L. Для этого воспользуемся делителем напряжения на резисторах. Подключаем резистор R2 на 10 кОм между выводом RX (SIM800L) и выводом RX (Arduino) и другой резистор R1 на 10 кОм между выводом RX (SIM800L) и GND.

Далее рассмотрим схему реверсивного пуска электропривода жалюзи. Питание для двигателя 12 В и GND по проводам 1, 2 поступают на драйвер L298N (поз. A15) с A1. Для питания логики необходимо 5 В, которые поступают с A1 по проводу 7. Управляющие сигналы с контактов 26, 27 группы контактов XP1.3 платы A1, подаются на входы IN1, 2 A15 (контакты 1,2 группы XP15.1). С выходов OUT1,2 A15 (контакты 4, 5 группы XT15.1) следует питание на двигатель по проводам 37, 38.

Модуль радиосвязи (A12) и считыватель меток (A13), как и в случае с первой схемой подключаются и работают по протоколу SPI. На соответствующие контакты подключаются линии данных MISO, MOSI, SCK. С контактов 10, 9 (цифровых выводов 24, 23) группы XP15.3 контроллера A1 по проводам с номерами 15, 21 подается сигнал для сброса данных на модуль связи A12 и считыватель A13 соответственно.

С контактов 7, 6 (цифровых выводов 21, 20) группы XP15.3 контроллера A1 подается сигнал выбора устройства по проводам с номерами 19, 20 на модуль связи A12 и считыватель A13 соответственно. Питание VCC 3,3 V поступает на модули с 3-го контакта XP15.1 контроллера A1 по проводу 14.

На цифровые входы 35, 36, 37 (группа XP15.3 контакты 13, 14, 15) контроллера A1 подаются сигналы с выходного контакта 3 (OUT) датчиков прикосновения A9-A11. Питание сенсорных датчиков обеспечивает выход 5 В (XP15.1, контакт 4) A1, который размножается от датчика к датчику, включая датчики движения A6-A8 и присоединяется к контактам 1, 2.

Датчики движения A6-A8 имеют цифровой выход с контакта 3 (OUT), которые перемыкаются друг с другом, а к ним подключен цифровой вход 20 (XP15.3 контакт 3). Таким образом, при срабатывании любого из датчиков будет получен сигнал на включение сигнализации.

К цифровым выходам контроллера A1 с 2-9 (XP15.5 контакты 2-9) присоединены устройства A2-A4 для управления нагрузкой, питающейся от сети 220 В (розетки, осветительные приборы). Устройство A2 является диммером для управления нагрузкой и имеет 0-ой и 1-ый контакты для питания (VCC, GND) и контакты 2-5 (COM1-COM4) для логического сигнала с контроллера. A3 и A4 предназначены для нерегулируемой нагрузки и также имеют контакты 0 и 1 (VCC, GND) для питания, контакты 2, 3 (COM1, COM2) для сигнала. Нагрузка 220 В подключается к клемному зажиму XT этих устройств.

В разработанной схеме таких элементов, как каналов для подключения нагрузки, датчиков прикосновения, датчиков движения, схем для управления жалюзи может быть как меньше, так и больше, чем изображено на схеме. Элементы могут быть добавлены по такому же принципу соединения, а в программе меняются соответствующие константы.

# 6 ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ

## 6.1 Принцип работы устройств и среда разработки

Большая часть проекта приходится на программирование микроконтроллеров. Пользуясь упрощенным для разработчиков языком, пишется код, который в конечном счете становится готовой программой. Сама программа, используя ресурс контроллера, сможет выполнять необходимые для проекта задачи.

Прежде чем представить программный код, для понимания общей картины имеет смысл описать главные принципы взаимодействия пользователя с системой, а также самих контроллеров друг с другом.

WiFi контроллер ESP8266 подключается к домашнему WiFi маршрутизатору. Маршрутизатор присваивает локальный IP адрес для обращения к устройству. Чтобы обратится к устройству из внешней сети в маршрутизаторе настраивается «проброс» одного порта, который перенаправляет запросы на локальный IP адрес.

Рисунок 6.1 – Пример настройки «проброса» порта

Затем зная этот порт и IP адрес, который выделил провайдер для внешней сети, мы можем отправлять запросы и получать ответы от WiFi контроллера ESP8266, на котором записан программный код.

Программный код выполняет функции веб-сервера. Чтобы устройство было доступно по одному и тому же постоянному адресу, даже при перезапуске маршрутизатора, контроллер при первом запуске отправляет IP (который выдал провайдер интернета для доступа во внешнюю сеть) на сервер службы, которая его запоминает и выдаёт его же в ответе уже по постоянному адресу, доменному

имени. Иными словами: переходя по постоянной, известной ссылке, получаем в ответ обновленный, изменяющийся IP адрес устройства.

Для подключения к системе «Умный дом», необходимо в строке браузера любого устройства с доступом в интернет вписать IP адрес домашней сети или фиксированное доменное имя, добавить к нему через двоеточие номер порта, по которому после подключения к веб-серверу устройства через браузер, пользователю открывается визуальный интерфейс веб-сервера, где располагаются «инструменты» для управления «Умным домом», а именно: диалоговые окна, вкладки, кнопки, ползунки и т. д. Воздействуя на эти элементы пользователь отправляет запрос по стандартным протоколам, сервер обрабатывает его и формирует ответ пользователю. В то же время программный код WiFi контроллера, на котором работает сервер отправляет команду на исполнение другому контроллеру ATmega2560. А происходит это следующим образом: контроллеры связаны при помощи последовательного серийного порта и могут отправлять и считывать разные символы и строки данных; контроллер ESP8266 отправляет команду в общий серийный порт, ATmega расшифровывает эту команду и отправляет свою о том, что прошло успешное чтение команды и было выполнено соответствующее ей действие. Если к примеру поступила команда от пользователя на включение какого-либо канала, ESP отправляет команду ATmeg’e в порт, ATmega читает её, включается соответствующий канал и отправляется ответ.

Такая связь обеспечивает расширение числа управляемых нагрузок и устройств, потому как сама ESP имеет недостаточное количество входов/выходов.

Среда разработки программного обеспечения для используемых плат контроллеров – компьютерная программа Arduino IDE. Она была разработана для оригинальных продуктов под названием Arduino. Для написания программы контроллера платы используется язык программирования – C++. Кроме редактирования основного файла программы имеется возможность подключать дополнительные файлы и библиотеки, функции которых будут доступны в

основном файле программы. В Arduino IDE присутствует компилятор, который преобразует написанный языком C++ код в машинный код из единиц и нулей готовый к «прошивке», записи в память контроллера. Также в ней имеется терминал для общения и функция записи «прошивки» контроллера, используя последовательный интерфейс UART.

В проекте три программируемых микроконтроллера (ESP8266, ATmega2560, ATmega328), значит необходимо написать три соответствующих программы.

На основании описанных выше алгоритма работы и принципиальной электрической схемы были разработаны программы.

## 6.2 Описание кода программы для ESP8266

Код программы для ESP8266, представленный в приложении А начинается с подключения к главному файлу файлов библиотек. Библиотека «ESP8266WiFi» необходима для использования таких функций контроллера как подключение к сети или создание WiFi точки, для задания настроек режима работы и вывода текущей информации о подключении и состоянии сети. Библиотека «WiFiClient» нужна для работы с удаленными хостами, отправки и получения данных. При помощи библиотеки «ESP8266WebServer» контроллер может принимать запросы и отправлять ответы в виде набора параметров и страниц клиенту по протоколу HTTP ( англ. HyperText Transfer Protocol).

«ESP8266mDNS» - эта библиотека позволяет программе отвечать на многоадресные DNS - запросы с доменных имен.

Также подключены библиотека для работы с энергонезависимой памятью контроллера «EEPROM» и «FS» для работы с файлами.

Далее создаются константы и необходимые переменные, а также объекты. В языке программирования объект – это переменная типа «класс». Класс описывает данные и методы (функции), которые будут использоваться объектом

этого класса. Главной, является константа, которая определяет количество каналов управления. На её значение опираются все фунции программы.

