МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«Тольяттинский государственный университет»

Институт энергетики и электротехники

(наименование института полностью)

Кафедра « Промышленная электроника »

(наименование кафедры)

11.04.04. Электроника и наноэлектроника

(код и наименование направления подготовки)

Проектирование интеллектуальных систем зданий и сооружений

(направленность (профиль)

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему Система «Умный дом»

Студент А.С. Ковбасенко _____

(И.О. Фамилия) (личная подпись)

Научный руководитель А.В. Прядилов _____

(И.О. Фамилия) (личная подпись)

Руководитель программы к.т.н. доцент А.А. Шевцов _____

(ученая степень, звание, И.О.Фамилия) (личная подпись)

«____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н. доцент А.А. Шевцов _____

(ученая степень, звание, И.О. Фамилия) (личная подпись)

Тольятти 2019

Аннотация

Объем 77 с., 63 рис., 2 табл., 30 источников, 1 прил.

Цель данной диссертационной работы - исследование технологий и системы «Умного дома», проектируется и реализуется управляемый стенд «Умный дом» на базе печатных плат с микроконтроллерами. Данный стенд разработан с функционалом, позволяющим стенду автоматически управлять освещением, климат контролем, электросетью дома и охранным функционалом.

Стенд рассчитан на последующую модификацию и расширение за счет добавления новых модулей в созданную сеть за счет построения его в виде беспроводной децентрализованной системы. Кроме того разработанный стенд может получать данные с датчиков и осуществлять управление освещением, сигнализацией и электросетью как автоматически так и через радио управление и при этом не завесить от внешних источников питания.

В данной диссертационной работе проводится исследование существующих решений, а так же разработка стенда «умный дом» с децентрализованной беспроводной системой, с возможностью автоматически управлять освещением, электросетью, охранными функционалом и системой управления климат контролем умного дома.

Задачи, которые были решены в рамках данной диссертационной работы:

- анализ существующих систем и готовых решений;
- разработка стенда «Умный дом»;
- проектирование и разработка программы для контроля системы;
- изготовление и отладка макета

Практическая значимость. Так как для разработки данного стенда используются нестандартные решения, а так же совершенно новый программный комплекс разработанный стенд послужит в роли программно-аппаратного комплекса, на основе которого можно будет проводить обучение

студентов обучающихся как по направлению бакалавриата, так и по магистерскому направлению.

На основе данного стенда можно будет создать собственный проект умного дома для использования в частных домах и квартирах с целью повышения экономии электропитания, а так же осуществления автоматического и радиоуправления всеми системами умного дома. Данный стенд обладает возможностями легко расширяться, изменяется и адаптироваться под требуемую жилищную среду, что позволит использовать данный программно-аппаратный комплекс при проведении практических занятий в качестве лабораторного стенда.

Содержание

Аннотация 2

Введение 5

1 Обзор известных систем по типу умный дом и готовых решений 6

1.1 Постановка задачи 6

1.2 Анализ исходных данных и готовых решений 7

1.3 Анализ существующих на данный момент архитектур системы «Умный дом» 11

1.4 Анализ существующих контроллеров исполняющих роль управления системой «Умный дом» 17

2 Исследование и построение решения 22

2.1 Подбор аналогов оборудованию применяющего в системах типа «Умный дом» для разработки макета. 23

2.2 Выбор среды передачи данных 26

2.3 Выбор устройств на основе беспроводной технологий. 30

2.4 Сравнение модулей и выбор требующегося для построения стенда 33

2.5 Выбор ПО для программирования стенда «Умный дом» 35

2.6 Подключение настройка и первый запуск модулей Wi-fi связи ESP-01S Wemos D1 R1 mini 37

3 Построение стенда, разработка программного обеспечения, создание пульта управления и построение беспроводной сети 49

Заключение 74

Список используемой литературы 75

Приложение А 78

Листинг модуля создания сети и веб-сервера 78

Введение

На сегодняшний день наш мир представляет из себя довольно уютное место, где большинство повседневных задач автоматизированы или максимально упрощены. С каждый годом данная тенденция увеличивается. Современный человек усовершенствовал технологии автоматического и удалённого управления настолько, что данные технологии помогаю не только экономить время и деньги, а так же позволяют не беспокоиться о безопасности своего жилища, ведь зачем переживать когда технологии сами могут включить кондиционер в жару, включить освещение при входе в темную комнату или самостоятельно позвонить в полицию. Возрастающая популярность автоматизированных систем, такого типа как «умный дом», обуславливается стремлением человека к комфортной и удобной жизни. Дополнительным фактором привлекательности данных систем является безопасность, от противопожарных систем до сигнализации с дистанционным оповещением.

«Умный дом» на сегодняшний день представляет из себя аппаратно-программный комплекс современных инструментов разработанного с целью повышения уровня комфорта и жизни человека, а так как большинство процессов происходит автоматически, то это делает его актуальным для изучения и совершенствования.

1 Обзор известных систем по типу умный дом и готовых решений

Данный раздел создан для изучения существующих, на сегодняшний день аналогов, выполняющих такие же или схожие задачи. Будет проведен анализ дополнительных систем с целью выявления достоинств и недостатков.

Что по сути представляет из себя система «Умный дом»? Данная система является комплексом стандартов взаимодействующих с разного рода приборами в единой системе, либо несколько интегрированных в строение систем.

Системы бывают следующие:

- система автоматического управления микроклиматом;
- система безопасности;
- система электропитания;
- система управления водопровода;
- система связи;
- система удалённого управления;
- система голосового управления.

1.1 Постановка задачи

Основной целью данной диссертационной работы является создание стенда «Умный дом», который поможет в изучении данного направления.

Стенд будет разработан с целью изучение возможностей разработки, построения, расширения и модернизации такой технологии как «Умный дом» в различных вариациях его системы включая:

- системы, построенные с использованием силовой проводки;
- централизованные системы;
- радиошинные системы;
- децентрализованные системы.

Кроме того будет рассмотрен способ создания универсального пульта управления для систем умный дом, который можно будет настроить без

дополнительного программного обеспечения, и использовать его через любое цифровое устройство включая ПК, мобильный телефон и т.д.

1.2 Анализ исходных данных и готовых решений

Структура умного дома. Зачастую представляет из себя комбинацию основных управляющих систем контролируемых посредством центрального контроллера, либо при помощи отдельных модулей взаимодействующих между собой и выполняющих схожий процесс.

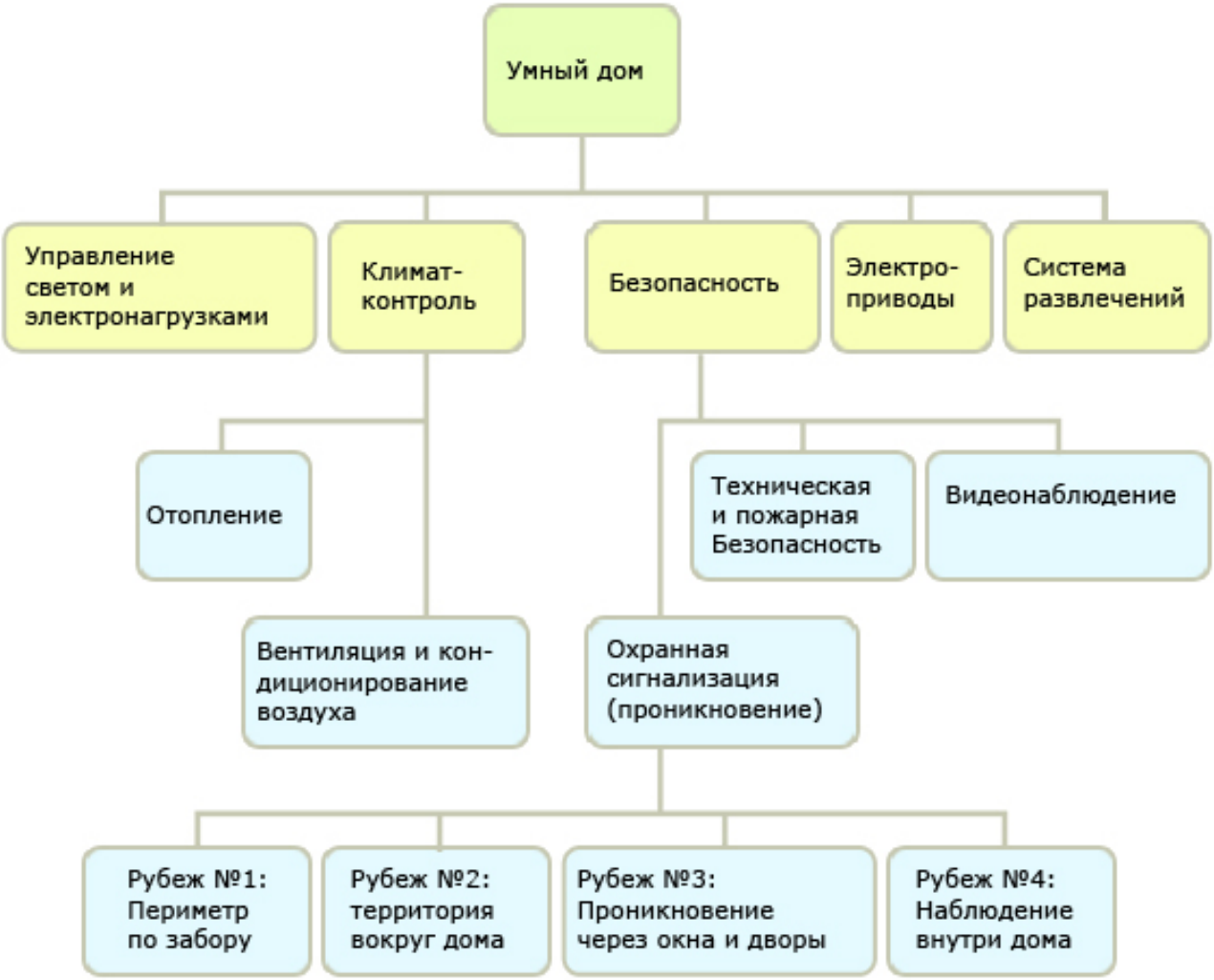

Рисунок 1 - Структура умного дома

Освещение и электросеть. Представляет из себя самый востребованный на сегодняшний день блок среди пользователей.

Данный блок помимо контроля отдельных «умных розеток» и использования органов управления освещением, производит контроль уровня освещённости в различных помещениях домах или сценариями освещённости в зависимости от уровня освещенности на улице либо временного периода.

Для каждого помещения дома чаще всего достаточно трёх уровней освещённости: минимальный, средний, максимальный, а так же функции выключения.

Рисунок 2 – Датчик освещения Z-wave и умная розетка Xiaomi

Климат-контроль. Данная система рассчитана на контроль температуры влажности и температуры внутри помещения в зависимости от пожелания владельца. Работает данная система по следующему принципу: пользователю, вместо ручной настройки каждого из отдельных органов управления для получения комфортного уровня климата, может лишь задать требуемые ему условия на пульте управления системой или на центральном контролере, все остальное система сделает сама. Так же данная система рассчитана на приспосабливание к различным изменениям климата, чтобы не беспокоить конечного пользователя. Иными словами пользователь данной системы выставляет требуемые настройки климата лишь раз, и система

самостоятельно используя свой функционал следит чтобы выставленные требования соблюдались, при изменении настроек пользователем, системмма так же реагирует и меняет свой приоритет.

Рисунок 3 – Датчик температуры и влажности Xiaomi

Безопасность. Данная система выполняет функционал обеспечения безопасности заданного периметра посредством внедрения трёх групп решений: систему видеонаблюдения периметра, защита от вторжения посторонних личностей, а так же предупреждения и ликвидации техногенных аварий. Система безопасности делит зоны собственной ответственности на рубежи: начиная от территории вокруг дома и заканчивая безопасностью внутри дома.

Рисунок 4 – Сирена Z-wave SE812

Электроприводы. Данный раздел существует с целью управления воротами, жалюзи, маркизами и другим оборудованием.

Рисунок 5 – Кран с электроприводом

Система развлечений. Данная система занимает последнее место так как на сегодняшний день пользователь не заинтересован в идее распределённого видео контента, так как сейчас для комфортного просмотра ТВ, кино и сериалов достаточно лишь доступа к интернету, а облачные технологии и сервисы по типу «NETFLIX» «AMEDIATEKA» и «YOUTUBE» позволяют просматривать интересный контент и вне дома.

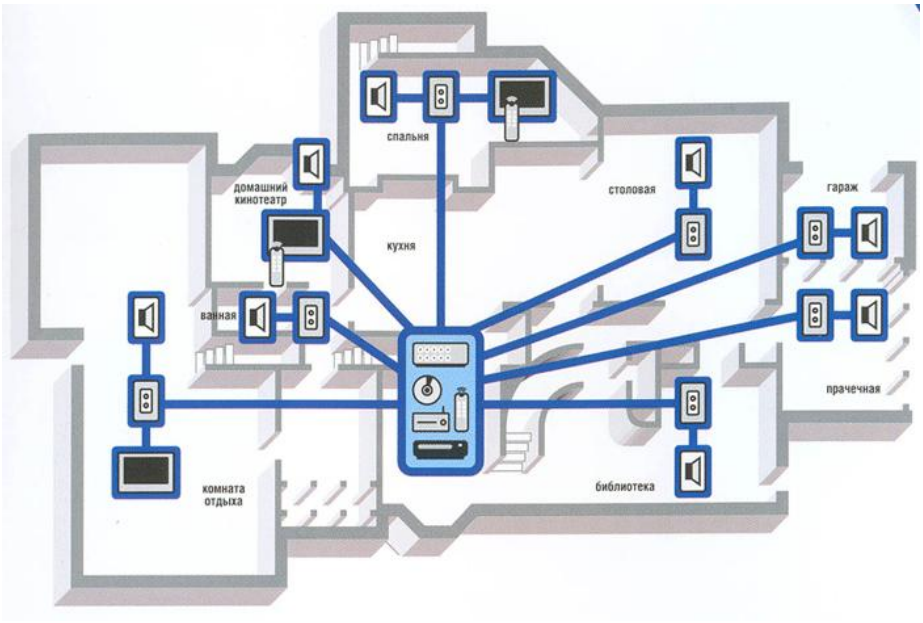

Рисунок 6- Схема системы Multiroom.

Погода и полив растений. Данные системы контроля используются в частных домах, цель данной установки следит за уровнем влажности в почве, а так же погодном состоянии и в случае изменения погодных условий реагировать на данные изменения. Как пример можем рассмотреть установку локальной метеостанции, это обеспечит конечного пользователя оперативной информацией о температуре, влажности, давлении, скорости и направлении ветра. Некоторые из таких систем могут осуществлять сбор статистики о количестве выпавших осадков за определенный период времени. В случае длительного отсутствия осадков автоматическая система полива по заданному расписанию польёт газон и растения.

Рисунок 7 – Таймер автополива

1.3 Анализ существующих на данный момент архитектур системы «Умный дом»

На сегодняшний день существует несколько типов архитектуры управления системой «Умный дом», данные архитектуры разбираются виды посредством системы управления.

Централизованная система состоит из программируемого центрального контроллера, к которому подключаются модули.