В функции «setup», которая выполняется контроллером только один раз при включении, помещены инициализация входов контроллера, устройств связи, вызов функции синхронизации данных из энергонезависимой памяти, генерация уникального ключа для автоматической авторизации. Сюда же помещены методы отправки соответствующих ответов клиенту, когда тот запросит адрес или файл. Добавлены функция восстановления последнего состояния каналов, после потери питания или перезагрузки и функция отправки IP адреса на сервер.

Функция «loop» выполняется непрерывно. В ней опрашивается наличие запроса от пользователя, добавлено условие, что когда нет высокого уровня сигнала на 5-ом пине контроллера, то нужно сохранить все данные в «EEPROM». Далее прописан цикл «while», где опрашивается присутствие в серийном порте устройства каких-либо данных, в нашем случае символов (команд). Разбирая значение команды функцией «pars» контроллер считывает ответ от другого контроллера ATmega2560 и записывает его значение в свою память. Так происходит обратная связь с контроллером ATmega2560. В функции «handleAction», которая также находится в «loop» разбирается запрос от клиента по аргументам. Сперва проверяется наличие в запросе логина и пароля. Если их нет, то в ответ клиенту отправляется код ошибки, а если есть, то происходит сравнение с записанными в переменные контроллера логином и паролем. При совпадении логина и пароля в ответ отправляется ключ HASH и код «успешно».

В этой же функции происходит поиск других аргументов и их значений, которые определяют какое действие запросил пользователь и отправляют контроллеру ATmega2560 команду на выполнение этих действий (включить, отключить, регулировать, установить таймер и .т.д.).

Более подробно код программы описан комментариями в самом листинге. Комментарий начинается с символов «//».

Также для работы интерфейса вебсервера контроллера ESP8266 необходимо написать код страницы с функциями (скриптами). Основная задача – создать визуальную оболочку для взаимодействия с пользователем интерфейса. Вся страница со скриптами имеют форму файлов с расширениями .html, .js и т.д. Эти файлы при помощи плагина для Arduino IDE загружаются в SPIFFS память контроллера. И при помощи ранее упомянутой библиотеки «FS» эти файлы можно прочитать и выдать клиенту.

## 12.3 Описание кода программы для ATmega2560

Код для данного устройства в приложении Б также начинается с подключения к главному файлу файлов библиотек. «SPI.h» - библиотека для работы с устройствами, которые передают данные по последовательному периферийному интерфейсу. «MFRC522.h» - библиотека для работы модуля считывания RFID меток. «nRF24L01.h», «RF24.h» - для модуля радиосвязи. «Wtv020sd16p.h» - для модуля воспроизведения аудиофайлов.

Затем задаются константы, чтобы было удобней работать с библиотеками и константы для настройки каналов управления. Тут также присутствует основная константа, определяющая количество каналов «MAX_CHANNELS», значение которой должно совпадать с константой в коде ESP8266.

В функции «setup» выполняется инициализация устройств, портов ввода/вывода и последовательных интерфейсов.

Функция «loop» начинается с задания цикла, который считывает символы (команды), и если есть совпадение набора символов с определенной командой, происходит расшифровка и выполнение соответствующих действий на которые указывает эта команда. То есть эта часть кода аналогична той, что прописана в ESP8266, и таким образом два контроллера могут обмениваться командами.

Далее прописано условие, что если массив для передачи данных через радиосвязь отличается от состояния пина определяющего работу ворот, то

присваиваем ему текущее значение и отправляем по радиосвязи в контроллер ворот ATmega328.

Затем прописаны условия, определяющие алгоритм работы жалюзи и остановку их по истечении времени, условия для работы сенсорных выключателей.

В нижней части бесконечного цикла «loop» идет вызов таких функций: «timer» - в ней осуществлен алгоритм включения/отключения каналов по истечению запрограммированного времени; «readcard» - функция для чтения меток; «alarm» - здесь условия срабатывания сигнализации от появления движения.

Подробное описание строчек кода доступно в комментариях приложения Б.

## 12.4 Описание кода программы для контроллера ATmega328

Вверху программы приложения В, как и в случае с предыдущим контроллером расположены строчки, подключающие библиотеки (состав библиотек одинаков), задаются константы и переменные. В «setup» происходит инициализация всех портов и подключенных устройств.

В непрерывном цикле «loop» написан код условия для принятых данных от радиомодуля контроллера ATmega2560. Если имеются принятые данные, то поместить их в массив. Если в первой ячейке массива единица, то вызывается функция открытия ворот, иначе функция закрытия.

Далее идет условие отслеживания положения ворот по одному датчику. Если текущее положение закрыто, значит двигаемся в сторону открытия и по мере того, как ворота проходят контрольные точки положения (калитка, открыто) переменная положения ворот инкрементируется. И наоборот, если открыто, то с прохождением точек (калитка, закрыто) переменная декрементируется. Следующее условие в том, что если с момента открытия ворот прошло 40 секунд, то запустить функцию закрытия. Дальше условие, при котором ворота

останавливаются, когда возникает препятствие на пути (сигнал с датчика препятствия). И в конце вызывается функция остановки ворот.

Функции для работы логики открытия, закрытия и остановки ворот, прописаны после завершения кода функции «loop». Здесь же добавлены функции для настройки радиомодуля и обработки чтения метки.

Для подробного рассмотрения программы, доступны комментарии напротив строчек в приложении В.

## ЗАКЛЮЧЕНИЕ

По итогу в данной работе была разработа многофункциональная система «Умный дом» с автоматизацией внутри дома и автоматическими воротами.

В ходе работы были выполнены следующие задачи: рассчитаны моменты статических сопротивлений и выполнен предварительный расчет мощности двигателя. По результатам предварительного расчета мощности был выбран

асинхронный двигатель А71А2 мощностью 750 Вт и планетарным редуктором ЗП-25, с передаточный числом 25. Были приведены статические моменты и моменты инерции к валу двигателя, выполнена предварительная проверка электродвигателя по нагреву и производительности. Выбран преобразователь частоты ZW-AT1-1500W. Построены статические характеристики двигателя, а также переходные процессы.

Разработана система автоматизации, построена функциональная схема, определены входные и выходные параметры, построены схемы алгоритма, произведен выбор основных элементов схемы и выполнены принципиальные электрические схемы.

Большая часть затраченного в процессе проектирования времени пришлась на написание и отладку программного кода. Получилось три программы для микроконтроллеров, плюс невошедшее в текстовую часть работы программирование визуального интерфейса с применением языков для вёрстки сайтов. Для написания интерфейса использовались языки html, js, css. Вешний вид интерфейса представлен в приложении Д.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Первый умный дом в истории/ Интеграл-Д – http://integral-d.ru/stati/pervyj-umnyj.html (дата обращения: 5.03.20).

2. Какие бывают "умные дома". Обзор. / Павел Николаев – http://www.besmart.su/article/kakie-byvayut-umnye-doma (дата обращения: 10.03.20).

3. Борисов, А.М. Автоматизация технологических процессов (технические средства, проектирование, лабораторный практикум): Учебное пособие./ Н.Е. Лях. – Челябинск: Издательство ЮУрГУ, 2001. – Ч.1. – 404 с.

4. Расчет и конструирование станков: Метод. указ. к практическим занятиям / Сост. А.Ф.Денисенко – Самара: Самар. гос. техн. ун-т., 2013.

5. Стандартные асинхронные электродвигатели с короткозамкнутым ротором. Серии А, АИР с градацией мощности и присоединительных размеров по ГОСТ Р 51689 / ГОСТ Р 51689 – https://inni.info/produkt/elektrodvigateli-2/standartnyye-asinkhronnyye-elektrodvigateli-s-2 (дата обращения: 22.03.20).

6. Редукторы, мотор-редукторы / Редуктор НТС-К – https://reduktorntc-k.com.ua/katalog/planetarnie.pdf (дата обращения: 10.04.20).

7. Драчев, Г.И. Теория электропривода: Учебное пособие к курсовому и дипломному проектированию / Г.И. Драчев – Челябинск: Изд. ЮУрГУ, 2012. – 168 с.

8. Драчев, Г.И. Теория электропривода: Учебное пособие. – Челябинск: Издательство ЮУрГУ, 2005. Часть 1. – 209 с.

8. Каталог продукции компании Robotdyn / Robotdyn comp. – https://robotdyn.com/catalog.html (дата обращения: 16.04.20).

9. Платы Arduino, различные шилды, модули и датчики для Ардуино. / pl1mitino store – https://pl-1.org/catalog-chip/arduino-etc?p=3_30 (дата обращения: 23.04.20)

## ПРИЛОЖЕНИЕ А

### Листинг кода программы для ESP8266

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
```

```
#include <EEPROM.h>
#include "FS.h"

#define MAX_CHANNELS 10  //максимальное количество каналов
#define RANDOM_PIN A0 // установить пин к которому ничего не подключено для рандомизации хэша (значение берётся из шума пина)
// вписываем данные для подключения к сети:
const char* ssid = "Логин"; //логин сети
const char* password = "Пароль";//пароль сети

const char* ip_update_host = "hldns.ru";// Указываем хост (адрес сайта) службы DynDNS
const char* ip_update_get = "/update/AYKI0FAY2Q456KB7HHPW26MAUG1KFQ";// Указываем GET-запрос (набор параметров) адреса идентификации в службе DynDNS

ESP8266WebServer server(10200);//создаём объект для работы с веб-сервером, указав номер порта, по которому он будет доступен
WiFiClient client; //создаём объект для работы с удалёнными хостами
MDNSResponder mdns;

String getContentType(String filename); // функция отправления нужного заголовка по расширению файла
bool handleFileRead(String path);      // отправка нужного файла клиенту

String inString;//создаем переменную для строковых данных, полученных с МЕГИ
String chanel;//создаём переменную для формирования номера канала
//данные, которые передаются с МЕГИ и записываются в ЕСП
String DATA [MAX_CHANNELS] = {};
//наименования каналов
String CHANNELNames[MAX_CHANNELS] = {};
//типы каналов
String CHANNELTypes[MAX_CHANNELS] = {};
//состояния каналов
String STATE [MAX_CHANNELS] = {};
//состояния ползунков
String SLIDER [MAX_CHANNELS] = {};
//время до включения канала
String TIMERON [MAX_CHANNELS] = {};
//время до выключения канала
String TIMEROFF [MAX_CHANNELS] = {};
```

Продолжение приложения А

```
/*
  type 0 - выключатель со слайдером
  type 1 - обычный выключатель
*/
String HASH; //ключ для веб интерфейса
String Login = "admin"; //логин от сайта
```

```
String Password = "admin";//пароль от сайта

void write_String(char add,String data);
String read_String(char add);

bool f = 0; //флаг для единичной записи данных в EEPROM

void setup(void){
  pinMode(A0, INPUT);
  pinMode(5, INPUT_PULLUP);
  //инициализация массивов, вдруг данные не считаются с EEPROM
  for(int i = 0; i < MAX_CHANNELS; i++){
  TIMERON[i] = "0";
  TIMEROFF[i] = "0";
  SLIDER[i] = "100";
  STATE[i] = "0";
  CHANNELTypes[i] = "0";
  CHANNELNames[i] = "Chanel"+String(i);
  }
  // подготовка:
  Serial.begin(115200);
  while (!Serial) {
    ; //ждём пока подключится сериал порт
  }
  //синхронизация данных с EEPROM
  EEPROM.begin(512);
  readData(); //считываем данные с EEPROM

  // подключение к WiFi
  WiFi.mode(WIFI_STA); //выбираем режим работы WiFi контроллера
- клиент
  WiFi.begin(ssid, password);
  Serial.println("");

  // ожидание соединения:
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Подключились к ");  //  "Подключились к "
  Serial.println(ssid);
  Serial.print("IP-адрес: ");  //  "IP-адрес: "
  Serial.println(WiFi.localIP());
```

Продолжение приложения А

```
  //  Проверка запуска MDNS
  if (mdns.begin("esp8266", WiFi.localIP())) {
    Serial.println("MDNS запущен");
  }

  //генерация хэша
```

```
      randomSeed(analogRead(RANDOM_PIN)); //чтобы работала функция
random, исходя из шумов аналогового пина
      for(int i = 0; i < 5; i++){
      HASH += String(random(1000)*random(1000)+i);
      }
      Serial.println("Создан уникальный ключ "+HASH);

     // server.on("/", handleRoot);
      server.on("/action", handleAction);
      server.on("/getstate", handleCheck);
      server.on("/getstateall", handleCheckAll);

      server.begin();
      Serial.println("HTTP сервер запущен");
      ip_update();
      SPIFFS.begin();   // инициализация файловой системы

      server.onNotFound([]() {    // если клиент запросил URI
        if (!handleFileRead(server.uri()))   // отправляем его, если
существует
          server.send(404, "text/plain", "404: Not Found"); // если
нет, то отправляем сообщение 404
      });

      for (int chan = 0; chan < MAX_CHANNELS; chan++) { //цикл для
восстановления каналов, после потери питания
        if (STATE[chan] == "1") { //если канал был включен
          Serial.println("<ON>_" + String(chan) + "~");//отправить в
сериал команду МЕГЕ на включение
          if (CHANNELTypes[chan] == "0") Serial.println(SLIDER[chan]
+ "_"+ String(chan) +"@"); //если регулируемый, то выдать значение
          delay(200);
        }
        else //иначе
          Serial.println("<OFF>_" + String(chan) + "~");//отправить
в сериал команду МЕГЕ на отключение
          delay(200);
      }
    }

    void  ip_update(){//  Функция,  обновляющая  IP  маршрутизатора
(роутера) в службе динамических DNS
      if   (client.connect(ip_update_host,80))   //   Если   возможно
соединиться с хостом на 80-й порт:
```

Продолжение приложения А

```
        client.print("GET ");   //  Эта и следующие 7 строк отправляют
    системные заголовки на сервер службы динамических DNS
        client.print(ip_update_get);   //  Формат заголовков строгий
и в нём важны даже переводы строк
        client.println( " HTTP/1.1");
        client.print("Host:");
```

```
        client.println(ip_update_host);
        client.println( "Connection: close" );
        client.println();
        client.println();
        delay(500);  //  Задержка в 0.5 сек., чтобы дать удалённому
серверу возможность обработать и отправить ответ на наш запрос
        Serial.println("\nResponse of DDNS-Server:"); //   Шлём в
монитор серийного порта сообщение о том, что далее будет выводиться
ответ от сервера службы DynDNS
        while (client.available()) {   //  Пока сервер на связи
          char a = client.read();    //     считываем его ответ по
одному символу
          Serial.print(a);  //   и выводим на монитор серийного порта
        }
      }
    }

    void loop(void){
      server.handleClient(); //функция обработки запросов клиента
      if (digitalRead(5)&& millis() > (long)10000 && !f) {
        writeData();
        Serial.println("Пропало основное питание");
        f = 1; //флаг сохранения
      }
      if (!digitalRead(5) && f) f = 0; //если появилось питание, то
обнулить для след. сохранения
    //Проверка события на порту Serial
      while (Serial.available()) {
        char inChar = Serial.read();
        inString += inChar;//каждый цикл в строку записывается символ
        if (inChar == '^') {//если последний символ был "^"
          parse();
          int chan = chanel.toInt();
          if (inString.indexOf("ON")>0) {
            STATE[chan] = "1";
          }
          else if (inString.indexOf("OFF")>0) {
            STATE[chan] = "0";
          }
          inString = "";//после обработки обнуляем строки
          chanel = "";
        }
        else if (inChar == '#') {//если последний символ "#"
          parse();
```

Продолжение приложения А

```
          int chan = chanel.toInt();
          int value = inString.toInt();//получаем значение ползунка
в int
          inString.trim();
          SLIDER[chan] = inString;
```