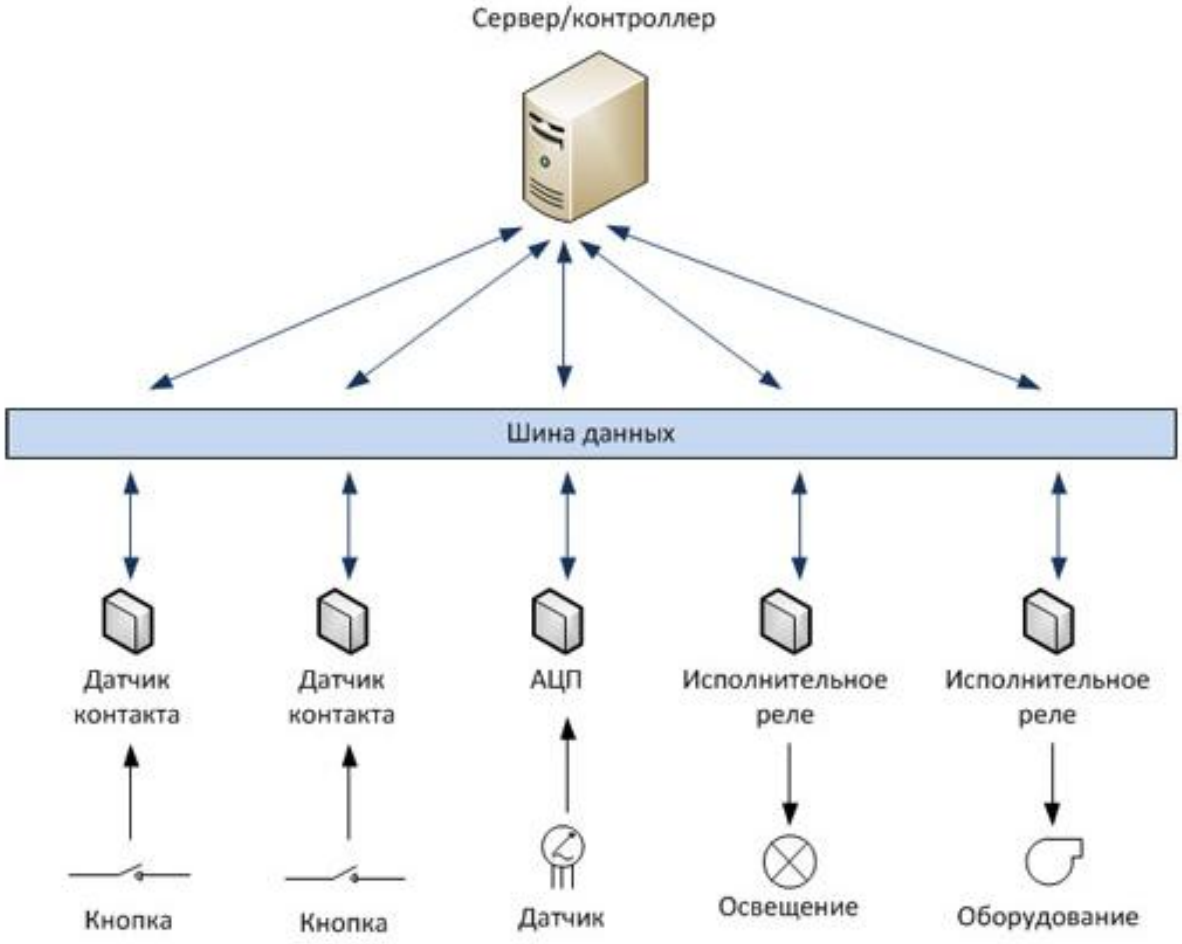

Рисунок 8- Графическое изображение централизованной архитектуры управления

Преимущества централизованных систем:

- централизованной системе строить сложные программы управления, завязанные на времени, температуре и состоянии пользователя. Центральный контролер обладает достаточной производительностью и обладает информацией о подключённых к нему модулям, что позволяет управлять всей системой в едином интерфейсе.

- централизованные системы обладают большей скоростью обработки информации, так как сбором информации с модулей, центральный контроллер занимается, не прибегая к модульной обработке;

- модули используемые в таких системах компактны, дешевы и имеют простую техническую реализацию, что дает подключить к контроллеру практически любое оборудование.

Недостатки централизованных систем:

- высокая цена центрального контроллера;

- ненадежность. При выходе из строя центрального контроллера, вся система перестанет функционировать;

- человеческий фактор. Если с программистом что производил наладку и программирование центрального контакт утерян, то в случае необходимости перепрограммирования придется заново писать всю программу.

Децентрализованная система управления «умным домом», устройства не зависят друг от друга так как каждый модуль несет в себе микропроцессор с энергонезависимой памятью. Данная архитектура строит системы на шине.

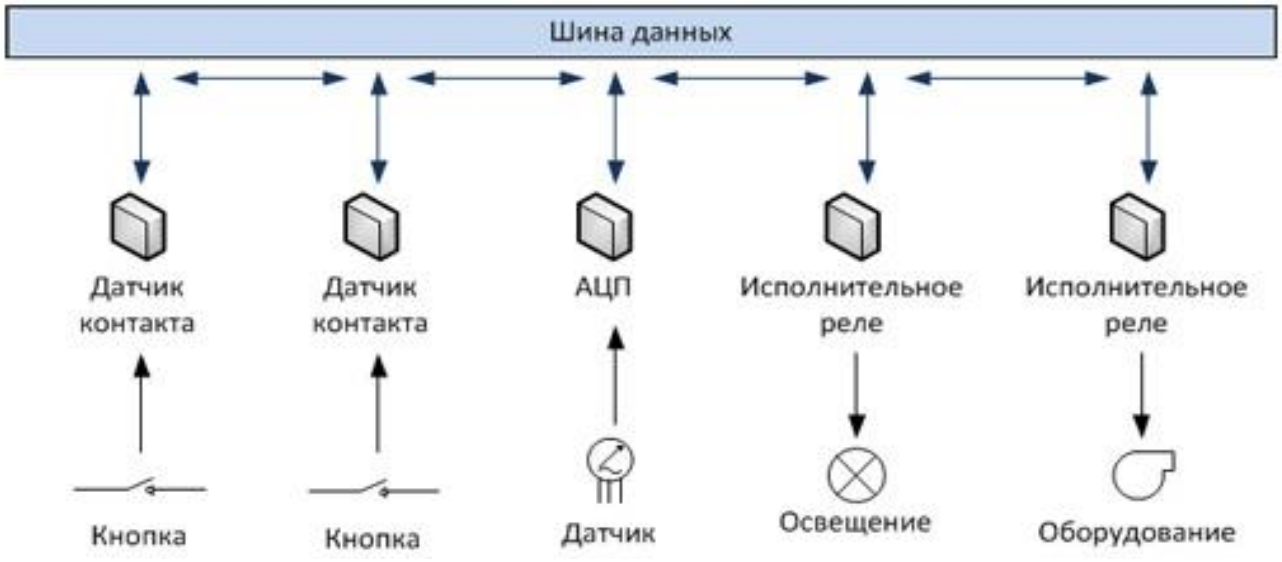

Рисунок 9 - Графическое изображение централизованной архитектуры управления

Достоинства децентрализованных систем:

- надежность. Из-за отсутствия центрального контроллера, выход из строя одного или нескольких модулей, существенно не повлияет на работу системы в целом. Из чего следует, децентрализованная система обладают повышенной надежностью.

- децентрализованные системы просты в расширении. На имеющуюся шину легко добавляются новые модули, поддерживающие протокол передачи данных используемой шины.

Недостатки децентрализованных систем:

- датчики, имеют собственные контроллеры обработки данных, из-за чего имеют высокую стоимость, а так же становятся технически сложными.

- скорость работы системы ниже из-за обработки данных в разных модулях.

Облачная система управления «умным домом» базируется на управлении всей системой при помощи облачных технологий, с возможностью подключения пользователя в любой момент и настройкой изменений по желанию. Основной принцип данной технологии заключается в том что данная система не имеет как такого центрального контроллера, а вся работа с данными ложится на дата центры.

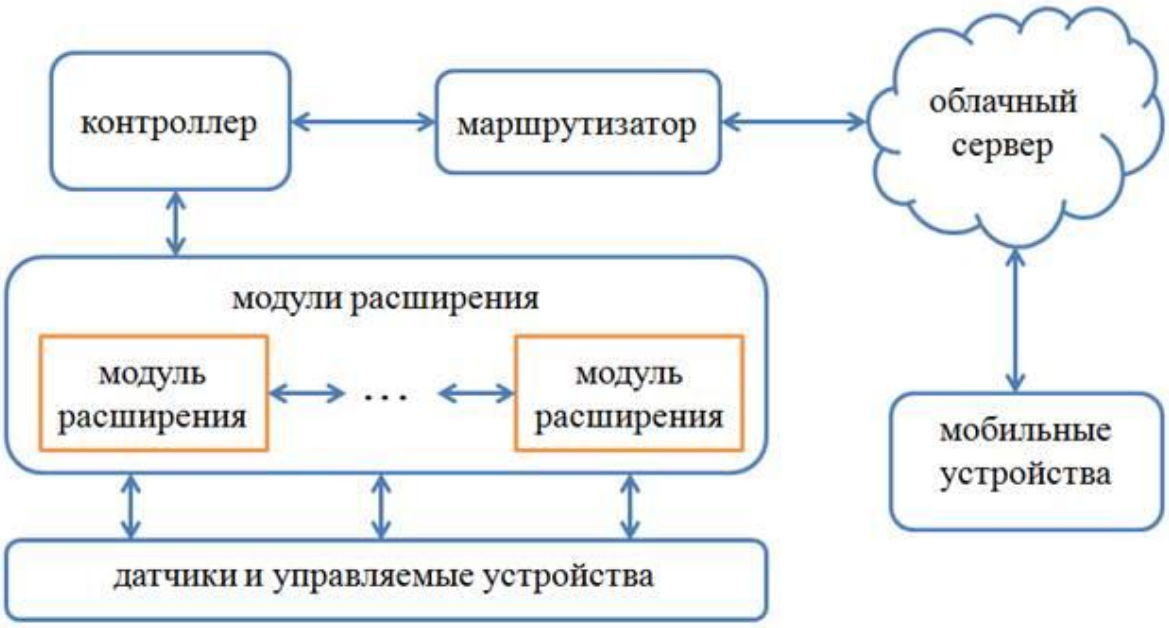

Рисунок 10 - Графическое изображение облачной архитектуры управления

Достоинства облачной системы управления «Умный дом»:

- так как у системы нет центрального процессора то и затраты энергопотребления будут в разы меньше чем при использовании централизованной системы;

- так как в системе нет сложных в техническом плане устройств цена данной системы в разы ниже чем у предшествующих систем;

- при высокой скорости передачи данных посредством использования интернета работа датчиков и управляющих устройств будем практически мгновенной.

Недостатки облачной системы управления:

- большинство современных компаний предоставляющих использование своих дата центров могут взимать ежемесячную плату, что в итоге может превысить цену конечного продукта;

- если в работе интернета случаются перебои то они значительно влияют на работу всей системы в целом;

- если вы используете слабо защищенный канал для передачи данных то он может подвергнутся взлому, что может дать злоумышленнику не только все ваши личные данные, но так же ваше расписание дел и полный контроль над всеми системами дома что может закончиться печально.

Радиошинная система управления умным домом базируется на использовании радио-модулей. Данная технология разработана на основе современных беспроводных технологий, представляет из себя идеальную систему для встраивания в уже готовый объект инфраструктуры а так же в небольшое строение.

Принцип работы осуществляется следующим образом, в помещение устанавливается оборудование двух типов радиопередающее и радиоприемное. Радиопередающее устройство отправляет команду на все исполняющие устройства, но её исполнением займется лишь то устройство что настроено под отправленную команду.

Достоинства радиошинной системы управления «Умный дом»:

- все передатчики получают питание от батареи, что позволяет устанавливать их в любом помещении даже в том где нет электропроводки;

- для монтажа и демонтажа системы не требуется группы специалистов, что позволит сэкономить деньги на установке;

- для установки данной системы не требуется проводить разбор стен и выполнять сложных перестановок, достаточно просто закрепить оборудование, так как для передачи данных не используются провода;
- при переезде на новое место данную систему можно забрать с собой;
- низкая цена в сравнении с предыдущими аналогами.

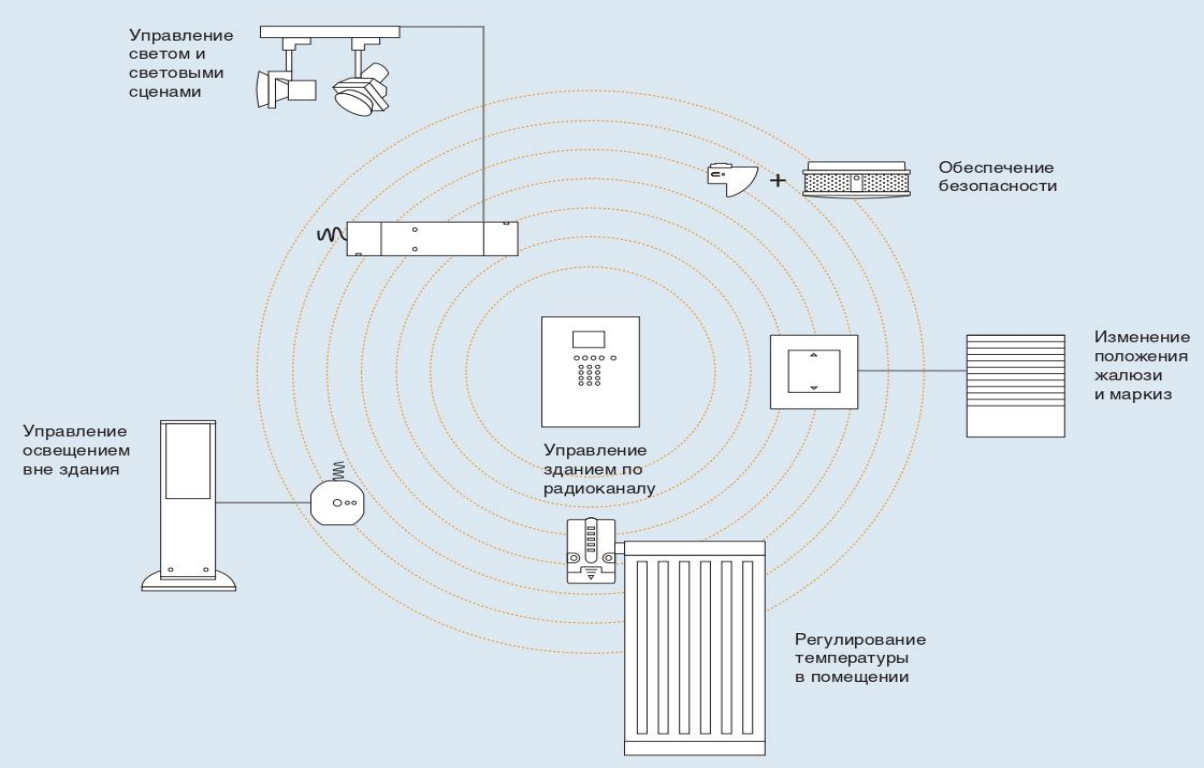

Рисунок 11 - Графическое изображение радиошинной архитектуры управления

Недостатки радиошинной системы управления «Умный дом»:

- система не имеет настроек, что плохо для начинающего пользователя, так как ему придется самостоятельно разбираться как проводить настройку, либо вызывать специалиста;
- настройка позволяет записать до 30 команд на каждое управляющее устройство что может быть малым количеством для отдельных пользователей;
- на данный момент разработкой и установкой данной системы занимается лишь одна контора Gira, что приводит нас к повышенной цене на всё оборудование данной технологии.

1.4 Анализ существующих контроллеров исполняющих роль управления системой «Умный дом»

Существует множество контроллеров разных производителей, которые могут исполнять роль управления системой «Умный дом». Они имеют разные технологии передачи данных. Рассмотрим некоторые из них.

LanDriver SIDER2 – представляет из себя контроллер применяющийся в универсальных системах управления "умный дом" компании LanDriver, осуществляет контроль всех электронных устройств здания. Данный контроллер применяется в централизованных системах управления, для передачи данных использует между устройствами использует протокол передачи данных Modbus основанный на архитектуре master-slave. Обладает объемом памяти равным 4МБ, и может программироваться через интерфейс программы LanDrive Configurator Pro. Кроме стандартных портов имеет возможность установки передатчиков использующих протокол передачи данных ZigBee.

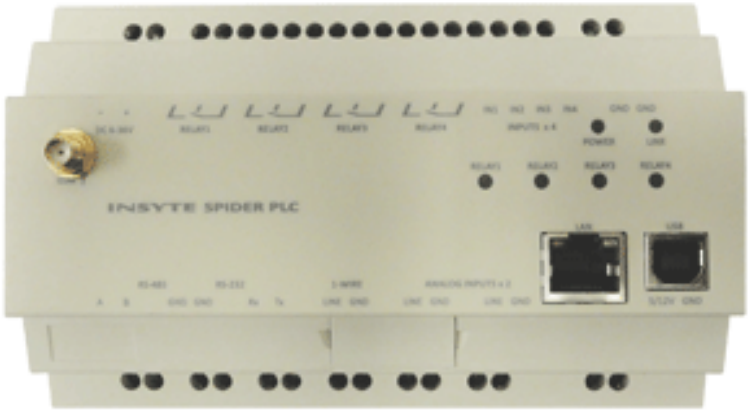

Рисунок 12 - Центральный управляющий контроллер «SPIDER2», используемый в системе LanDriver

EIB/KNX – Данная система использует для работы децентрализованную систему управления то есть управление осуществляется в пределах устройств. Для передачи данных между устройствами EIB/KNX применяется протокол передачи данных EIB основанный на семиуровневой сетевой модели OSI.

Рисунок 13 - Исполняющие датчики системы EIB/KNX оснащенные собственными контроллерами KNX

Netlinx NI-3100 – Контроллер рассчитанный для работы с программно-аппаратными средства удаленного управления, системой видеонаблюдения, медиа системой а так же широким спектром датчиков. Передача данных данного контроллера осуществляется через протоколы Ethernet и axlink (специальный протокол передачи данных использующийся компанией AMX), микроконтроллер обладает 64Мб памяти и 1 Гб flash памяти что позволяет настраивать его под различные программы и выполнение сложных команд.