```
          inString = "";
          chanel = "";
        }
        else if (inChar == '&') {//если последний символ "#"
          parse();
          int chan = chanel.toInt();
          int len = inString.length();//создаем переменную хранящую
длину строк
          int pos = inString.indexOf('!');//создаем переменную и
присваиваем ей позицию разделительного знака
          //разделяем пришедшие значения таймера
          String valon, valoff;
          for (int i = 0; i < pos; i++) {
            valon += inString[i];
          }
          for (int i = pos+1; i < len; i++) {
            valoff += inString[i];
          }
          valon.trim();//функция удаления литералов из переменной
          valoff.trim();
          TIMERON[chan] = valon;
          TIMEROFF[chan] = valoff;
          inString = "";
          chanel = "";
        }
      }
    }
    void parse() {
       int len = inString.length();//создаем переменную хранящую
длину строк
       int pos = inString.indexOf('_');//создаем переменную и
присваиваем ей позицию с которой начинается номер канала
       for (int i = pos+1; i < len-1; i++) {//запускаем цикл
формирования номера канала
        chanel += inString[i];//номер канала
       }
       inString.remove(pos);//удаляем конец строки с номером канала,
оставляя только значение
       return ;
    }
```

Продолжение приложения А

```
    void handleCheck() {
      if (server.hasHeader("Cookie")){  //есть ли в заголовках http
параметр coockie
        String cookie = server.header("Cookie"); //считываем параметр
куки
```

```
        if (cookie.indexOf("hash="+HASH) == -1) { //не нашёл, то
вернуть ошибку
          server.send(403, "text/plain", "Cookie Access Denied");
          return;
        }
      }
      int channel = server.arg("chanel").toInt();
      Serial.println("_"  +  String(channel)  +  "=");//отправляем
команду для обновления времени таймеров, если 0, то не включен
      String Json = "{"; //формируем json массив с набором параметров
для ответа серверу
        Json += "\"id\": "+server.arg("chanel")+",";
        Json += "\"name\": \""+CHANNELNames[channel]+"\",";
        Json += "\"type\": \""+CHANNELTypes[channel]+"\",";
        Json += "\"on\": \""+TIMERON[channel]+"\",";
        Json += "\"off\": \""+TIMEROFF[channel]+"\",";
        if(STATE[channel] == "1")  Json += "\"enabled\": true,";
        else Json += "\"enabled\": false,";
        Json += "\"value\": \"" + SLIDER[channel]+ "\"";
        Json += "}";
      server.send(200, "text/plain", Json);
    }

    void  handleCheckAll(){//функция  формирования  массива  для
обновления данных всех каналов
      String Json = "{";
      Json += "\"chanels\": [";
      for(int i = 0; i < MAX_CHANNELS; i++) {
        Json += "{";
            Json += "\"id\": "+String(i)+",";
            Json += "\"name\": \""+CHANNELNames[i]+"\",";
            Json += "\"type\": \""+CHANNELTypes[i]+"\",";
            Json += "\"on\": \""+TIMERON[i]+"\",";
            Json += "\"off\": \""+TIMEROFF[i]+"\",";
            if(STATE[i] == "1")  Json += "\"enabled\": true,";
            else Json += "\"enabled\": false,";
            Json += "\"value\": \"" + SLIDER[i]+ "\"";
        Json += "}";
        if(i < MAX_CHANNELS-1) Json += ",";
      }
      Json += "]";
      Json += "}";
      server.send(200, "text/plain", Json);
    }
```

Продолжение приложения А

```
    void handleAction(){
      if(!server.hasArg("action")) { //если вообще нет ключа action
то вернуть сразу 404
        server.send(404,  "text/plain",  "Action  and  channel  not
found");
```

```
        return;
      }
      String action = server.arg("action"); //получаем аргумент,
какой сценарий запустить
      if(action == "auth"){
        if(server.hasArg("login")                                    &&
server.hasArg("password")){//если имеется логин и пароль
          if(Login   ==   server.arg("login")   &&   Password   ==
server.arg("password")){//проверяем на совпадение
            server.send(200, "text/plain", HASH);//если совпадает,
то отправляем успешно и ключ
            return;
          } else {
            server.send(403, "text/plain", "auth Access Denied");
            return;
          }
        } else {
          server.send(403, "text/plain", "auth Access Denied");
          return;
        }
      }
        if(action == "authcheck"){
            if(!server.hasArg("hash")) {
                server.send(403,  "text/plain",  "authcheck  Access
Denied");
                return;
            } else {
              if(server.arg("hash") == HASH) {
                  server.send(200, "text/plain", "success");
                  return;
              } else {
                  server.send(403,   "text/plain",   "hash   Access
Denied");
                  return;
              }
            }
        }
        if (server.hasHeader("Cookie")){  //есть ли в заголовках http
параметр coockie
            String  cookie  =  server.header("Cookie");  //считываем
параметр куки
            if (cookie.indexOf("hash="+HASH) == -1) { //не нашёл, то
вернуть ошибку
                server.send(403,   "text/plain",   "Cookie   Access
Denied");
```

Продолжение приложения А

```
                return;
            }
        }
        if(!server.hasArg("chanel")) { //если вообще нет ключа chanel
то вернуть сразу 404
```

```
            server.send(404, "text/plain", "Action and chanel not
found");
            return;
        }
        int channel = server.arg("chanel").toInt();
        if (action == "on") { //если у нас совпадение сценария с on
            server.send(200, "text/plain", "on success"); //вернуть
200 OK
            Serial.println("<ON>_"     +     String(channel)     +
"~");//отправить в сериал команду МЕГЕ
            if        (CHANNELTypes[channel]        ==        "0")
Serial.println(SLIDER[channel] + "_"+ String(channel) +"@");//если
тип канала "0"(регулятор), то отправляем значение ползунка МЕГЕ
            return;
        }
        if (action == "off") { //если у нас совпадение сценария с off
            server.send(200, "text/plain", "off success");
            Serial.println("<OFF>_"     +     String(channel)     +
"~");//отправить в сериал команду МЕГЕ
            return;
        }
        if (action == "range") { //если это ползунок
            //слайдеру выдать то значение которое пришло с аргумента
value
            if(server.hasArg("value")){
                server.send(200,  "text/plain",  "range  success");
//вернуть 200 OK
                Serial.println(server.arg("value")      +      "_"+
String(channel) +"@");//отправляем значение с ползунка в сериал МЕГЕ
            }
            return;
        }
        if (action == "settings") {
            server.send(200,  "text/plain",  "settings  success");
//вернуть 200 OK
            if(server.hasArg("chanel_name")) CHANNELNames[channel] =
server.arg("chanel_name");
            if(server.hasArg("type")) {
              CHANNELTypes[channel] = server.arg("type");
              Serial.println(server.arg("type")     +     "_"     +
String(channel) + "?");
            }
            return;
        }
        if (action == "timer") {
```

Продолжение приложения А

```
            server.send(200,   "text/plain",   "timer   success");
//вернуть 200 OK
            if(server.hasArg("timeon")){
```

```
                Serial.println(server.arg("timeon")    +    "_"    +
String(channel) + "+");//отправляем команду МЕГЕ на ВКЛючение канала
спустя value минут
            }
            if(server.hasArg("timeoff")){
                Serial.println(server.arg("timeoff")    +    "_"    +
String(channel) + "-");//отправляем команду МЕГЕ на ВЫКЛючение канала
спустя value минут
            }
            return;
        }
        server.send(404,  "text/plain",  "Action  and  chanel  not
found");
    }

    void readData(){
      Serial.println("Чтение данных из EEPROM"); //выводим что сейчас
будем читать из EEPROM
      unsigned int offset = 0; //начало смещения
      for(int i = 0; i < MAX_CHANNELS; i++){ //заводим цикл от 0 до
9 (10 итераций)
        String recivedData; //сюда придёт дата которая была в EEPROM
        recivedData = read_String(offset); //считываем из EEPROM с
учётом смещений
        Serial.println("Chanel   Name:   "+recivedData+";   Length:
"+recivedData.length()+"; Index: "+i+"; Offset: "+offset); //выводим
для отладки
        offset += 15+1; //создаём смещение исходя от длины предыдущих
данных +1 это обрыв строки \0
        recivedData.trim();
        CHANNELNames[i] = recivedData; //записываем имя канала

        recivedData = read_String(offset); //считываем из EEPROM с
учётом смещений
        Serial.println("Chanel   Type:   "+recivedData+";   Length:
"+recivedData.length()+"; Index: "+i+"; Offset: "+offset); //выводим
для отладки
        offset += 2; //+2 это у нас 1 сам тип и 1 обрыв строки \0
        recivedData.trim();
        CHANNELTypes[i] = recivedData; //записываем тип канала

        recivedData = read_String(offset); //считываем из EEPROM с
учётом смещений
        Serial.println("Chanel   STATE:   "+recivedData+";   Length:
"+recivedData.length()+"; Index: "+i+"; Offset: "+offset); //выводим
для отладки
```

Продолжение приложения А

```
        offset += 2; //создаём смещение исходя от длины последней
записи +2 это обрыв строки \0 и размер состояния канала
        recivedData.trim();
```

```
        STATE[i] = recivedData; //записываем тип канала

        recivedData = read_String(offset); //считываем из EEPROM с
учётом смещений
        Serial.println("Chanel  SLIDER:  "+recivedData+";  Length:
"+recivedData.length()+"; Index: "+i+"; Offset: "+offset); //выводим
для отладки
        offset += 4; //создаём смещение исходя от длины последней
записи +4 это обрыв строки \0 и размер значения
        recivedData.trim();
        SLIDER[i] = recivedData;//записываем значение ползунка
        delay(10);
      }
    }

    void writeData(){//функция записи всех данных в EEPROM (будет
вызвана один раз, при потере основного питания)
      unsigned int offset = 0; //индикатор смещения

      Serial.println("Запись данных в EEPROM"); //говорим что сейчас
начнём запись
      for(int i = 0; i < MAX_CHANNELS; i++){ //запускаем цикл
        write_String(offset,  CHANNELNames[i]);    //записываем  по
индексу массива с учётом смещений
        Serial.println("Chanel  Name:  "+CHANNELNames[i]+";  Length:
"+CHANNELNames[i].length()+"; Index: "+i+"; Offset: "+offset); //для
наглядности выводим инфу для отладки
        offset += 15+1; //создаём смещение исходя от длины последней
записи +1 это обрыв строки \0

        write_String(offset,  CHANNELTypes[i]);    //записываем  по
индексу массива с учётом смещений
        Serial.println("Chanel  Type:  "+CHANNELTypes[i]+";  Length:
"+CHANNELTypes[i].length()+"; Index: "+i+"; Offset: "+offset); //для
наглядности выводим инфу для отладки
        offset += 2; // 1 сам тип и 1 обрыв строки \0

        write_String(offset,  STATE[i]);   //записываем  по  индексу
массива с учётом смещений
        Serial.println("Chanel    STATE:    "+STATE[i]+";    Length:
"+STATE[i].length()+";  Index:  "+i+";  Offset:  "+offset);  //для
наглядности выводим инфу для отладки
        offset += 2; //создаём смещение исходя от длины последней
записи 1 сам тип и 1 обрыв строки \0

        write_String(offset, SLIDER[i]);   //записываем по индексу
массива с учётом смещений
```

Продолжение приложения А

```
        Serial.println("Chanel   SLIDER:   "+SLIDER[i]+";   Length:
"+SLIDER[i].length()+";  Index:  "+i+";  Offset:  "+offset);  //для
наглядности выводим инфу для отладки
```

```
        offset += 4; //создаём смещение исходя от длины последней
записи +4 это обрыв строки \0 и размер значения
        delay(10);
      }
    }

    String read_String(char add) {
      char data[100]; //Изначально задаём максимальное количество
байтов которое может записаться из EEPROM
      int len = 0;
      unsigned char k; //сама метка для считывания char данных с
EEPROM
      k = EEPROM.read(add);
      while(k != '\0' && len < 100) {    //Начинаем считывание в
бесконечном цикле пока наша char k не будет равняться обрыву \0 или
же длинна len не выйдет за пределы 100
        k = EEPROM.read(add + len); //начинаем считывать из EEPROM со
смещением извне add + смещение внутри len++
        data[len] = k; //записываем в char массив данных (если вдруг
k найдёт \0, то цикл завершается)
        len++; //плюсуем для смещения
      }
      return String(data); //превращаем в string и возвращаем
    }

    void write_String(char add,String data) {
      int len = data.length(); //получаем длину строки
      int i; //для цикла
      for(i = 0;i < len; i++) { //начинаем перебирать каждый char
символ строки
        EEPROM.write(add + i,data[i]); //записываем в данные с учётом
смещений от add (которое пришло извне) + i которая уже в самой строке
      }
      EEPROM.write(add + len,'\0');//В конце добавляем обрыв строки
\0
      EEPROM.commit();
    }
    bool handleFileRead(String path){  // Функция работы с файловой
системой
      if(path.endsWith("/"))       path       +=       "index.html";
//  Если устройство вызывается по корневому адресу, то должен
вызываться файл index.html (добавляем его в конец адреса)
      String       contentType       =       getContentType(path);
// С помощью функции getContentType (описана ниже) определяем по типу
файла (в адресе обращения) какой заголовок необходимо возвращать по
его вызову
```

Продолжение приложения А

```
      if(SPIFFS.exists(path)){ // Если в файловой системе существует
файл по адресу обращения
```

```
        File file = SPIFFS.open(path, "r");  //  Открываем файл для
чтения
        size_t  sent  =  server.streamFile(file,  contentType);    //
Выводим содержимое файла по HTTP, указывая заголовок типа содержимого
contentType
        file.close();    //  Закрываем файл
        return true;     //  Завершаем выполнение функции, возвращая
результатом ее исполнения true (истина)
      }
      return false;     // Завершаем выполнение функции, возвращая
результатом ее исполнения false (если не обработалось предыдущее
условие)
    }

    String getContentType(String filename){  // Функция, возвращающая
необходимый заголовок типа содержимого в зависимости от расширения
файла
      if     (filename.endsWith(".html"))     return     "text/html";
// Если файл заканчивается на ".html", то возвращаем заголовок
"text/html" и завершаем выполнение функции
      else   if   (filename.endsWith(".css"))   return   "text/css";
// Если файл заканчивается на ".css", то возвращаем заголовок
"text/css" и завершаем выполнение функции
      else        if        (filename.endsWith(".js"))        return
"application/javascript";   // Если файл заканчивается на ".js", то
возвращаем заголовок "application/javascript" и завершаем выполнение
функции
      else  if  (filename.endsWith(".ico"))  return  "image/x-icon";
// Если файл заканчивается на ".ico", то возвращаем заголовок
"image/x-icon" и завершаем выполнение функции
      else if (filename.endsWith(".svg")) return "image/svg+xml";
      return "text/plain";                                          //
Если ни один из типов файла не совпал, то считаем что содержимое файла
текстовое, отдаем соответствующий заголовок и завершаем выполнение
функции
    }
```

## ПРИЛОЖЕНИЕ Б

Листинг кода программы для ATmega2560

```
    #include <SPI.h>
    #include <MFRC522.h>// библиотека "RFID"
    #include "nRF24L01.h" // библиотеки для радио-модуля
    #include "RF24.h"
    #include <Wtv020sd16p.h>
    //константы для музыкального модуля
    #define resetPin 31
    #define clockPin 32
    #define dataPin 33
    #define busyPin 34
    //константы для модулей SPI
    #define SS_PIN 20
    #define SS_PIN2 21
    #define RST_PIN 23
    #define RST_PIN2 24
    //константы для настройки радиомодуля
    #define CH_NUM 0x60// номер канала (должен совпадать с
передатчиком)
    #define SIG_POWER RF24_PA_MIN// уровень мощности
    #define SIG_SPEED RF24_1MBPS// скорость обмена
    //константы для каналов управления
    #define START_PIN_OUT 2 //Начальный пин нулевого выходного кнала
    #define START_PIN_IN 35//Начальный пин нулевого входного канала
для выключателей
    #define GATE_PIN 10 // пин для канала управления ворот
    #define ALARM_PIN 11 // пин для канала сигнализации
    #define JAL_PIN 12 //начальный пин для каналов управления жалюзи
    #define MAX_JAL 1 // количество жалюзи
    #define MAX_CHANNELS 11 //максимальное количество каналов