Рисунок 14 - Центральный управляющий контроллер «NI-700», используемый в системе AMX

Z-wave Plus центральный контроллер использующийся в технологиях умного дома от компании Z-wave использующий для передачи данных и управления исполняющими датчиками технологию связи Wi-Fi. Так как данный контроллер оснащен дополнительным аккумулятором контроллер можно устанавливать в любом помещении даже с отсутствующим электропитанием. Для работы используется система OpenWRT Linux, сам модуль оснащен 512 Мб оперативной память и до двух гигабайт flash памяти что позволяет настраивать его под различные нужды.

Рисунок 15 - Центральный управляющий контроллер «Z-wave Plus», используемый в системе Z-wave

UniPing server solution v3/SMS контроллер семейства устройств, разработанные отечественной компанией «Alentis Electronics» для мониторинга окружающей среды. Данный контроллер рассчитан для работы в централизованных системах типа «Умный дом» и может применяться как для удаленного контроля и мониторинга устройств в доме, так и в офисе. Обладает специальным функционалом позволяющим сообщать пользователю по средствам SMS и Email о состоянии дома. Данный контроллер использует протокол передачи данных Ethernet.

Рисунок 16 - Устройство удалённого мониторинга датчиков по сети Ethernet/Internet «UniPing server solution v3» включенная в структуру умного дома.

Микроконтроллер Home Center 2 – работает в системах умный дом работающих на основе облачных технологий, пользователь может контролировать состояние дома из любой точки мира. Контроллер обладает мощным процессором компании intel, 1 Гб оперативной памяти, 2 Гб flash память и встроенным жестким диском на два гигабайта для создания бекапа. Данный контроллер использует протоколы Ethernet (TCP/IP) и Wi-Fi для работы с удаленными устройствами. Данный контроллер постоянно сканирует систему, а при необходимости информирует пользователя о различных изменениях в работе системы.

Рисунок 17 - Центральный управляющий контроллер «Home Center 2», используемый в системе Fibaro и Z-wave

Из рассмотренных контроллеров самым простым и многофункциональным является Z-wave, так как обладает простым в настройке и может поддерживать работу с различными компонентами компании Z-wave. При этом обладает относительно невысокой ценой в сравнении с остальными устройствами. При этом контроллер SPIDER2 представляет из себя наиболее безопасное решение так как для взаимодействия со своими модулями может поддерживать протокол передачи данных ZigBee который на данный момент является самым защищенным.

Так же стоит обратить внимание на датчики EIB/KNX которые поддерживаю работу друг с другом без использования центрального контроллера, но данная работа выполняется за счет того что каждый модуль обеспечен собственным контроллером, позволяющим датчикам считывать информацию после чего практически сразу реагировать на любые изменения.

Самым универсальным решением является контроллер Home Center так как он позволяет использовать облачные технологии для контроля за состоянием всех систем своего жилища при этом находясь в любой точке мира, а так же имеет поддержку систем Z-wave.

2 Исследование и построение решения

В данном разделе будет проведен анализ рассмотренных в предыдущем разделе известных технологий и решений для систем «Умного дом» после чего на основе полученных данных будет разработан стенд, удовлетворяющий всем требованиям современных решений. Будет произведен подбор аналогов для существующих датчиков и центральных контроллеров, произведена разработка системы умный дом, написано программное обеспечение, а так же выбрана среда передачи данных.

На начальной стадии проектирования определим состав системы «Умного дома», исходя из поставленных задач. Так как система должна осуществлять функции управления освещением (с учетом движения объектов в зоне системы), климат контролем, то в схему включим датчики движения, датчик освещенности, термодатчик, привод жалюзи климат-контроля, блоки управления процессами включения-выключения света, вентиляции на основе показания датчиков, система коммуникаций устройств между собой, удаленная система внешнего управления.

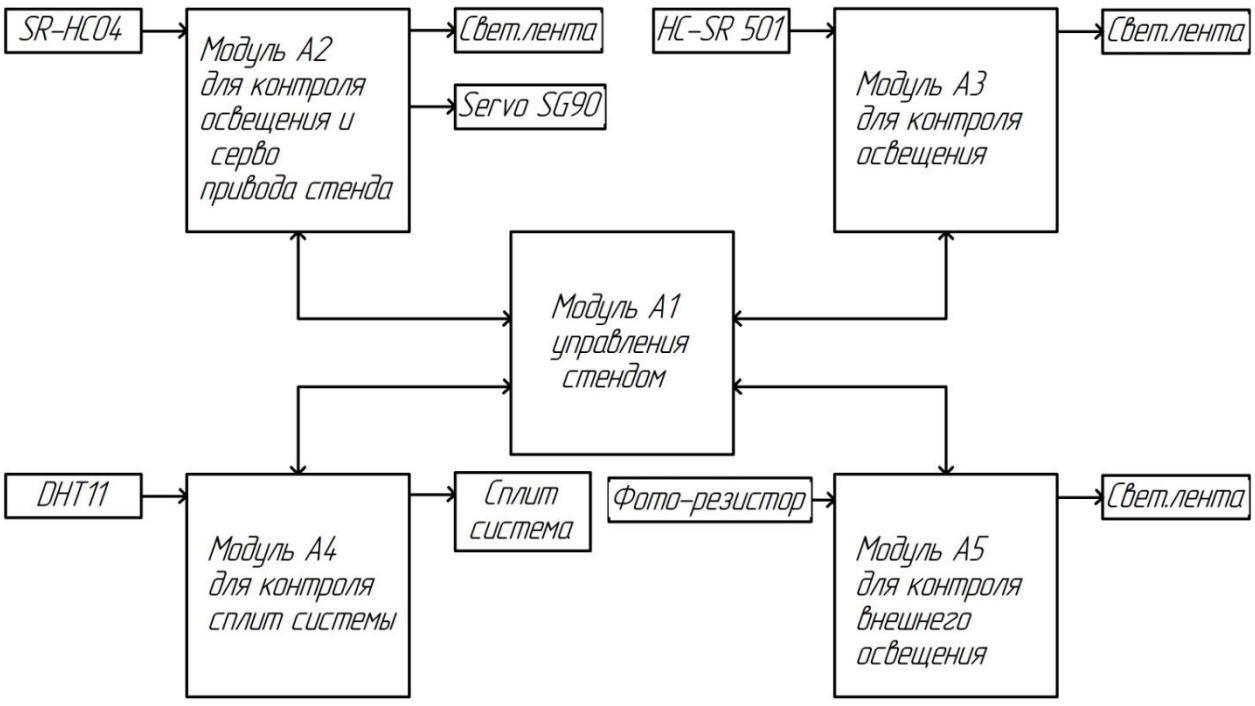

Рисунок 18 - Общая структурная схема системы

2.1 Подбор аналогов оборудованию применяющего в системах типа «Умный дом» для разработки макета.

В системах типа «Умный дом» зачастую используются дорогостоящие датчики, отслеживающие передвижение пользователя, температуру и влажность в помещениях, а так же уровень освещения внутри и снаружи дома.

Датчик освещённости – представляет из себя устройство, позволяющее работать системе автоматического управления освещения, и подстраивается под изменения состояния освещенности пространства. Данный датчик посылает различные типы сигналов в зависимости от уровня освещения на исполняющий элемент который в зависимости от полученных сигналов делает различные действия, включает или выключает свет. Аналогом данному устройству в макете послужит более дешевый датчик освещённости используемый совместно с платой Ардуино.

Рисунок 19 - Датчик освещенности и его аналог на базе Ардуино

Датчик движения - это приборы, работающие на основе инфракрасного излучения, которое излучают все живые существа, которые обнаруживают перемещение объектов, и как следствие, замыкают цепь и свет включается. Аналогом данному устройству в макете послужит более дешевые датчики движения используемые совместно с платой Ардуино.

Рисунок 20 - Датчик движения и его аналог на базе Ардуино

Климат контроль - при использовании блока климат-контроля владелец дома один раз выставляет желаемые параметры микроклимата в помещениях, и после этого система управления сама следит за работой всех инженерных подсистем, отдавая приоритет тому оборудованию, чья работа в данный момент более выгодна пользователю. Аналогом данному устройству в макете послужит более дешевые датчики температуры используемые совместно с платой Ардуино.

Рисунок 21 - датчик температуры и его налог на базе Ардуино

Электроприводы - данный раздел существует с целью управления воротами, жалюзи, маркизами и другим оборудованием. Аналогом данному устройству в макете послужит более дешевые сервоприводы используемые совместно с платой Ардуино.

Рисунок 22 - сервопривод, применяющийся в системах умный дом и его аналог на базе Ардуино

Пульт управления системой умный дом - устройство, позволяющее контролировать все системы Умного дома. Аналогом данному устройству послужит телефон со специально разработанным пультом управления, либо со специальным программным обеспечением.

Рисунок 23 - пульт управления умным домом

Как можно заметить в данной главе все аналоги подбираются так чтобы быть совместимыми с платами компании Ардуино, это делается для того чтобы студенты которые начнут работу с разрабатываемым стендом не испытывали проблем с настройкой и подключением модулей для стенда.

2.2 Выбор среды передачи данных

Из чего мы переходим ко второму пункту рассуждения, какая среда передачи данных будет применяться в стенде? Существуют:

- физическая среда передачи данных;
- беспроводная среда передачи данных.

Физическая среда передачи данных. Для данной среды характерно использование кабелей или проводов, в основном на медной жиле, для передачи информации. Для передачи данных в физической среде применяются следующие типы кабелей:

Коаксиальный кабель – характеризующийся одной прямой медной жилой выполняющей роль внутреннего проводника, окруженной изоляцией из сплошного или полувоздушного диэлектрика, внешнего проводника представляющего из себя оплетку из фольги, покрытой слоем алюминиевой пленки, изоляционную оболочку изготавливают из полиэтилена устойчивого к ультрафиолетовому излучению. Длина передачи данных через такой кабель зависит от типа кабеля и может быть от 100 до 500 метров, скорость передачи данных достигает до 10 Мбит на сегодняшний день данный кабель является устаревшим и используется для передачи данных между аналоговыми устройствами.

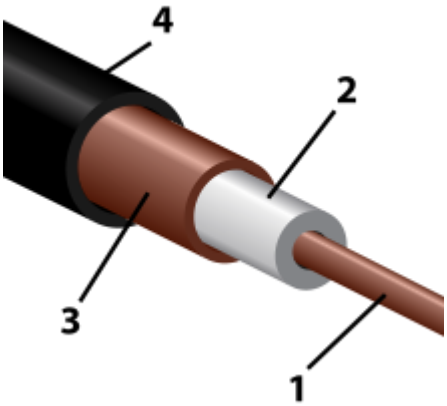

Рисунок 24 – Коаксиальный кабель: 1 — внутренний проводник, 2 — изоляция, 3 — внешний проводник, 4 — оболочка

Витая пара – характеризуется одной или нескольких пар изолированного проводника покрытых пластиковой оболочкой такой тип кабеля

характеризуется высокой скоростью передачи данных в зависимости от типа кабеля может обладать максимальной скоростью передачи данных до 1Гбит и длиной до 1 километра, является среднем типом кабеля, является весьма популярным среди интернет провайдеров. На данный момент времени на данном типе кабеля строятся локальные сети и сети типа интернет;

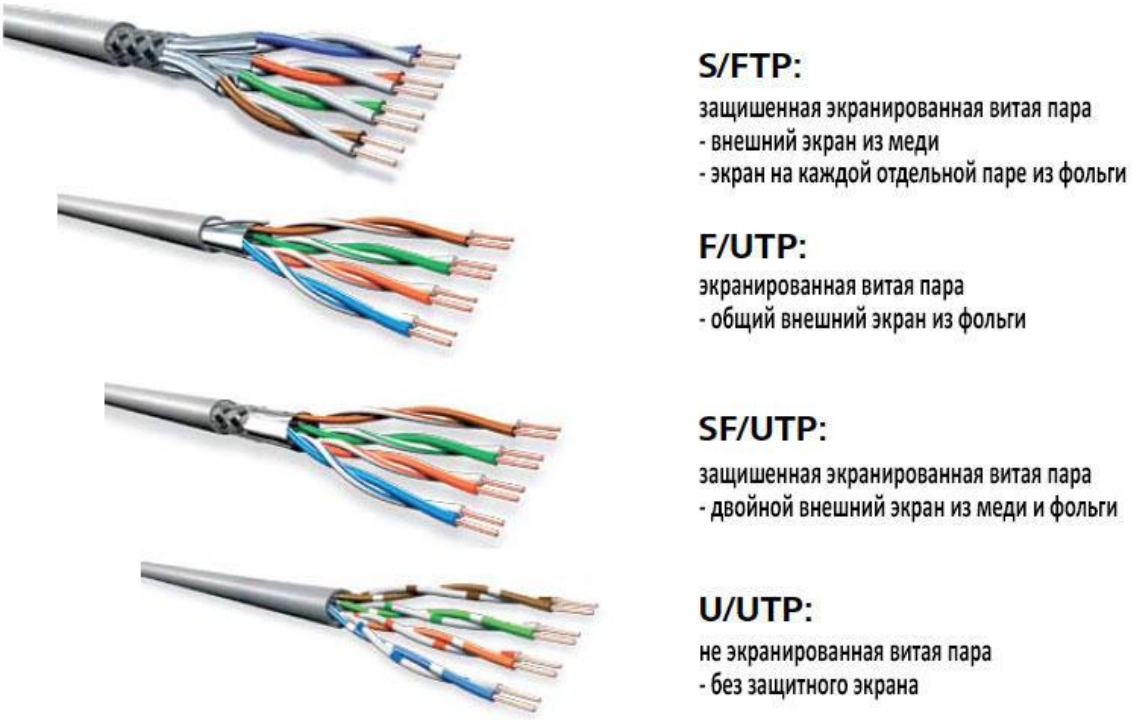

Рисунок 25 – Типы витых пар

Волоконно-оптический кабель – использующий тип передачи данных на основе волоконных световодов, предназначенный для передачи оптических сигналов в линиях связи, в виде фотонов, со скоростью меньшей скорости света из-за отсутствия прямолинейного движения.

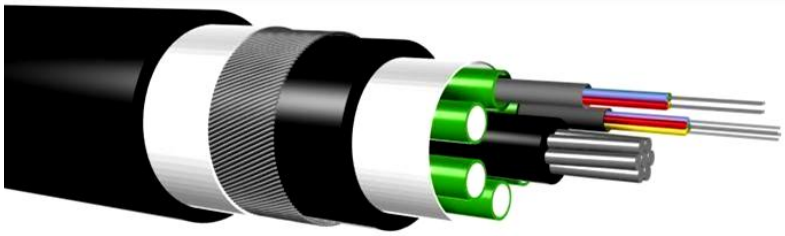

Рисунок 26 - Волоконно-оптический кабель

Беспроводная среда передачи данных, это среда в которой для передачи данных используется нефизический тип передачи данных, иными словами

вместо кабелей, проводов и т.д. применяются радиоволны, микроволновое излучение, инфракрасное излучение и т.п.

Для передачи данных и построения сети можно использовать следующие типы связи:

- инфракрасное излучение;
- Bluetooth;
- Wi-Fi.

Инфракрасное излучение представляет из себя электромагнитное излучение, находящееся в радиусе спектральной области между красным светом и коротковолновым радиоизлучением. Данный тип связи состоит из одного передатчика и одного приемника, связь прервется если между двумя данными устройствами появится препятствие.

Bluetooth – принцип работы данного протокола связи основан на радиоволн в ISM диапазоне, имеет довольно большое распространение и обладает следующими характеристиками радиус передачи данных до 100 метров, и скоость передачи данных до 24Мбит.

Свое название Bluetooth получил в честь короля дани Харольда I синезубого, объединившего враждовавшие датские кланы, по сути своей данная технология делает нечто схожее так как сама технология объединяет между собой устройства.

Wi-Fi - технология беспроводной локальной сети с устройствами на основе стандартов связи для коммуникации в беспроводной локальной сетевой зоне частотных диапазонов 0,9; 2,4; 3,6; 5 и 60 ГГц. Состоит из точки доступа и клиента, не может иметь ни одного ни другого менее чем в одном количестве, скорость и передача данных разнится в зависимости от оборудовании применяющегося в построения такой сети.

Выбор сети – из всех рассмотренных ранее технологий следует выбрать одну для построения сети на основе среды передачи данных.

Рассмотрим преимущества и недостатки рассмотренных ранее сред передачи данных.