    String inString;//создаем переменную для строковых данных,
полученных с ESP
    String chanel;//создаём переменную для номера канала

    unsigned int TIMERON[MAX_CHANNELS] = {};//храним уставку таймера
на включение в минутах
    unsigned int TIMEROFF[MAX_CHANNELS] = {};//храним уставку таймера
на выключение в минутах
    unsigned int CHANNELTypes[MAX_CHANNELS] = {};//храним типы
каналов
    unsigned int STATE[MAX_CHANNELS] = {};//храним состояния каналов
    unsigned int SLIDER[MAX_CHANNELS] = {};//храним состояния
ползунков

    unsigned long timeron[MAX_CHANNELS] = {};//значения миллисекунд
в момент установки таймера на включение
```

Продолжение приложения Б

```
    unsigned long timeroff[MAX_CHANNELS] = {};//значения миллисекунд
в момент установки таймера на выключение
    unsigned long timerjal[MAX_JAL] = {};//зачения миллисекунд в
момент начала движения жалюзи
    unsigned long uidDec, uidDecTemp;// для храниения номера RFID
метки в десятичном формате
    unsigned long base[10] = {};//база меток для
активации/деактивации охранной сигнализации
    unsigned long alarm_timer; // значение миллисекунд в момент
активирования сигнализации
    unsigned long sensor_timer; // значение миллисекунд в момент
нажатия на сенсорный выключатель

    int transmit_data[2];  // массив пересылаемых данных

    bool alarm_working; // сигнализация работает(сирена)
    bool alarm_active; // сигнализация активна
    bool j = 0;
    bool k = 0;

    MFRC522 mfrc522(SS_PIN, RST_PIN);//создаем обект для считывателя
и указываем пины
    RF24 radio(RST_PIN2, SS_PIN);   // создаем объект для радиомодуля
    Wtv020sd16p      wtv020sd16p(resetPin,clockPin,dataPin,busyPin);
//объект для работы музыкального модуля

    // Настройка
    void setup() {
      SPI.begin();//инициализация SPI
      mfrc522.PCD_Init();// инициализация MFRC522
      wtv020sd16p.reset();// инициализация музыкального модуля
      radioSetup(); //настройка радиомодуля
    // Инициализация
      Serial3.begin(115200);
      Serial2.begin(9600);
      Serial.begin(115200);
      for (int i = START_PIN_OUT; i > START_PIN_OUT + MAX_CHANNELS;
i++) {//инициализируем выходы
        pinMode(i, OUTPUT);
      }
      for (int i = START_PIN_IN; i > START_PIN_IN + MAX_CHANNELS;
i++) {//инициализируем входы
        pinMode(i, INPUT);
      }
      for (int i = 0; i < MAX_JAL; i++) { //инициализация выходов для
управления транзисторами
        pinMode (40+(2*i), OUTPUT);
        pinMode (41+(2*i), OUTPUT);
      }
      pinMode(22, INPUT); //сигнал датчика движения
```

Продолжение приложения Б

```
      pinMode(RST_PIN, OUTPUT);
      pinMode(RST_PIN2, OUTPUT);
      pinMode(SS_PIN, OUTPUT);//для SS
      pinMode(SS_PIN2, OUTPUT);//для SS
      while(!Serial2.available()){  //  ждем  инициализацию  модуля
SIM800L
         Serial2.println("AT"); // отправляем команду "AT"
         delay(1000);
         Serial.println("Подключение SIM800L");
         }
         Serial.println("Подключились!");
         Serial2.println("AT+CMGF=1");    //    отправляем    команду
AT+CMGF=1
         delay(1000);
         Serial2.println("AT+CNMI=1,2,0,0,0"); // отправляем команду
AT+CNMI=1,2,0,0,0
         delay(1000);
         Serial2.println("AT+CMGL=\"REC UNREAD\"");
    }

    // Выполнение
    void loop() {
      while (Serial3.available()) {//3
        // Чтение данных из порта Serial3
        char inChar = Serial3.read();//3
        // Вывод прочитанных данных в порт Serial
        inString += inChar;//каждый цикл в строку записывается символ
        if (inChar == '~') {//если последний символ был ")"
          parse();
          int chan = chanel.toInt();//получаем номер канала
          if (inString.indexOf("ON")>0) {
            STATE[chan] = 1;
            digitalWrite(START_PIN_OUT + chan, HIGH);
            Serial3.println("<ON>_"          +          (String)chan
+"^");//отправляем ответ об успешной обработки ESP
          }
          else if (inString.indexOf("OFF")>0) {
            STATE[chan] = 0;
            digitalWrite(START_PIN_OUT + chan, LOW);
            Serial3.println("<OFF>_"         +          (String)chan
+"^");//отправляем ответ об успешной обработки ESP
          }
          inString = "";
          chanel = "";
        }
        else if (inChar == '@') {//если последний символ "@"
          parse();
          int chan = chanel.toInt();//получаем номер канала
          int value = inString.toInt();//получаем значение ползунка
в int для присвоения ШИМ
```

Продолжение приложения Б

```
          value = map(value, 0, 100, 0, 255);//накладываем маску на
значение
          value = constrain(value, 0, 255);//создаём жесткие границы
переменной
          SLIDER[chan] = value;
          if (value == 255) { //если максимальное заполнение, то
            digitalWrite(START_PIN_OUT   +   chan,   HIGH);   //выдать
высокий уровень
            Serial3.println(inString    +    "_"    +    (String)chan
+"#");//отправляем ответ об успешной обработке данных ESP
            inString = "";
            chanel = "";
          }
          else if (value == 0) {//если минимальное заполнение, то
            digitalWrite(START_PIN_OUT + chan, LOW); //выдать низкий
уровень
            Serial3.println(inString    +    "_"    +    (String)chan
+"#");//отправляем ответ об успешной обработке данных ESP
            inString = "";
            chanel = "";
          }
          else {
            analogWrite(START_PIN_OUT   +   chan,   value);//функция
генерации ШИМ сигнала на пине
            Serial3.println(inString    +    "_"    +    String(chan)
+"#");//отправляем ответ об успешной обработке данных ESP
            inString = "";
            chanel = "";
            delay(1000);
          }
        }
        else if (inChar == '+') {//если последний символ "+"
          parse();
          int chan = chanel.toInt();//получаем номер канала
          int value = inString.toInt();//получаем значение таймера
          if  (value  !=  0)  timeron[chan]  =  millis();//чтобы  не
занимать память
          TIMERON[chan]  =  value;//присваиваем  уставку  таймера  в
минутах
          inString = "";
          chanel = "";
        }
        else if (inChar == '-') {//если последний символ "-"
          parse();
          int chan = chanel.toInt();//получаем номер канала
          int value = inString.toInt();//получаем значение таймера
          if  (value  !=  0)  timeroff[chan]  =  millis();//чтобы  не
занимать память
          TIMEROFF[chan]  =  value;//присваиваем  уставку  таймера  в
минутах
```

Продолжение приложения Б