Физическая среда передачи данных разделяются на два типа построения структурированные кабельные системы и простые неструктурированные кабельные системы.

Структурированная кабельная система обладает следующими плюсами и минусами.

Плюсы:

- единая система для любого типа данных и приложений;
- расширяемость, легкость переконфигурации;
- документация на каждый узел системы;
- гарантия на систему до 20 лет эксплуатации;
- очень высокая надежность.

Минусы:

- высокая стоимость проектирования и инсталляции.

Простая кабельная система обладает следующими преимуществами и недостатками.

Плюсы:

- низкая стоимость монтажа по сравнению с СКС;
- сравнительно высокая скорость монтажа;
- высокая надежность.

Минусы:

- небольшая гарантия на систему;
- сложность расширения системы, дополнительные затраты на расширение.

Беспроводная среда передачи обладает следующими плюсами и недостатками:

Плюсы:

- простота и скорость развертывания сети;
- низкая стоимость развертывания;
- отсутствие проводов на рабочем месте.

Минусы:

- скорость передачи делится между всеми устройствами Wi-Fi в пределах обслуживания их одной и той же точкой доступа.

- малая надежность в сравнении с СКС;

- при плохой настройке обладает весьма слабыми показателями защиты;

- наличие стен и иных препятствии могут как незаметно, так и довольно сильно снизить скорость передачи данных.

Рассмотрев все плюсы и минусы данных сред передачи данных мы остановимся на беспроводной среде передачи данных, так как используя её мы сможем обеспечить обмен информацией между датчиками и исполняющими печатными платами и модулями не используя проводов. Из чего получается что каждый модуль можно будет разместить отдельно друг от друга на расстоянии до 100 метров не проводя монтажа кабеля, а так же не затрачивая лишние средства на новый кабель для увеличения сети, так же при построении беспроводной сети при выходе из строя какого либо элемента вся остальная сеть продолжает функционирование в прежнем режиме. Так же все элементы для построения беспроводной сети выходят по цене дешевле чем при построении проводной среды передачи данных.

2.3 Выбор устройств на основе беспроводной технологий.

Для передачи информации по беспроводной сети можно использовать отдельные типы модулей, использующие такие протоколы как Bluetooth, IEEE 802.11 или инфракрасный порт.

Bluetooth модуль HC-06 использует протокол Bluetooth 4.0 для передачи данных, данный протокол характеризуется следующими характеристиками низкое энергопотребление, скорость передачи данных до 1Мбит в секунду, скорость подключения менее чем 5 миллисекунд и радиус работы до 100 метров. Для работы требуется плата Ардуино, в то же время нужды в установке дополнительных библиотек ставить, отдельно не требуется. Для взаимодействия с данным модулем потребуется оборудование

поддерживающее протокол Bluetooth 4.0, а так же дополнительного ПО что позволит взаимодействовать с модулем HC-06.

Рисунок 27 – Bluetooth модуль HC-06

Wi-Fi модули на основе микроконтроллера ESP8266 имеет следующие характеристики поддерживает протокол IEEE 802.11, что позволяет использовать современные протоколы технологии Wi-Fi, для стабильной работы в сети требуется от 2,2 до 3,6 вольт, так как сам микроконтроллер не обладает встроенной памятью некоторые модули снабжены дополнительной flash памятью от 1МБ. Поддерживает работу в трех режимах клиента, подключающегося к уже развернутым сетям на основе Wi-Fi, точки доступа позволяющей разворачивать собственные сети на основе wi-fi, а так же совместный режим, позволяющий два предыдущих режима. Для настройки работы модулей с данным процессором не требуется наличие платы Ардуино, так как они сами по себе являются самодостаточными, но для первоначальной настройки требуется использование специальных переходников UART. Единственным недостатком данного модуля является требование наличия специальной библиотеки.

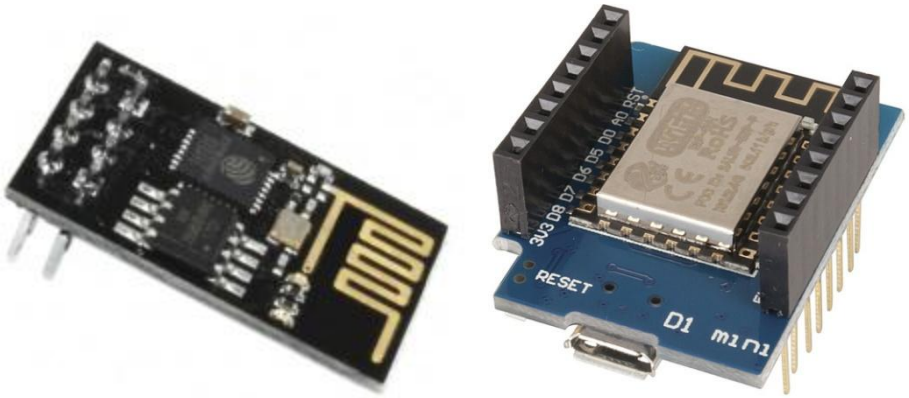

Рисунок 28 – Модули связи Wi-Fi ESP-01S и Wemos D1 mini построенных на микроконтроллере ESP-8266

IR – модуль работает на основе инфракрасного излучения, что значит между модулями поддерживающими передачу данных по инфракрасному порту не должно быть препятствий, так же для подключения и настройки данного модуля к на плате Ардуино требуется установка специальных библиотек.

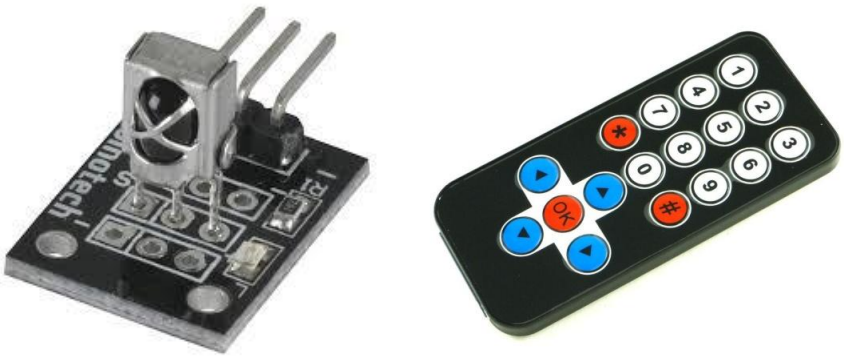

Рисунок 29 – IR модуль и пульт управления инфракрасного модуля

Отдельного внимания заслуживают модули ZigBee, применяющиеся на данный момент в технологиях для умного дома фирмы Xiaomi, использующие для работы стандартом IEEE 802.15.4. ZigBee и IEEE 802.15.4. Данная спецификация ориентирована на приложения, требующие гарантированной безопасной передачи данных при относительно небольших скоростях и возможности длительной работы сетевых устройств от автономных источников питания. Для работы данного модуля требуется напряжение от 2,8 до 3,4 В, а

так же в режиме приема 45 мА и 50 мА в режиме передачи. При работе в режиме энергосбережения 0,01 мА. Для работы данного модуля не требуется наличие платы Ардуино, он является самостоятельным, но в случае если потребуется данное подключение требуются дополнительные переходники.

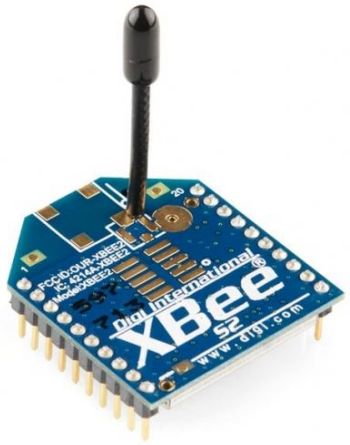

Рисунок 30 - Модуль XBee S2

2.4 Сравнение модулей и выбор требующегося для построения стенда

Таблица 1 – сравнение плюсов и минусов модулей передачи данных.

№	Название модуля	Скорость передачи данных	Возможность построения сети для передачи данных	Надежность и устойчивость сети	Энергопотребление	Цена Aliexpress
1	HC-06	До 1 Мбит	В сети может быть до 7 устройств	Сеть легко можно взломать, так как она не защищается при помощи пароля	Потребление напряжения в 3.0 В Среднее потребления тока 30мА	193,99 р
2	ESP-01S	До 13 Мбит	Может создавать сеть содержащую до 20 пользователей	При правильной настройке сети взлом сети буде затруднителен, т.к. сеть можно скрыть, и защитить паролем	3.3 В 220 мА	86,37 р

Продолжение таблицы 1

№	Название модуля	Скорость передачи данных	Возможность построения сети для передачи данных	Надежность и устойчивость сети	Энергопотребление	Цена Aliexpress
3	Wemos D1 R1 mini	До 13 Мбит	Может создавать и поддерживать сеть содержащую до 20 пользоватей	Имеет высокую защиту	3.6 В 70-240мА	90,35 р
4	IR	нет	невозможно	Защита слабая	5 В 1,05 мА	28,57 р
5	XBee	250 кБайт	64 пользователя на канал	Высокий уровень защиты обеспечивающийся шифрованием протокола ZigBee	3.3 В 45 мА	1 527,97 р

Проанализировав все рассмотренные ранее варианты мы остановим свое внимание на двух модулях, ESP-01S и D1 R1 Mini, так как они поддерживают совместную работу с платами Arduino, совместны с датчиками и модулями от самой платформы Arduino, а так же поддерживается специализированным программным обеспечением, выбранным нами. Единственным недостатком считается высокое потребление электрического тока, что заметно снизит показатели продолжительности работы без дополнительной подзарядки в сравнение с отдельными модулями из данного списка.

Вторая причина почему выбор пал на данные модули это их цена, в то время как для создания и построения сети на 3-5 модулях мы потратим чуть больше 500 рублей, модули Xbee обойдутся нам свыше полутора тысяч за одну модель.

Так же стоит отметить что данный тип модулей имеет собственный слот памяти в 4МБ, что позволит записывать программный код, данное действие позволит нам построить тип системы «Умный дом» на радиошине.

2.5 Выбор ПО для программирования стенда «Умный дом»

Построение кода применяющегося в стенде будет на языке C#, а так же Визуальные языки программирования FBD и Ladder, с помощью которых пишется программа, используются для программирования практически всех логических реле, и части промышленных контроллеров во всем мире. Так же было принято решение не использовать в коде переменную delay, так как её применение останавливает выполнение всего кода на заданный нами период времени, что приведет полной остановке кода, которая может привести к лишним задержкам, а так же вызвать сбои в работе кода.

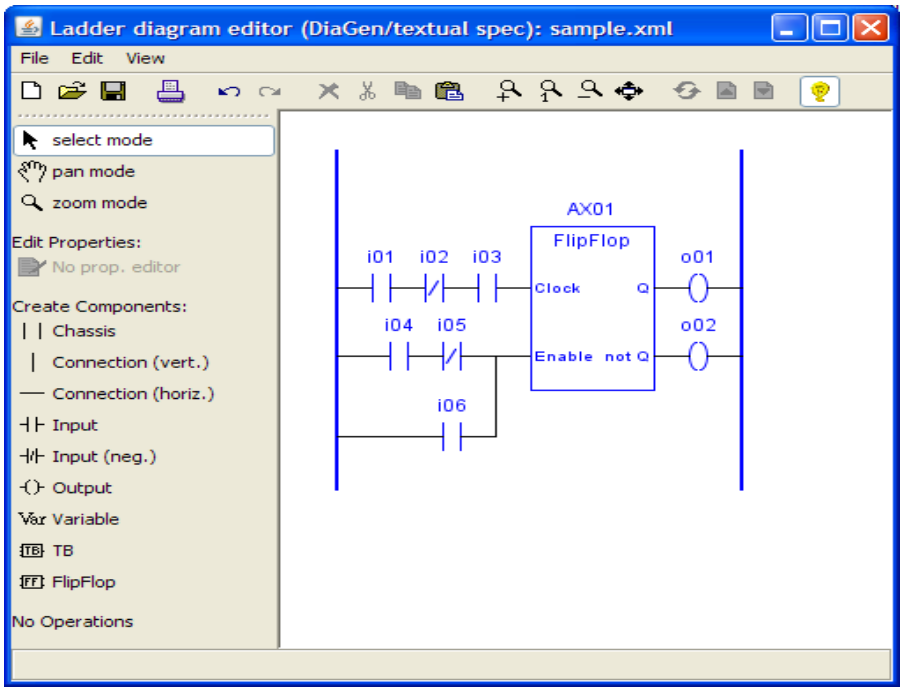

Рисунок 31 – Применение языка Ladder

В качестве среды программирования для создания управляющего кода стенда были выбраны 2 программы: FLProg, позволяющая использовать визуальные языки программирования FBD и Ladder, а так же Arduino IDE позволяющая конвертировать написанный нами код в программный язык и загружать его в плату для выполнения действий заложенных нами в программу.

Как итог к чему это приведет?

Для построения программы с применением данного программного обеспечения мы сможем легко построить сложную программу, зная лишь азы

данного языка, работа такой программы не будет прерываться, а так же данное ПО позволит настроить взаимодействие между несколькими печатными платами, через беспроводную среду передачи данных.

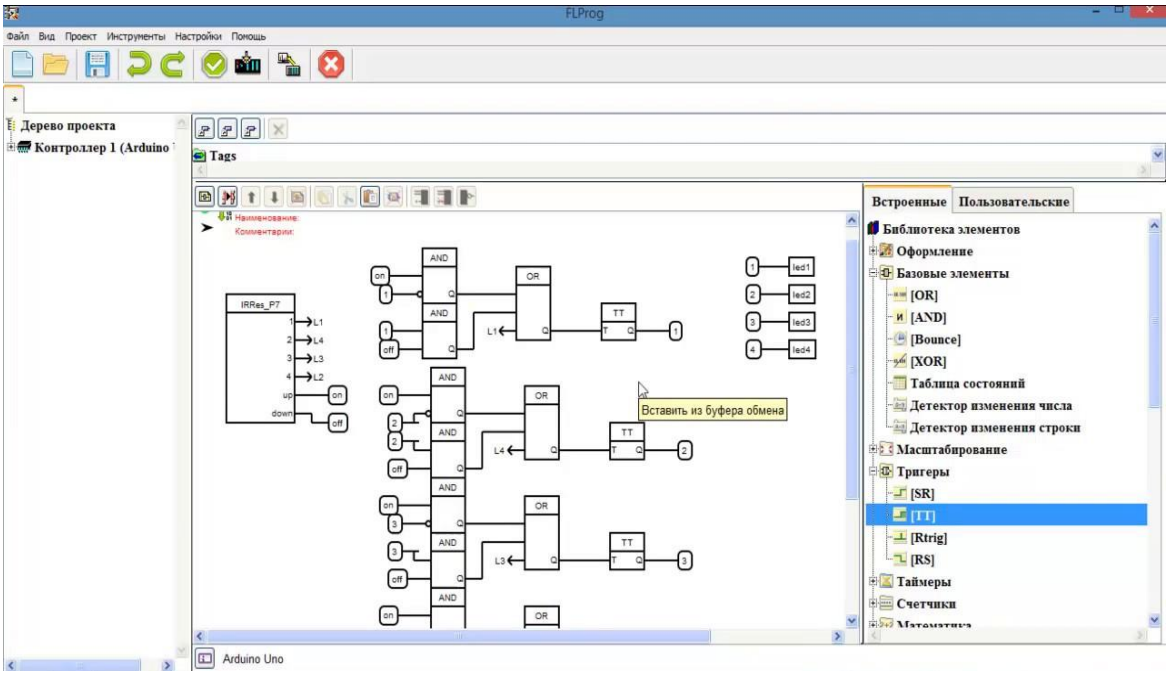

Рисунок 32 – Интерфейс программы FLProg

Рисунок 33 – Интерфейс программы Arduino IDE

2.6 Подключение настройка и первый запуск модулей Wi-fi связи ESP-01S Wemos D1 R1 mini

Для настройки модуля Wemos D1 R1 mini нам не требуется платы Ардуино и UART модуля, так как данная плата оснащена mini USB а так же имеет чип CH340G который позволяет использоваться USB порт в виде Com-порта.

Подключение осуществляется точно так же как у обычной платы Ардуино, а именно через USB порт.

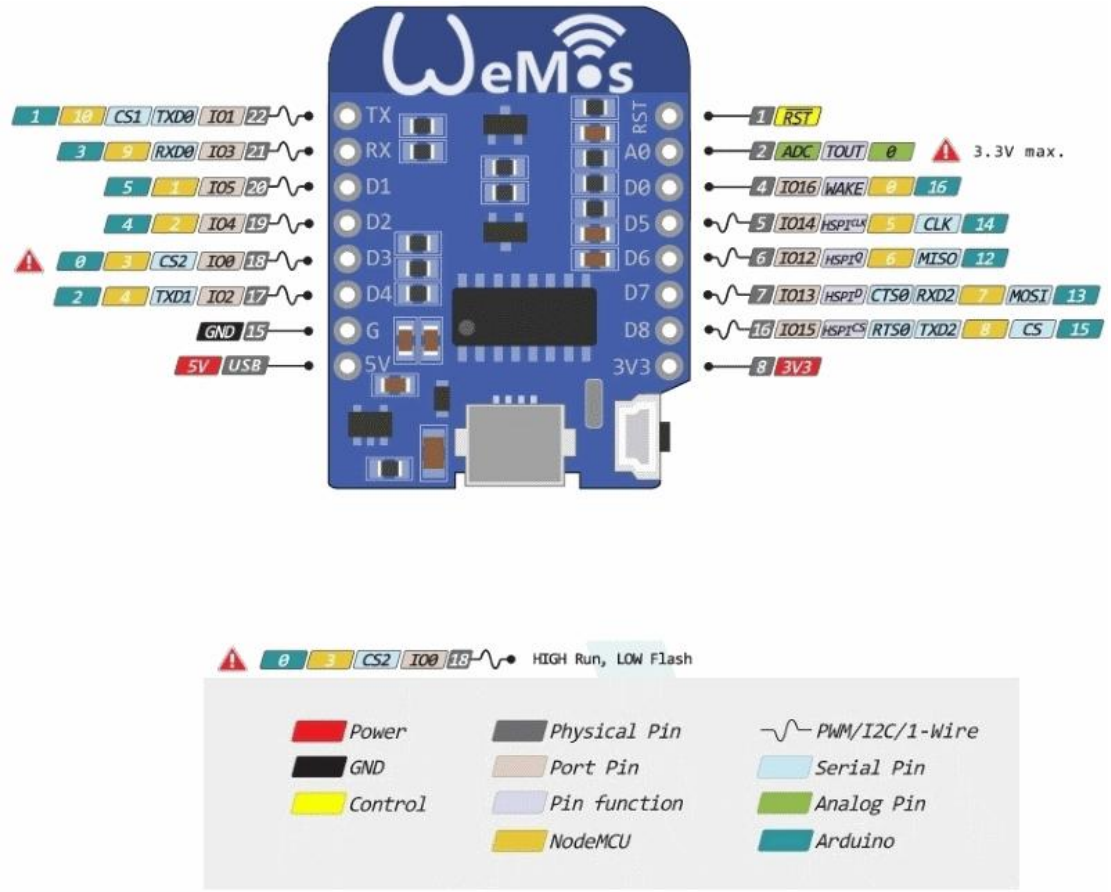

Рисунок 34 – Схема и расположение контактов платы Wemos D1 R1 mini

Для модуля ESP – 01S настройка и подключение выглядят немного иначе обладает следующими выходами (рисунок 35):

- GND – земля;

- TXD0, данный контакт служит для подключения светодиода, загорающегося при низком логическом уровне на GPIO1 либо при передаче данных по UART;

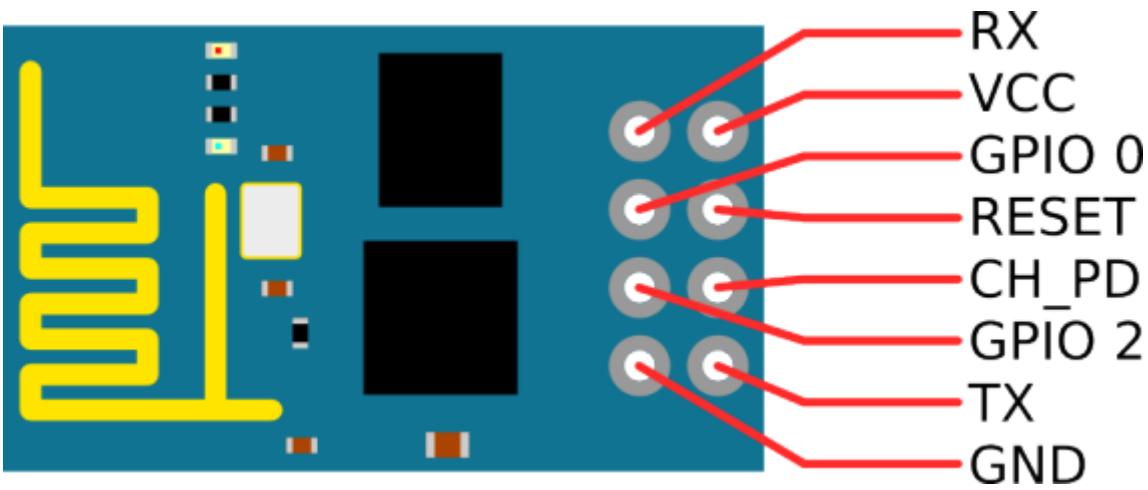

Рисунок 35 – Распиновка портом модуля ESP-01S

- GPIO2, порт ввода и вывода;
- CP_PD, данный контакт применяется для перевода устройства в энергосберегающий режим;
- GPIO0, порт ввода и вывода, позволяющий менять режим программирования устройства устройство в режим программирования, в случае подачи на порт напряжения;
- RST, порт использующийся для перезагрузки микроконтроллера в случае подачи на него низкого логического уровня;
- RXD0 аппаратный порт применяющийся для прошивки и перепрошивки модуля;
- VCC – питание.

Модуль ESP-01S можно подключить к персональному компьютеру двумя способами:

- используя в виде переходника печатную плату Arduino UNO с последовательной загрузкой специализированного скетча и дальнейшей настройкой модуля при помощи AT команд;
- подключить плату через UART переходник.

Рассмотрим оба варианта.

При первом случае подключение приобретет следующий вид (рисунок 36).

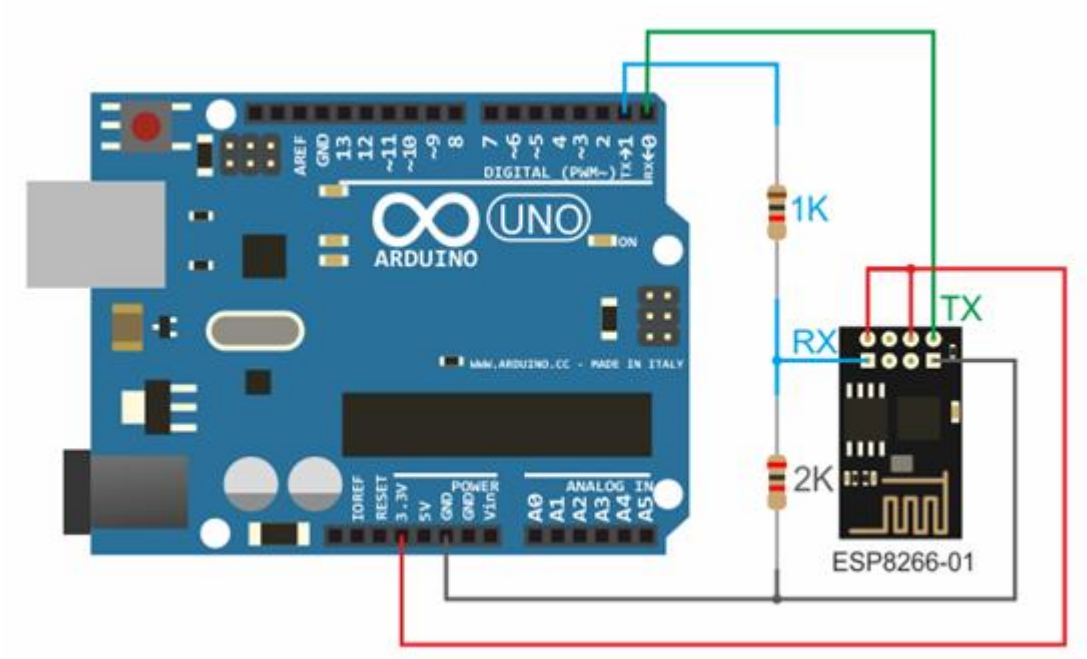

Рисунок 36 – Схема подключение модуля ESP-01S

Питание с Ардуино равное 3.3 Вольта поступает на порты модуля ESP-01S Vcc & CH_PD порт GND соединяется с портом GND, порт TX с Ардуино подключается к порту RX модуля ESP-01S, аналогичным способом подключается порты TX Ардуино и RX с ESP-01S.

Так же стоит помнить что у ESP8266 напряжение питания не может быть выше 3,6В, в то время как на пате Ардуино напряжение равно 5В. Соединять 2 микроконтроллера нужно с помощью резистивных делителей.

При данном подключении программирование модуля происходит через серийный порт платы Arduino UNO при помощи специального скетча загружаемого на плату через программное обеспечение Arduino IDE.

Скетч имеет следующий вид:

// serial-порт к которому подключён Wi-Fi модуль

```
#define WIFI_SERIAL    Serial1
void setup()
{
  Serial.begin(9600);
  while (!Serial) {
  }
  Serial.print("Serial init OK\r\n");
  WIFI_SERIAL.begin(115200);
}
void loop()
{
  if (WIFI_SERIAL.available()) {
    Serial.write(WIFI_SERIAL.read());
  }
  if (Serial.available()) {
    WIFI_SERIAL.write(Serial.read());
  }
```

Сам скетч действует следующим образом:

Выбирается порт к которому будет происходить подключение wi-fi модуля, затем открываем последовательный порт для мониторинга действий в программе и передаём скорость 9600 бод после чего ждём, пока не откроется монитор последовательного порта для того, чтобы отследить все события в программе.

После чего в разделе программы void loop запускается следующий шаг, если приходят данные из Wi-Fi модуля - отправим их в порт компьютера, в случае же если приходят данные из компьютера - отправим их в Wi-Fi модуль.

Настройка модуля происходит через монитор порта с использованием AT команд. Список AT команд представлен в таблице ниже.

Таблица 2 – список AT команд

Команда	Описание	Тип	Выполнение
AT	Проверка модуля	базовая	AT
AT+RST	Перезапуск модуля	базовая	AR+RST
AT+GMR	Отобразить версию прошивки.	базовая	AT+GMR
AT+GSLP	Переход в режим пониженного энергопотребления	базовая	AT+GSLP=<время в мс>
ATE	включить/выключить эхо	базовая	ATE0 ATE1
AT+RESTORE	Сбросить на заводские настройки	базовая	AT+RESTORE
AT+UART_DEF	Команда полностью аналогична AT+UART_CUR	базовая	
AT+CWMODE_CUR	Переключение режима wifi для текущего сеанса (current, т.е. без сохранения во flash память)	wifi	AT+CWMODE_CUR=<режим>
AT+CWMODE_DEF	Команда полностью аналогична AT+CWMODE_CUR	wifi	
AT+CWJAP_CUR	Подключение к AP, точке доступа (current, т.е. без сохранения во flash память)	wifi	AT+CWJAP_CUR=<идентификатор сети>,<пароль>
AT+CWJAP_DEF	Команда полностью аналогична AT+CWJAP_CUR	wifi	
AT+CWLAP	Отобразить список доступных точек доступа	wifi	AT+CWLAP показывает все доступные точки доступа
AT+CWQAP	Отключение от точки доступа	wifi	AT+CWQAP
AT+CWSAP_CUR	Создать SoftAP (точку доступа) для текущего сеанса	wifi	AT+CWSAP_CUR= <идентификатор сети>,<пароль>,<канал>, <тип шифрования>

Продолжение таблицы 2

Команда	Описание	Тип	Выполнение
AT+CWSAP_DEF	Команда полностью аналогична AT+CWSAP_CUR	wifi	
AT+CWLIF	Отобразить IP адреса станций, подключенных к ESP8266 SoftAP точке доступа	wifi	AT+CWLIF
AT+CWDHCP_CUR	Включить или выключить DHCP сервер для текущего сеанса	wifi	AT+CWDHCP_CUR=<режим>,<вкл>
AT+CWDHCP_DEF	Команда полностью аналогична AT+CWDHCP_CUR	wifi	
AT+CWAUTOCONN	Автоматическое подключение к точке доступа (сохраняется во флеш память)	wifi	AT+CWAUTOCONN=<вкл>
AT+CIPSTAMAC_CUR	посмотреть/установить MAC адрес в режиме station для текущего сеанса	wifi	AT+CIPSTAMAC=
AT+CIPSTAMAC_DEF	Команда полностью аналогична AT+CIPSTAMAC_CUR	wifi	
AT+CIPAPMAC_CUR	посмотреть/установить MAC адрес в режиме SoftAP (точка доступа) для текущего сеанса	wifi	AT+CIPAPMAC_CUR=
AT+CIPAPMAC_DEF	Команда полностью аналогична AT+CIPAPMAC_CUR	wifi	
AT+CIPSTA_CUR	посмотреть/установить IP адрес в режиме station для текущего сеанса	wifi	AT+CIPSTA_CUR=[,<шлюз>,<маска>]
AT+CIPAP_CUR	Посмотреть/установить IP адрес в режиме SoftAP (точка доступа) для текущего сеанса	wifi	AT+CIPAP_CUR=

Продолжение таблицы 2

Команда	Описание	Тип	Выполнение
AT+CIPAP_DEF	Команда полностью аналогична AT+CIPAP_CUR	wifi	
AT+CWSTOPSMART	Команда останавливает процесс SmartConfig	wifi	AT+CWSTOPSMART
AT+CIPCLOSE	Закрыть соединение TCP или UDP	TCP/IP	1. Множественное подключение: (+CIPMUX=1) AT+CIPCLOSE=2 . Одиночное подключение (+CIPMUX=0) AT+CIPCLOSE
AT+CIFSR	Отобразить локальные IP адреса, адрес, который получили от точки доступа, к которой подключены и IP адрес ESP8266 SoftAP (локальной точки доступа)	TCP/IP	AT+CIFSR
AT+CIPMUX	Выбрать режим одиночного или множественных подключений	TCP/IP	AT+CIPMUX=<режим>
AT+CIPSERVER	Запустить (перезапустить) TCP сервер	TCP/IP	AT+CIPSERVER= <режим>[,<порт>]
AT+CIPSTO	Установить/посмотреть таймаут сервера	TCP/IP	AT+CIPSTO=<таймаут>
AT+CIUPDATE	Обновление прошивки через облако. Модуль должен быть в режиме 1 или 3 и быть подключен к точке доступа с выходом в интернет.	TCP/IP	AT+CIUPDATE

Продолжение таблицы 2

Команда	Описание	Тип	Выполнение
AT+CIPMODE	Установить сквозной режим "unvarnished transmission mode"	TCP/IP	AT+CIPMODE=<режим>
AT+SAVETRANSLINK	Save transparent transmission link to Flash		AT+SAVETRANSLINK=<режим>,,<порт>
AT+PING	Пинг по имени хоста или IP адресу	TCP/IP	AT+PING=ip

Подключение платы через UART переходник имеет следующий вид (рисунок 37).

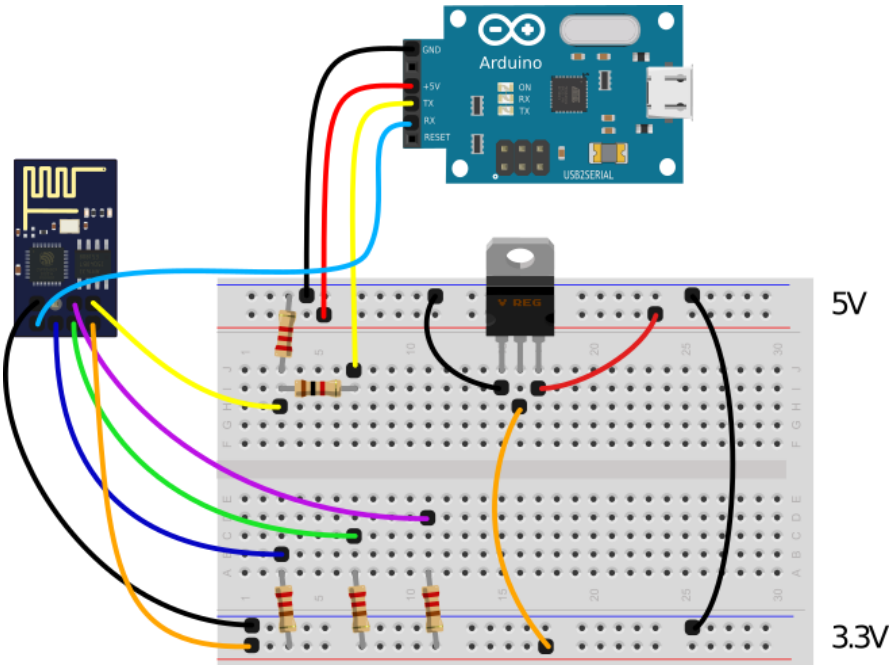

Рисунок 37 – Подключение ESP-01S модуля к UART для последующего программирования

При данном подключении модуль работает в режиме работы по умолчанию в зависимости от прошивки. Для перевода в режим загрузки скетча + требуется проделать комбинацию следующих действий:

- отключите питание от модуля;
- подключите пин 0 к GND — фиолетовый провод к земле;
- подключите модуль к питанию;

- притяните пин 0 к 3.3V — фиолетовый провод через резистор к питанию.