```
          inString = "";
          chanel = "";
        }
        else if (inChar == '=') {//если последний символ "="
          parse();
          int chan = chanel.toInt();//получаем номер канала
          int value = inString.toInt();//получаем значение таймера
          //отправить оставшееся время
          String on, off;
          if (TIMERON[chan] != 0) on = String((timeron[chan] / 1000
+ (long)TIMERON[chan] *60) - millis() / 1000);
          //если таймер включен, то присвоить оставшееся время в
секундах
          else on = "0";
          if (TIMEROFF[chan] != 0) off = String((timeroff[chan] /
1000 + (long)TIMEROFF[chan] *60) - millis() / 1000);
          //если таймер включен, то присвоить оставшееся время в
секундах
          else off = "0";
          Serial3.println(on + "!" + off + "_" + (String)chan + "&");
//ответ
          inString = "";
          chanel = "";
        }
        else if (inChar == '?') {//если последний символ "?"
          parse();
          int chan = chanel.toInt();//получаем номер канала
          int value = inString.toInt();//получаем значение
          CHANNELTypes[chan] = value;//присваиваем тип канала
          inString = "";
          chanel = "";
        }
      }
      //тут проверка на изменение передаваемых данных
(энергоэффективная отправка)
      if (transmit_data[0] != digitalRead(GATE_PIN)) {//если не равно
текущему
        transmit_data[0] = digitalRead(GATE_PIN); //присвоить
        radio.write(&transmit_data,          sizeof(transmit_data));
//отправить в эфир
      }
      //условия для работы жалюзи
      for (int i = 0; i < MAX_JAL; i++) { //цикл проверки каналов
жалюзи
        if (digitalRead(JAL_PIN+i) && !j && !k) {//если есть высокий
уровень сигнала(открыть)
          j = true;
          k = true;
          timerjal[i] = millis(); //запоминаем время начала движения
          digitalWrite(41+(2*i), LOW); //снимаем другой сигнал
```

Продолжение приложения Б

```
          digitalWrite(40+(2*i), HIGH); //подаем сигнал "открыть"
        }
        else if (!digitalRead(JAL_PIN+i) && j && !k) { //если есть
низкий уровень сигнала(закрыть)
          j = false;
          k = true;
          timerjal[i] = millis(); //запоминаем время начала движения
          digitalWrite(40+(2*i), LOW); //снимаем другой сигнал
          digitalWrite(41+(2*i), HIGH); //подаем сигнал "закрыть"
        }
        if (millis() - timerjal[i] > (long)38000) { //если
закончилось время движения
          k = false;
          digitalWrite(40+(2*i), LOW);//остановить
          digitalWrite(41+(2*i), LOW);
        }
      }
      //условия для работы сенсорных выключателей
      for (int i = START_PIN_IN; i < START_PIN_IN + MAX_CHANNELS;
i++) { //запускаем цикл для опрашивания сигналов выключателей
       if (digitalRead(i) && STATE[i-START_PIN_IN] == 0 && millis()
- sensor_timer > (long)2000) {// если есть сигнал и состояние канала
этого пина - 0

//и прошло больше 2с с последнего сигнала, то
        Serial3.println("<ON>" + (String)i + "~"); // отправляем
команду на включение канала
        sensor_timer = millis(); // обновляем время последнего
сигнала
        return;
       }
       else if (digitalRead(i) && STATE[i-START_PIN_IN] == 1 &&
millis() - sensor_timer > (long)2000) {// если есть сигнал и состояние
канала этого пина - 1

//и прошло больше 2с с последнего сигнала, то
        Serial3.println("<OFF>" + (String)i + "~"); // отправляем
команду на выключение канала
        sensor_timer = millis(); // обновляем время последнего
сигнала
        return;
       }
      }
      timer();//вызов функции таймеров
      readcard();//вызов функции чтения метки
      alarm_();//вызов функции сигнализации
    }

    void parse() {//функция чтения и разбора строки пришедшей в сериал
порт
```

Продолжение приложения Б

```
        int len = inString.length();//создаем переменную хранящую
длину строк
        int pos = inString.indexOf('_');//создаем переменную и
присваиваем ей позицию с которой начинается номер канала
        for (int i = pos+1; i < len-1; i++) {//запускаем цикл
формирования номера канала
         chanel += inString[i];//номер канала
        }
        inString.remove(pos);//удаляем конец строки с номером канала,
оставляя только значение
        return;
    }

    void timer() {
      for (int chan = 0; chan < MAX_CHANNELS; chan++) {
        if(STATE[chan] == 0){ //если канал выключен, то отсчитываем
время включения
            if (TIMERON[chan] != 0 && millis() - timeron[chan] >=
(long)TIMERON[chan] * 60 * 1000) {
                if         (CHANNELTypes[chan]         !=         0)
digitalWrite(START_PIN_OUT + chan, HIGH); //если тип канала не равен
0, то сразу выдать высокий уровень
                else if (SLIDER[chan] != 0 && SLIDER[chan] != 255)
analogWrite(START_PIN_OUT + chan, SLIDER[chan]);//иначе генерировать
ШИМ сигнала на пине
                else digitalWrite(START_PIN_OUT + chan, HIGH);
                Serial3.println("<ON>_"   +   (String)chan   +"^");
//ответ
                STATE[chan] = 1;//обновляем состояние канала
                timeron[chan] = 0;//обнуляем таймер
                TIMERON[chan] = 0;//обнуляем уставку
            }
        } else timeron[chan] = millis();

        if(STATE[chan] == 1){ //если канал включен, то отсчитываем
время выключения
            if (TIMEROFF[chan] != 0 && millis() - timeroff[chan] >=
(long)TIMEROFF[chan] * 60 * 1000) {
                digitalWrite(START_PIN_OUT + chan, LOW);
                Serial3.println("<OFF>_"   +   (String)chan   +"^");
//ответ
                STATE[chan] = 0;//обновляем состояние канала
                timeroff[chan] = 0;//обнуляем таймер
                TIMEROFF[chan] = 0;//обнуляем уставку
            }
        } else timeroff[chan] = millis();
      }
    }

    void readcard() {
```

Продолжение приложения Б

```
        // Поиск новой метки
      if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
      }
      // Выбор метки
      if (!mfrc522.PICC_ReadCardSerial()) {
        return;
      }
      uidDec = 0;
      // Выдача серийного номера метки.
      for (byte i = 0; i < mfrc522.uid.size; i++)
      {
        uidDecTemp = mfrc522.uid.uidByte[i];
        uidDec = uidDec * 256 + uidDecTemp;
      }
      Serial.println("UID метки: ");
      Serial.println(uidDec); // выводим UID метки
      for (int i = 0; i < 10; i++) {
        if (uidDec == base[i]) {// сравниваем Uid метки с базой и
если метка верная, то
          // включаем/выключаем активность сигнализации
          if (digitalRead(ALARM_PIN)){
            digitalWrite(ALARM_PIN, LOW);
            //отправить команду ESP
          }
          else {
            digitalWrite(ALARM_PIN, HIGH);
            //отправить команду ESP
          }
        }
        return;
      }
    }

    void alarm_() {
      if (!digitalRead(ALARM_PIN)) {
        wtv020sd16p.stopVoice();   //останавливаем   воспроизведение
звука сигнализации
        alarm_working = false; // обнуляем событие "сигнализация
работает"
        alarm_active = false; // обнуляем активность
      }
      else if (digitalRead(ALARM_PIN) && !alarm_active) {
        alarm_active = true; // включаем событие активности
        alarm_timer = millis(); //запомнить время, когда активировали
сигнализацию
      }
      if (digitalRead(ALARM_PIN) && digitalRead(22) && millis() -
alarm_timer >= 30*1000 && !alarm_working) {
```

Продолжение приложения Б

```
      // если сигнализация включена, есть сигнал с датчика и
активность > 30 сек., то
        wtv020sd16p.playVoice(1);    //    воспроизводим    аудиофайл
сигнализации (воспроизведение не зависит от остального кода)
        sms("Тревожное сообщение!", "+7xxxxxxxxxx"); //отправляем
сообщение о проникновении на указанный номер
        alarm_working = true; // активируем событие "сигнализация
работает"
      }
    }

    void sms(String text, String phone) { // отправка SMS
      Serial.println("Отправка SMS");
      Serial2.println("AT+CMGS=\"" + phone + "\"");
      delay(500);
      Serial2.print(text);
      delay(500);
      Serial2.print((char)26);
      delay(500);
      Serial.println("Отправка SMS завершена");
    }

    void radioSetup() {          // настройка радио
      radio.begin();               // активировать модуль
      radio.setAutoAck(1);          // режим подтверждения приёма, 1
вкл 0 выкл
      radio.setRetries(0,  15);          //  (время  между  попыткой
достучаться, число попыток)
      radio.enableAckPayload();   // разрешить отсылку данных в ответ
на входящий сигнал
      radio.setPayloadSize(32);    // размер пакета, байт
      radio.openReadingPipe(1, 0xAABBCCDD11LL);   // адрес трубы
      radio.setChannel(CH_NUM);                 // выбираем канал (в
котором нет шумов!)
      radio.setPALevel(SIG_POWER);              // уровень мощности
передатчика
      radio.setDataRate(SIG_SPEED);           // скорость обмена
      radio.powerUp();         // начать работу
      radio.startListening();  // начинаем слушать эфир, мы приёмный
модуль
    }
```

ПРИЛОЖЕНИЕ В

Листинг кода программы для ATmega328