После чего модуль переходит в режим загрузки и на него можно загрузить любой скетч. Для загрузки скетча в плату Wemos D1 R1 mini и модуля ESP-01S требуется подготовить программное обеспечение, иными словами подключить специализированные библиотеки для работы с микроконтроллером ESP8266.

Первым делом для того чтобы программа Arduino IDE смогла найти требуемые библиотеки и подключить их, нам надо разместить ссылку на существующие библиотеки, данный шаг осуществляется в разделе «Настройки» в поле «Дополнительные ссылки для Менеджера плат:». Ссылка для подключения библиотек имеет следующий вид: «http://arduino.esp8266.com/stable/ package_esp8266com_index.json».

Чтобы все получилось правильно раздел «настройки» должен выглядеть как это показано на рисунке 38.

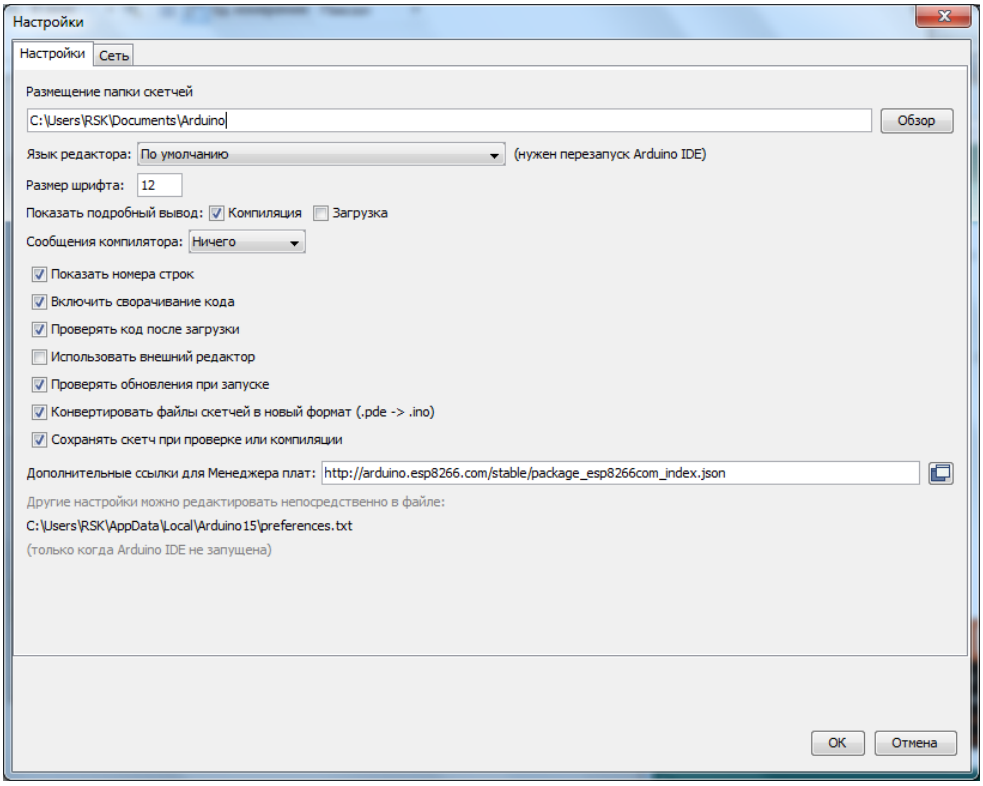

Рисунок 38 – Настройки программы Arduino IDE для подключения библиотеки ESP8266

После чего следует перейти в раздел «Менеджер плат», путь к данному разделу будет следующим «Инструменты/Плата:/Менеджер плат» и имеет вид как показано на рисунке 39.

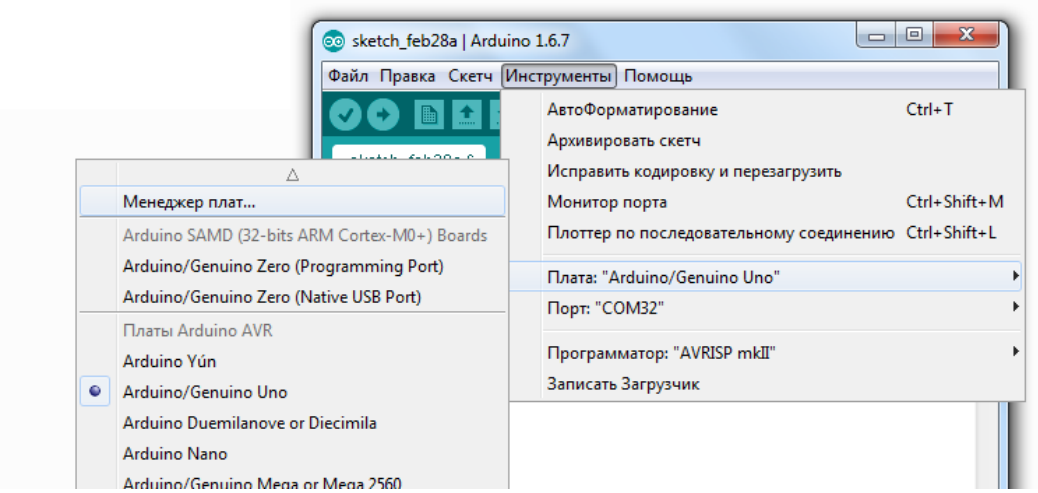

Рисунок 39 – Путь к «Менеджеру плат» в программе Arduino IDE

После чего откроется окно «Менеджер плат» отфильтровав результаты поиска по слову ESP8266 мы увидим следующее (рисунок 40).

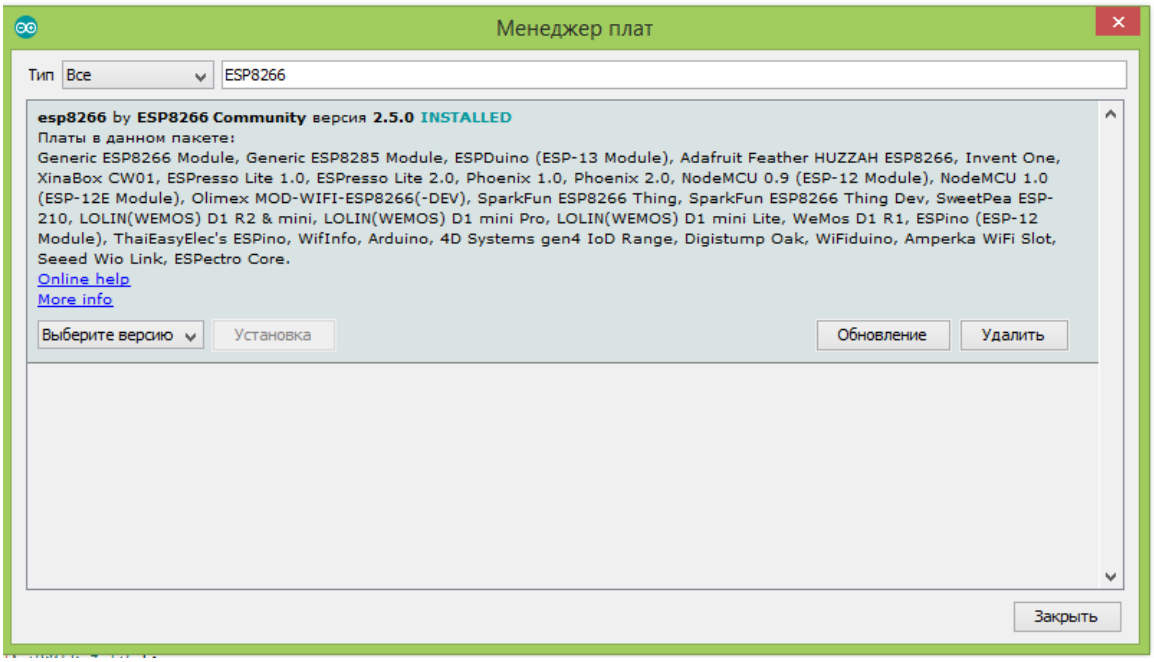

Рисунок 40 – Установленная библиотека ESP8266

После чего в списке плат появляеются новые доступные модули для работы что можно увидеть на рисунке 41.

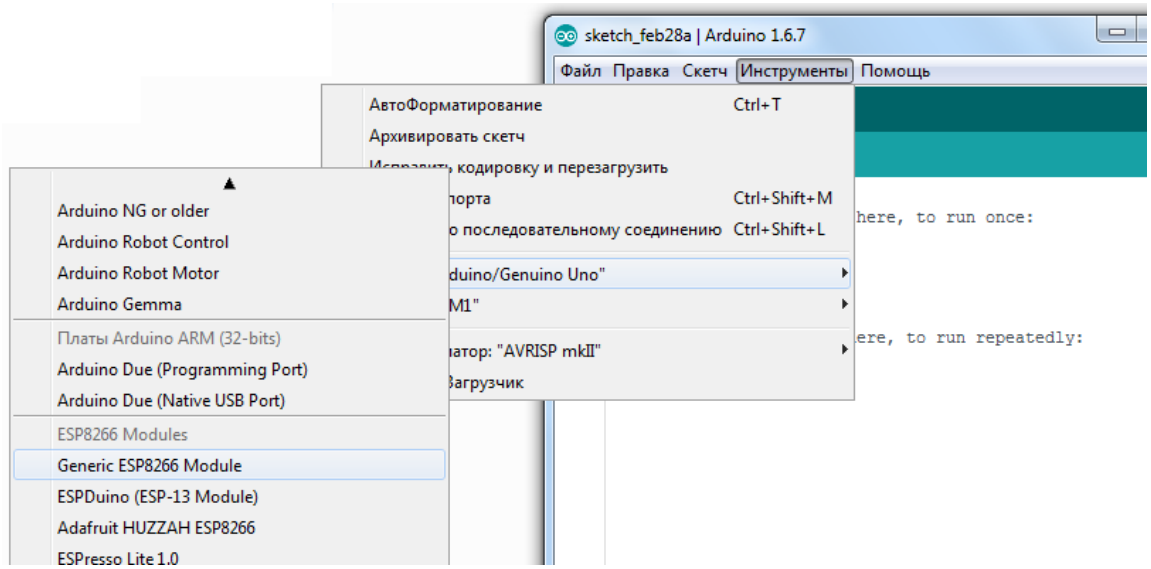

Рисунок 41 – Новые модули на базе микропроцессора ESP8266

Выбрав нужную плату мы можем загрузить написанный скетч, в данный момент это будет скетч по созданию точки доступа к сети.

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#ifndef APSSID
#define APSSID "ESPap"
#define APPSK  "thereisnospoon"
#endif
const char *ssid = APSSID;
const char *password = APPSK;
ESP8266WebServer server(80);
void handleRoot() {
  server.send(200, "text/html", "<h1>You are connected</h1>");
}
void setup() {
  delay(1000);
  Serial.begin(115200);
  Serial.println();
```

```
  Serial.print("Configuring access point...");
  WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.on("/", handleRoot);
  server.begin();
  Serial.println("HTTP server started");
}
void loop() {
  server.handleClient();
}
```

После чего прошиваем плату ESP-01S для работы как тока доступа, из-за того что в пунктах const char *ssid = APSSID и const char *password = APPSK мы прописали данные значения у нас появится сеть с именем APSSID и паролем APPSK для входа.

3 Построение стенда, разработка программного обеспечения, создание пульта управления и построение беспроводной сети

Для разработки программного обеспечения для контроля стенда системы умный дом будет применяться программный комплекс программ следующих типов:

FLProg - программное обеспечение, позволяющее писать программный код на графических языках Ladder и LBP, применяющихся в программировании промышленных контроллеров, а так же может компилировать данные языки в C#.

Arduino IDE - программное обеспечение, использующееся для работы с модулями компании Arduino, компилирующее язык C# в машинный код для платы и её микрочипа.

Чтобы обеспечить совместимость программного обеспечения Arduino IDE были проделаны манипуляции описанные в разделе «Подключение настройка и первый запуск модулей Wi-fi связи ESP-01S и Wemos D1 R1 mini» что позволило использовать данное программное обеспечение не только с платами Ардуино, но и с модулями на базе микроконтроллера ESP8266.

Прежде чем приступать к программированию следует определиться с блок схемой программного обеспечения каждого отдельного модуля.

Модуль A1 на базе Wemos D1 R1 mini выступает в качестве сервера, на котором будет строиться вся сеть. Данный модуль будет создавать точку доступа, обеспечивать безопасность сети и панели управления и позволять пользователю использовать ручное управление стендом. Чтобы избежать холостой работы стенда так же требуется настроить временные периоды работы системы. Система функционирует автоматически в период с 7 утра до 10 часов вечера, остальной временной период находится в состоянии покоя, если пользователю потребуется продлить данный период он может сделать это как вручную, так и перенастроив саму систему.

Так же следует определиться с доступом к функционалу всего стенда, для этого создадим несколько пользователей, и дадим каждому отдельному пользователю разный уровень доступа.

- администратор - имеет доступ ко всем функциям, может перенастроить доступ, сменить ip адрес каждого отдельного модуля имеет доступ к прошивке модуля по сети вай-фай и перезагрузке стенда;

- преподаватель, имеет доступ к системе стенда, может настраивать логин и пароль для ниже стоящих пользователей, обладает возможностью перезагружать стенд и выполнять ручное управление;

- студенты - группа пользователей имеющая доступ лишь к управляющей части стенда, для целей безопасности не имеют возможности перепрошивать стенд и перезапускать систему.

Определив функционал данного модуля перейдем к его настройке.

К модулю А1 не будет подключаться никаких вспомогательных устройств и датчиков, чтобы не вызвать перебоев в работе системы. На схеме А1 будет иметь следующий вид.

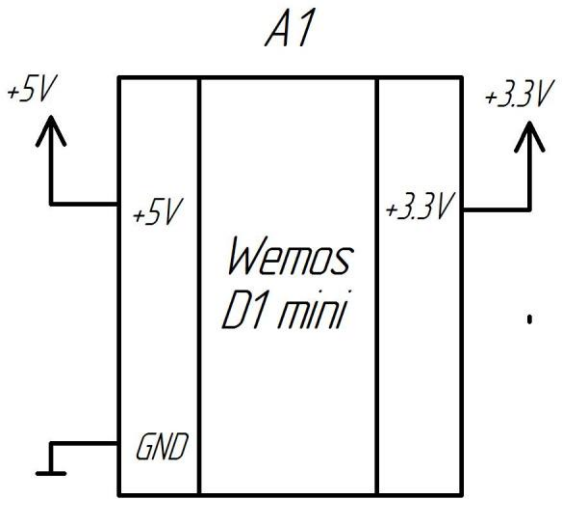

Рисунок 42 - Модуль А1 на базе Wemos D1 mini

Настройка программного обеспечения производится через программу FLProg и выглядит следующим образом.

Первым делом происходит выбор элемента который будет программироваться, в данном случае это WeMos Mini. Экран выбора имеет следующий вид (рисунок 43).

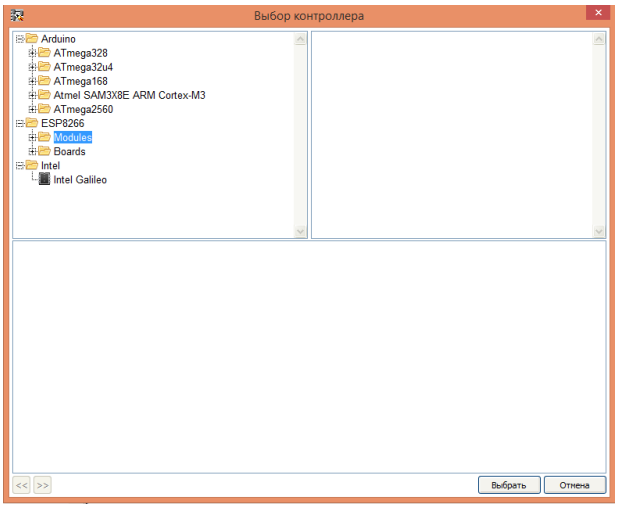

Рисунок 43 - Перечень плат поддерживаемых программным обеспечением

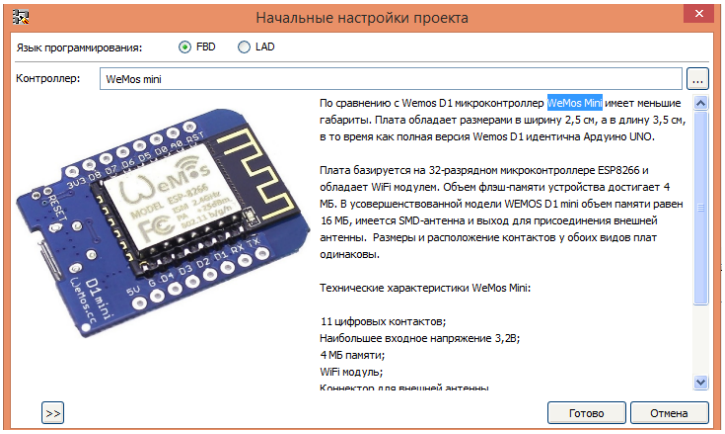

Рисунок 44 - Подробная информация о выбранной плате в программе FLProg

Настройка чипа и контроллера осуществляется вручную, и имеет следующий вид (рисунок 45).

Кроме данных настроек, так же проводится работа с Web-интерфейсом, создание пользователей, настройка доступа, создание страниц и специального функционала позволяющего контролировать иные стенды путем отправки сигналов, и изменения среды.

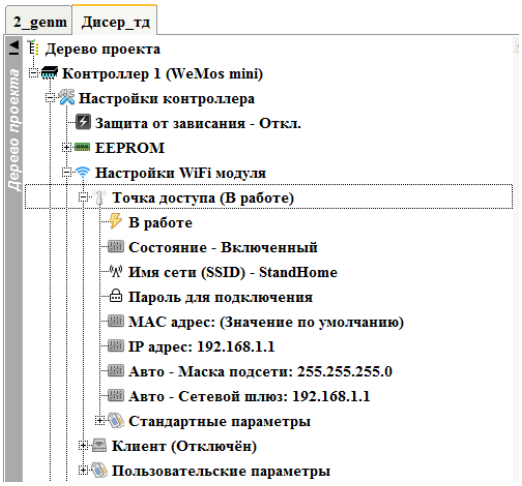

Рисунок 45 - Панель настроек контроллера

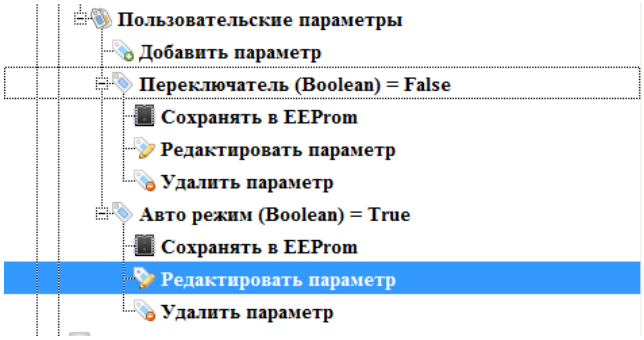

Рисунок 46 - Список пользовательских параметров

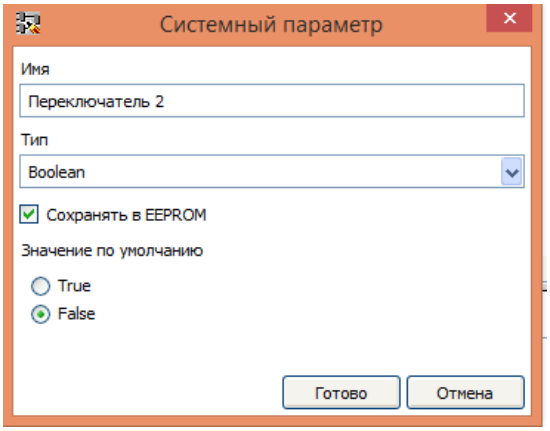

Рисунок 47 - создание пользовательского параметра с сохранением во flash память использующего логический тип данных со значением 0 по умолчанию

Для того чтобы разместить код программы на Web-интерфейсе, следует создать Web-интерфейс, определить пользователей которые получат доступ к системе, а так же определиться со стилем Web-страниц, которые будут загружаться на плату.

Стиль создается через язык CSS, и имеет следующий вид:

```
.menu
    {
    float:left;
    padding: 1%;
     margin: 1%;
    width:16%;
     border: 1px solid black;
    border-radius: 8px;
    }
.menuItem
    {
    font-weight: 600;
    font-family: 'Times New Roman', Times, serif;
    }
 .header
    {
    padding: 10px;
    left: 0px;
    right: 0px;
    top: 0px;
    background: #00FFFF;
    text-align: center;
    font-weight: 600;
    font-family: 'Times New Roman', Times, serif;
    font-size: 100%;
```

```
        }
.content
        {
        float:right;
        width:78%;
        }
.footer
        {
        float:left;
        padding: 1%;
        width:98%;
        background: #00FFFF;
        margin-top: 1%;
        text-align: center;
        font-weight: 600;
        font-family: 'Times New Roman', Times, serif;
        font-size: 150%;
        }
.buttonFlp
        {
        width:150px;
        border-radius:20px;
        background:#459DE5;
        color:#fff;
        font-size:12px;
        cursor:pointer;
        float:left;
        padding: 1%;
         margin: 1%;
        }
```

```
.buttonFlp:hover
    {
    background:#358DE5;    }
.buttonFlp:focus
    {
    outline:none;
    }
```

При использовании данного кода программное обеспечение настраивает страницу в соответствии с представленными выше требованиями.

Рисунок 48 – Первичный web-интерфейс загруженный на d1 mini

Сама программа позволяющая контролировать отдельные элементы сети разрабатывается на графическом языке программирования ladder. Программа будет выглядеть как настраиваемая блок схема каждый элемент которой будет выглядеть как отдельный настраиваемый блок. Для контроля блока А3 через контроллер А1, нужно создать пользовательские системные параметры которые будут отвечать за контроль, а так же настроить и подключить блоки передачи данных. В итоге программа должна приобрести следующий вид, где SysParRead это блок системного параметра вводимы пользователем в ручную для смены режима, а блоки SVFC 1 и 2 отвечают за отправку информации на клиентские модули.

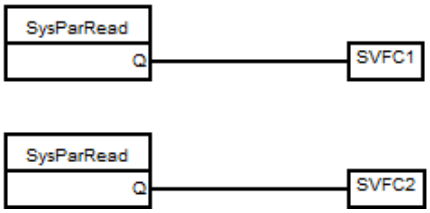

Рисунок 49 - Пульт управления контроллером А2, расположенный на контроллере А1

Далее следует провести настройку блока передачи данных по определенному ip адресу и порту.

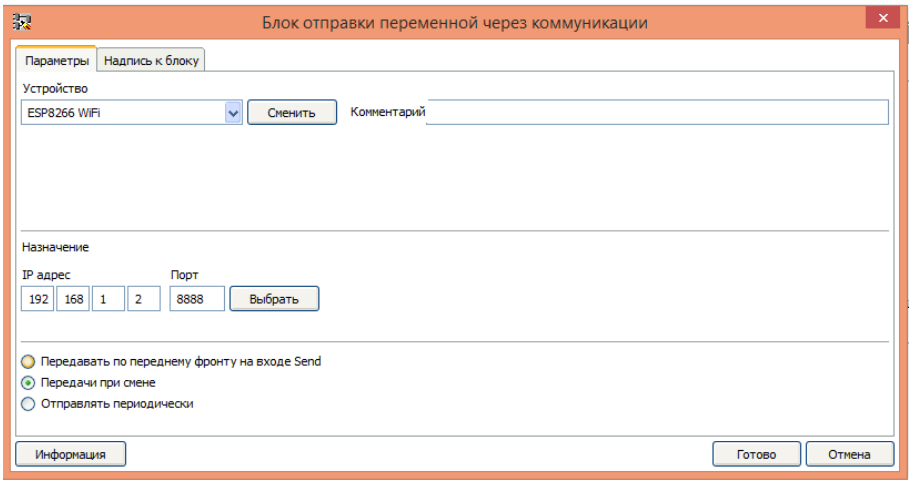

Рисунок 50 – настройка блока SVFC, для передачи данных между блоками по протоколу TCP/IP

Аналогичная настройка и построение блоков проводится для каждого отдельного модуля, но с изменением ip адреса назначения.

После проведения настроек и постарения требуемых блок схем, следует начать процесс компиляции программы. Все настройки и блок-схемы переведутся в формат языка C#, который можно будет загрузить на печатную плату через программное обеспечение Arduino IDE.

Разработанный код после загрузки создаст защищенную паролем точку доступа к локальной сети WI-FI, но без Web-интерфеса. Так как для его загрузки надо добавить к программному обеспечению Arduino IDE возможность загружать Web-элементы на плату.

Для того чтобы программное обеспечение Arduino IDE позволило загружать Web-элементы следует проделать следующие действия:

- зайти в папку tools в системном разделе Arduino IDE;
- загрузить специально разработанное для данных целей расширение ESP8266 using esptool в папку tools;
- перезагрузить программное обеспечение Arduino IDE;
- запустить скетч с элементами Web-интерфеса;

- перед загрузкой скетча убедиться в том что плата и порт выбраны верно;

- запустить ESP8266 Sketch Data Upload, в обновившемся меню устройства.

Код который получится в итоге будет выглядеть следующим образом, показанным в приложении 1.

Программное обеспечение данного модуля создает локальную сеть на DNS сервере с IP 192.168.1.1. Данная сеть получает наименование StendHome с защитой паролем 123456789. Для контроля стенда используется Web-интерфейс, доступ к которому имеют лишь два пользователя системы, Студент и Преподаватель. Контроль стендом осуществляется как в ручную, так и при помощи автоматических систем.

Остальные модули стенда, а именно A2, A3, A4 и A5 будут подключаться к стенду в роли клиентов сети при помощи заданных параметров для входа в сеть. Каждый модуль получит свой персональный ip адрес к которому сможет обращаться пользователь.

К микроконтроллеру A2 будут подключены следующие элементы: магнитное реле управляющее освещением, дальномер для контроля за периметром коридора, а так же серво привод для управления жалюзи и шторами. На схеме микроконтроллер A2 со всеми подключенными фрагментами будет иметь следующий вид (рисунок 51).

Программное обеспечение данного модуля позволяет подключаться к локальной сети по средством WI-FI. Как автоматически так и вручную управлять системой освещения коридорного помещения, а так же регулировать уровень штор и жалюзи.

Так как все модули кроме A1 являются клиентами сети, их настройка проводится в соответствии со следующими требованиями.

Включение в режиме клиента, подключение к сети при помощи имени и пароля, получение прописанного в ручную ip адреса для использования передачи данных через TCP/IP.

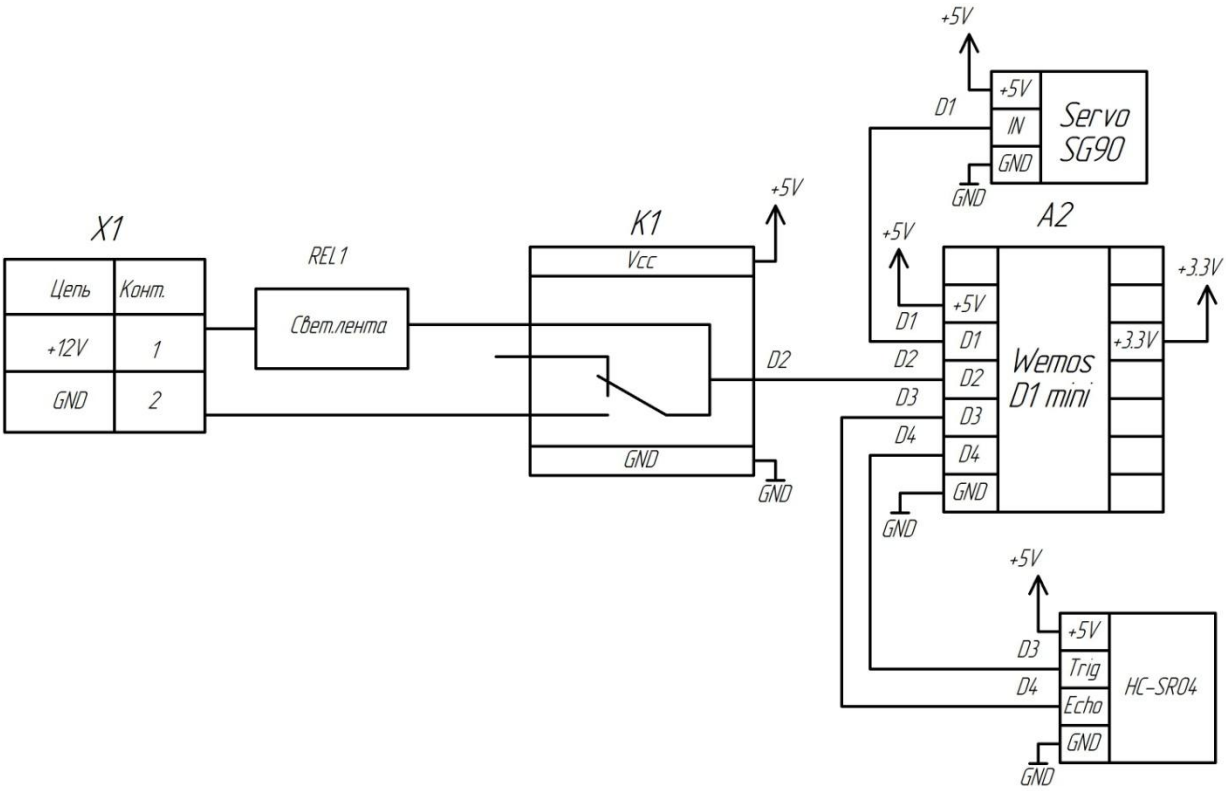

Рисунок 51 - Модуль A2 на базе Wemos D1 mini

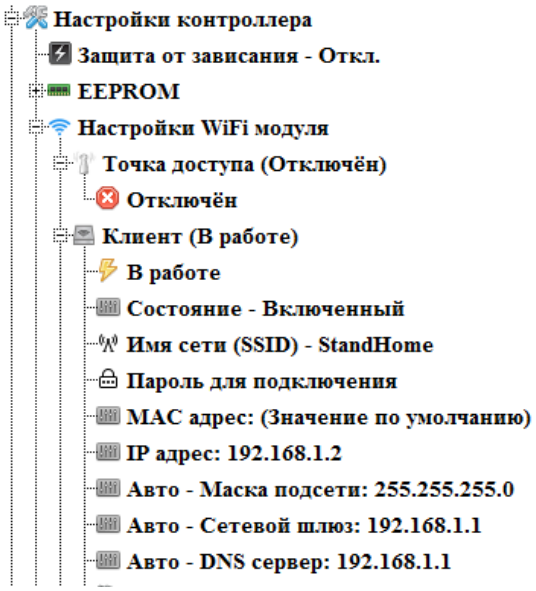

Рисунок 52 - Настройки применяющиеся к микроконтроллеру A2

После чего на модуль следует загрузить программу которая сможет контролировать систему освещения и управления жалюзи соответствующую блок-схемам.

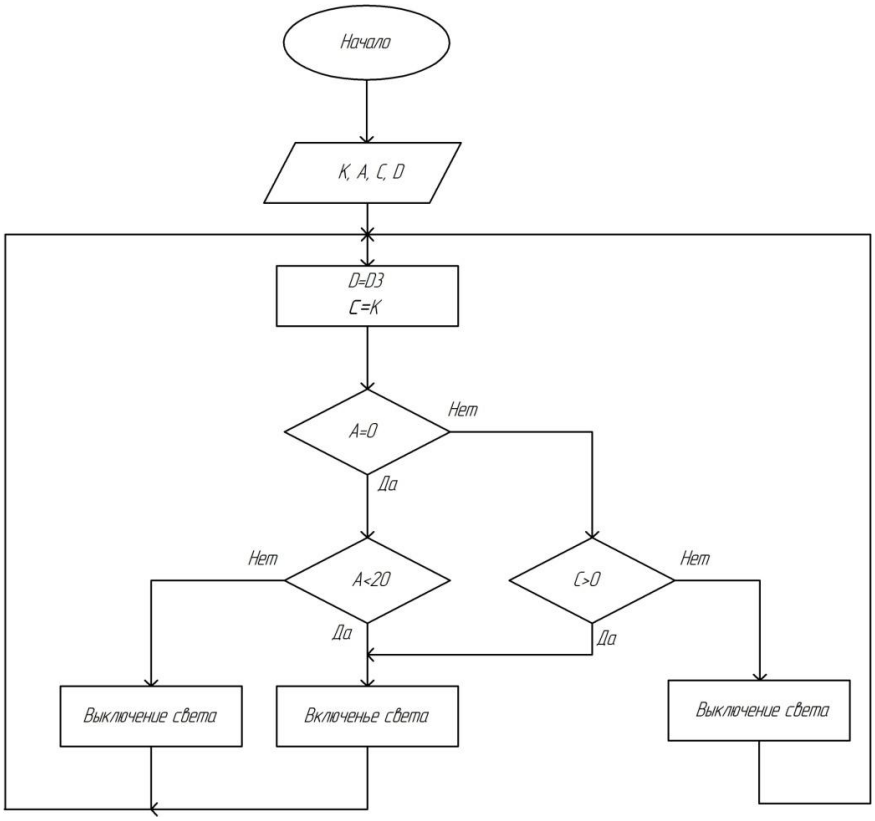

Рисунок 53 - Блок схема программы автоматического и ручного управления освещением в коридоре, К-это ручной контроль, А-это автомат, D-это дистанция

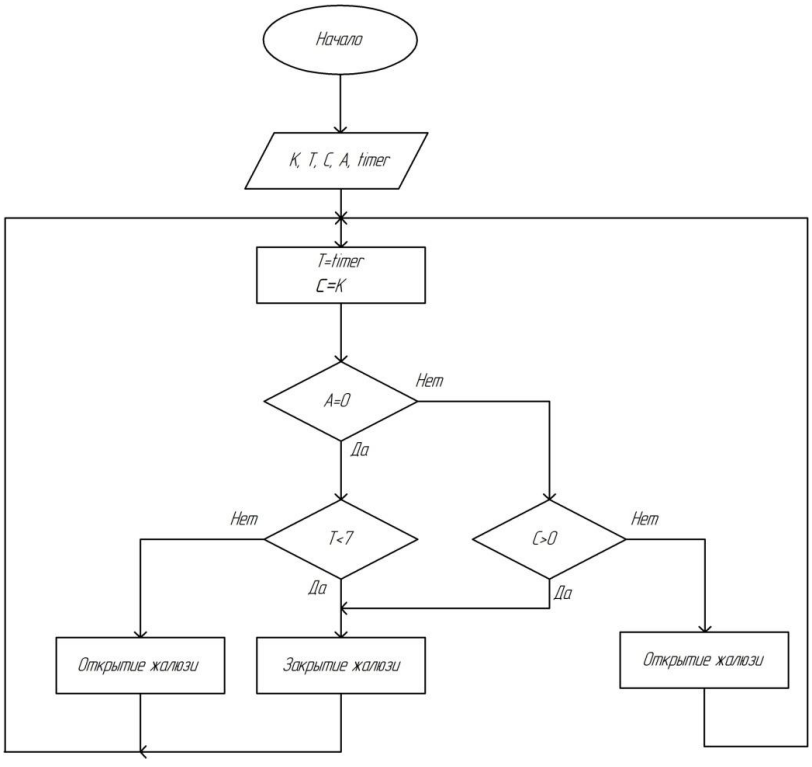

Рисунок 54 - Блок схема управления жалюзи, К-это ручной контроль, А-автомат, Т-время

На графическом языке ladder данные блок схемы примут следующий вид:

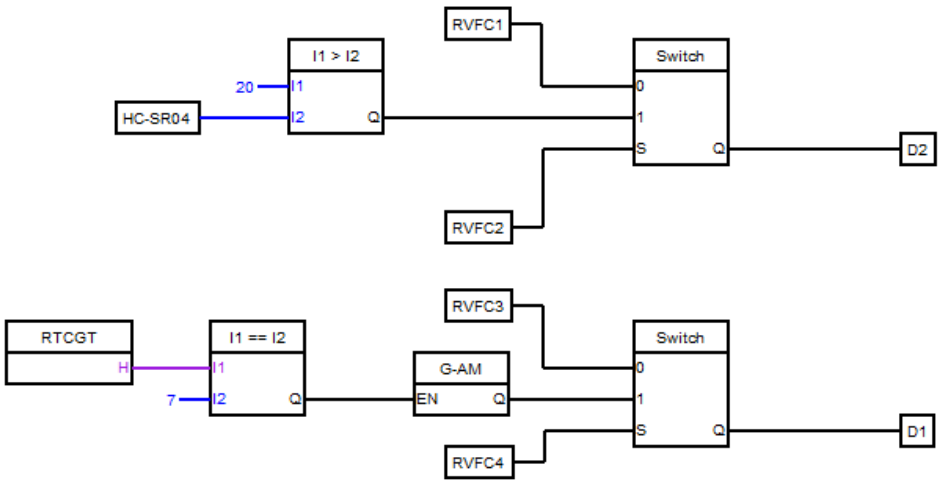

Рисунок 55 - Программы управления освещением сверху и жалюзи снизу на языке программирования Ladder

Данные программы выражены логическими блок схемами, каждая из которых отвечает за свои параметры.

D1 - выход D1, D2 - выход D2 на плате Wemos D1 mini, Switch переключатель сигнала, RVFC 1-4 сигналы управления приходящие с модуля A1, G-AM асинхронный мультивибратор, I1>I2 и I1==I2 логические операторы сравнения, HC-SR04 блок отвечающий за дальномер модуль HC-SR04, а блок RTCGT отвечает за реальное время.

При переводе на язык C# программа будет иметь следующий вид.

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <TimeLib.h>
extern "C" {
#include "user_interface.h";
}
int _resiveModeCommunication = 0;
String _resiveVariableStringCommunication;
int _resiveVariableIndexCommunication = 0;
String _resiveVariableValueCommunication;
bool ESP8266ControllerWifiClient_HRD = 0;
bool ESP8266ControllerWifiClient_status = 1;
bool ESP8266ControllerWifiClient_isDHCP = 0;
bool ESP8266ControllerWifiClient_IsNeedReconect = 0;
bool ESP8266ControllerWifiClient_workStatus = 1;
```

```