```
#include <SPI.h>
#include <MFRC522.h>// библиотека "RFID"
#include "nRF24L01.h" // библиотеки для радио-модуля
#include "RF24.h"
//константы для SPI устройств
#define SS_PIN 7
#define SS_PIN2 8
#define RST_PIN 9
#define RST_PIN2 6
//константы для настройки радиомодуля
#define CH_NUM 0x60// номер канала (должен совпадать с передатчиком)
#define SIG_POWER RF24_PA_MIN// уровень мощности
#define SIG_SPEED RF24_1MBPS// скорость обмена
//константы для ворот
#define PIN_STATE 2 // пин датчика положения ворот
#define PIN_CLOSE 3
#define PIN_OPEN 4
#define PIN_LET 5

MFRC522 mfrc522(SS_PIN, RST_PIN);//создаем обект для считывателя и указываем пины
RF24 radio(RST_PIN2, SS_PIN2);  // создаем объект для радиомодуля

unsigned long uidDec, uidDecTemp;// для храниения номера RFID метки в десятичном формате
unsigned long base_auto[10] = {};//здесь база меток машин, которые могут открыть ворота
unsigned long base_humans[10] = {};//здесь база меток людей, которые могут открыть ворота
unsigned long gate_timer; //таймер для ворот

int recieved_data[2];  // массив принятых данных
int gate_state = 0; // положение ворот

byte pipeNo; // номер трубы

bool human = 0;// входит человек
bool s; // флаг
bool move_; // направление движения

void setup() {
  pinMode(PIN_CLOSE, OUTPUT);//инициализируем пин на вывод (сигнал на закрытие)
  pinMode(PIN_OPEN, OUTPUT);//инициализируем пин на вывод (сигнал на открытие)
```

Продолжение приложения В

```
      pinMode(PIN_LET,  INPUT_PULLUP);//инициализируем пин на ввод
(сигнал с датчика препятствия) с подтяжкой
      pinMode(PIN_STATE,  INPUT_PULLUP);  //сигнал датчика положения
ворот
      pinMode(SS_PIN, OUTPUT);
      pinMode(SS_PIN2, OUTPUT);
      Serial.begin(9600);
      SPI.begin();//инициализация SPI
      mfrc522.PCD_Init();// инициализация MFRC522
      radioSetup();
    }

    void loop() {
      if  (radio.available(&pipeNo))  {  //  если  в  буфере  имеются
принятые данные, то получаем номер трубы по которой эти данные пришли
в переменную pipeNo
        radio.read(   &recieved_data,   sizeof(recieved_data));   //
помещаем данные из буфера в массив, указывая сколько всего байт может
поместиться в массив
        if  (recieved_data[0]  ==  1)  gate_open();  //если  в  первой
ячейке массива "1", то переходим в функцию открытия ворот
        else gate_close(); //иначе переходим в функцию закрытия ворот
      }
      readcard();//вызов функции чтения метки
      if  (digitalRead(PIN_STATE))  s  =  0;//если  сошли  с  датчика
обнуляем флаг
      if (gate_state == 0) move_ = 0; //если закрыто, то включаем
флаг что будем двигаться на "открытие"
      if (gate_state == 2) move_ = 1; //если открыто, то включаем
флаг что будем двигаться на "закрытие"
      if (!digitalRead(PIN_STATE) && gate_state < 2 && !move_ && !s)
{ // если есть контакт датчика и движемся на "открытие"
        gate_state++; // обновляем состояние
        s = 1; //устанавливаем флаг, что уже обновляли состояние
      }
      else if (!digitalRead(PIN_STATE) && gate_state >= 2 && move_ &&
!s) { //иначе, если есть контакт датчика и движемся на "закрытие"
        gate_state--; // обновляем состояние
        s = 1; //устанавливаем флаг, что уже обновляли состояние
      }

      if    (millis()    -    gate_timer    >=    (long)40    *    1000)
gate_close();//спустя 40 секунд после открытия - закрываем
      if (!digitalRead(PIN_LET)) gate_open(); //если препятствие, то
открываем ворота
      gate_stop();  //проверяем  положение  ворот  и  останавливаем  по
условиям
    }
    void gate_open() {
      if (gate_state != 2) { //если ворота не открыты, то
```

Продолжение приложения В

```
        digitalWrite(PIN_CLOSE, LOW); //снимаем сигнал на закрытие
        digitalWrite(PIN_OPEN, HIGH); //открываем ворота
      }
      else digitalWrite(PIN_OPEN, LOW); //останавливаем ворота
      gate_timer = millis();//обновляем таймер
    }
    void gate_close() {
      if (gate_state != 0) { //если ворота не закрыты, то
        digitalWrite(PIN_OPEN, LOW); //снимаем сигнал на открытие
        digitalWrite(PIN_CLOSE, HIGH); //закрываем ворота
      }
      else digitalWrite(PIN_CLOSE, LOW); //останавливаем ворота
    }

    void gate_stop() {
      if (gate_state = 2 && digitalRead(PIN_OPEN)) { //если открыто
и шло на открытие, то
        digitalWrite(PIN_OPEN, LOW); //останавливаем ворота
      }
      if (gate_state == 0 && digitalRead(PIN_CLOSE)) { //если закрыто
и шло на закрытие, то
        digitalWrite(PIN_CLOSE, LOW); //останавливаем ворота
      }
      if (gate_state == 1 && digitalRead(PIN_CLOSE) && human) { //если
шло на открытие и "человек" и среднее положение, то
        digitalWrite(PIN_CLOSE, LOW); //останавливаем ворота
        human = 0; // обнуляем событие "человек"
      }
    }

    void readcard() {
        // Поиск новой метки
      if (!mfrc522.PICC_IsNewCardPresent()) {
        return;
      }
      // Выбор метки
      if (!mfrc522.PICC_ReadCardSerial()) {
        return;
      }
      uidDec = 0;
      // Выдача серийного номера метки.
      for (byte i = 0; i < mfrc522.uid.size; i++)
      {
        uidDecTemp = mfrc522.uid.uidByte[i];
        uidDec = uidDec * 256 + uidDecTemp;
      }
      Serial.println("UID метки: ");
      Serial.println(uidDec); // выводим UID метки в консоль
      for (int i = 0; i < 10; i++) {
```

Продолжение приложения В

```
        if (uidDec == base_auto[i]) {// сравниваем Uid метки с базой
машин и если метка верная, то
          human = 0; // обнуляем событие "человек"
          gate_open();// открываем ворота
          return;
        }
        else if (uidDec == base_humans[i]) { //иначе, если метка из
списка людей, то
          human = 1; //поднимаем флаг ссобытия
          gate_open(); //открываем ворота
          return;
        }
      }
    }

    void radioSetup() {          // настройка радио
      radio.begin();               // активировать модуль
      radio.setAutoAck(1);          // режим подтверждения приёма, 1
вкл 0 выкл
      radio.setRetries(0,  15);          //  (время  между  попыткой
достучаться, число попыток)
      radio.enableAckPayload();   // разрешить отсылку данных в ответ
на входящий сигнал
      radio.setPayloadSize(32);    // размер пакета, байт
      radio.openReadingPipe(1, 0xAABBCCDD11LL);    // хотим слушать
трубу 1
      radio.setChannel(CH_NUM);                // выбираем канал (в
котором нет шумов!)
      radio.setPALevel(SIG_POWER);              // уровень мощности
передатчика
      radio.setDataRate(SIG_SPEED);           // скорость обмена
      radio.powerUp();         // начать работу
      radio.startListening();  // начинаем слушать эфир, мы приёмный
модуль
    }
```