char ESP8266ControllerWifiClient_SSID[40] = "StandHome";
char ESP8266ControllerWifiClient_password[40] = "123456789";
IPAddress ESP8266ControllerWifiClient_ip(192, 168, 1, 2);
 IPAddress  ESP8266ControllerWifiClient_dns (192, 168, 1, 1);
 IPAddress  ESP8266ControllerWifiClient_gateway (192, 168, 1, 1);
IPAddress ESP8266ControllerWifiClient_subnet (255, 255, 255, 0);
 uint8_t ESP8266ControllerWifiClient_mac[6] = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
WiFiUDP Udp;
char _udpPacketBuffer [UDP_TX_PACKET_MAX_SIZE];
bool _gen1I = 0;
bool _gen1O = 0;
unsigned long _gen1P = 0UL;
bool _RVFC2 = 0;
int _ultrasonic1O = 0;
unsigned long _ultrasonic1P = 0UL;
bool _swi1;
bool _RVFC3 = 0;
bool _swi2;
bool _RVFC1 = 0;
bool _RVFC4 = 0;
void setup()
{
WiFi.mode(WIFI_STA);
_esp8266WifiModuleClientReconnect();
Udp.begin(8888);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);

pinMode(2, OUTPUT);
pinMode(0, INPUT);
}
void loop()
{int _udpPacketSize = Udp.parsePacket();
 if (_udpPacketSize)  {
Udp.read(_udpPacketBuffer, UDP_TX_PACKET_MAX_SIZE);
 for(int i=0; i <= _udpPacketSize; i++) {
_ressiveByteFromCommunication(_udpPacketBuffer[i]);}
}
if(ESP8266ControllerWifiClient_IsNeedReconect)
{_esp8266WifiModuleClientReconnect(); ESP8266ControllerWifiClient_IsNeedReconect = 0; }
ESP8266ControllerWifiClient_status = WiFi.status() == WL_CONNECTED;
if (ESP8266ControllerWifiClient_status) { if ( ! ESP8266ControllerWifiClient_HRD) {
ESP8266ControllerWifiClient_ip =  WiFi.localIP();
ESP8266ControllerWifiClient_subnet =  WiFi.subnetMask();
ESP8266ControllerWifiClient_gateway =  WiFi.gatewayIP();
ESP8266ControllerWifiClient_dns =  WiFi.dnsIP();
WiFi.macAddress(ESP8266ControllerWifiClient_mac);
ESP8266ControllerWifiClient_HRD = 1;
}} else {ESP8266ControllerWifiClient_HRD = 0;}
if (abs((millis() - _ultrasonic1P))>100) {
digitalWrite(2, HIGH);
delayMicroseconds(10);
```

```
digitalWrite(2, LOW);
int _ultrasonicTemp=(pulseIn(0, HIGH))/58;
if( _ultrasonicTemp<200) {_ultrasonic1O=_ultrasonicTemp;}
_ultrasonic1P=millis();}
if(_RVFC2)
{_swi1=(20) > ((_ultrasonic1O));}
else
{_swi1=_RVFC1;}
digitalWrite(4, _swi1);
if (((hour())) == (7)) {if (! _gen1I) { _gen1I = 1; _gen1O = 1; _gen1P =  millis(); } } else {
_gen1I = 0 ;  _gen1O= 0;  } if (_gen1I ) { if (_gen1O) { if ( _isTimer( _gen1P , 43200000 )) {
_gen1P = millis(); _gen1O = 0; } } else  { if ( _isTimer( _gen1P , 0 )) {  _gen1P = millis(); _gen1O
= 1;  } } }
if(_RVFC4)
{_swi2=_gen1O;}
else
{_swi2=_RVFC3;}
digitalWrite(5, _swi2);
}
bool _isTimer(unsigned long startTime, unsigned long period )
  {
  unsigned long currentTime;
currentTime = millis();
if (currentTime>= startTime) {return (currentTime>=(startTime + period));} else {return
(currentTime >=(4294967295-startTime+period));}
  }
void _ressiveByteFromCommunication(byte resiveByte)
{
  if (_resiveModeCommunication==0) {
    if (resiveByte == 1) {
      _resiveModeCommunication=1;
     _resiveVariableStringCommunication="";
      return;
    }
  }
  if (_resiveModeCommunication==1) {
    if((resiveByte == 1 )||(resiveByte == 3)|| (resiveByte == 4)) {
     _resiveModeCommunication=0;
    return;
    }
    else
    {
      if(resiveByte==2) {
        _resiveModeCommunication=2;
       _resiveVariableIndexCommunication=_resiveVariableStringCommunication.toInt();
        _resiveVariableStringCommunication="";
        return;
      }
      else {
        _resiveVariableStringCommunication+= char(resiveByte);
      return;
      }
```

```
        }
       }
       if (_resiveModeCommunication==2) {
        if((resiveByte == 1 )||(resiveByte == 2)|| (resiveByte == 4)) {
          _resiveModeCommunication=0;
        return;
        }
        else
        {
          if (resiveByte==3) {
            _resiveModeCommunication=3;
           _resiveVariableValueCommunication =_resiveVariableStringCommunication;
            _resiveVariableStringCommunication="";
          return;
          }
          else {
            _resiveVariableStringCommunication+= char(resiveByte);
           return;
          }
        }
       }
       if (_resiveModeCommunication==3) {
        if((resiveByte == 1 )||(resiveByte == 2)|| (resiveByte == 3)) {
          _resiveModeCommunication=0;
        return;
        }
        else
        {
          if (resiveByte==4) {
            _resiveModeCommunication=0;
          _fillRessiveVariableasFromCommunication();
            _resiveVariableStringCommunication="";
          return;
          }
          else {
            _resiveVariableStringCommunication+= char(resiveByte);
           return;     }    }
        }
      }
      void _fillRessiveVariableasFromCommunication()
      {
      long  _tempId= _resiveVariableStringCommunication.toInt();
      if ((_resiveVariableIndexCommunication == 1) && (_tempId == 122532102)) {_RVFC1 =
((_resiveVariableValueCommunication).toInt());}
      if ((_resiveVariableIndexCommunication == 2) && (_tempId == 122532102)) {_RVFC2 =
((_resiveVariableValueCommunication).toInt());}
      if ((_resiveVariableIndexCommunication == 3) && (_tempId == 122532102)) {_RVFC3 =
((_resiveVariableValueCommunication).toInt());}
      if ((_resiveVariableIndexCommunication == 4) && (_tempId == 122532102)) {_RVFC4 =
((_resiveVariableValueCommunication).toInt());}
      }
      int hexStrToInt(String instring)
```

```
{
 byte len = instring.length();
 if  (len == 0) return 0;
 int result = 0;
 for (byte i = 0; i < 8; i++)   // только первые 8 цыфар влезуть в uint32
 {
  char ch = instring[i];
  if (ch == 0) break;
  result <<= 4;
  if (isdigit(ch))
   result = result | (ch - '0');
  else result = result | (ch - 'A' + 10);
 }
 return result;
}
 void _esp8266WifiModuleClientReconnect()
{
 if( _checkMacAddres(ESP8266ControllerWifiClient_mac)) {  wifi_set_macaddr(0,
const_cast<uint8*>(ESP8266ControllerWifiClient_mac));}
if( ESP8266ControllerWifiClient_isDHCP) { WiFi.config(0U, 0U, 0U, 0U, 0U); } else {
WiFi.config(ESP8266ControllerWifiClient_ip, ESP8266ControllerWifiClient_gateway,
ESP8266ControllerWifiClient_subnet, ESP8266ControllerWifiClient_dns ,
ESP8266ControllerWifiClient_dns );
}
 WiFi.begin(ESP8266ControllerWifiClient_SSID,
ESP8266ControllerWifiClient_password);
}
bool _checkMacAddres(byte array[])
{
 bool result = 0;
 for (byte i = 0; i < 6; i++)
 {
  if (array[i] == 255) {
   return 0;
  }
  if (array[i] > 0 ) {
   result = 1;
  }
 }
 return result;
}
```

Данное программное обеспечение позволяет как автоматически подстраиваться модулю под различные изменения, так и реагировать на команды пользователя.

Модули А3, А4 и А5 на базе ESP-01 имеют схожую структуру программы так как обладают одинаковой структурой, модули подключены к микроконтроллерам через одинаковые порты, а так же блок схемы программ весьма схожи.

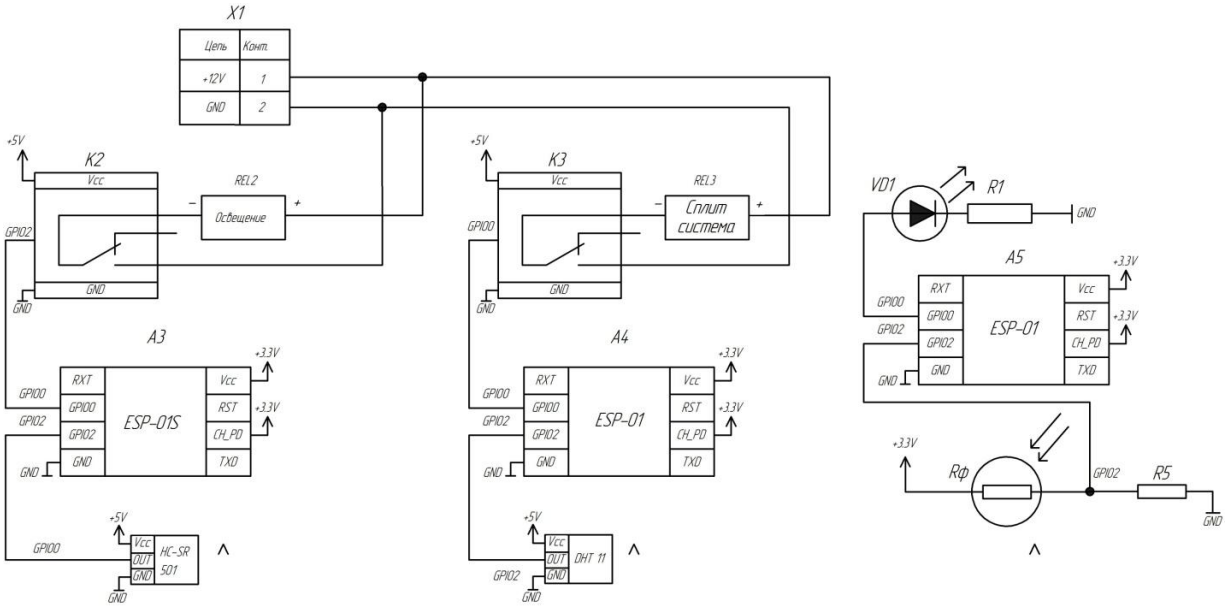

Рисунок 56 - Модули А3, А4, А5 выполненны на базе микроконтроллеров ESP8266

Блок-схема программного обеспечения для модуля А3 имеет следующий вид:

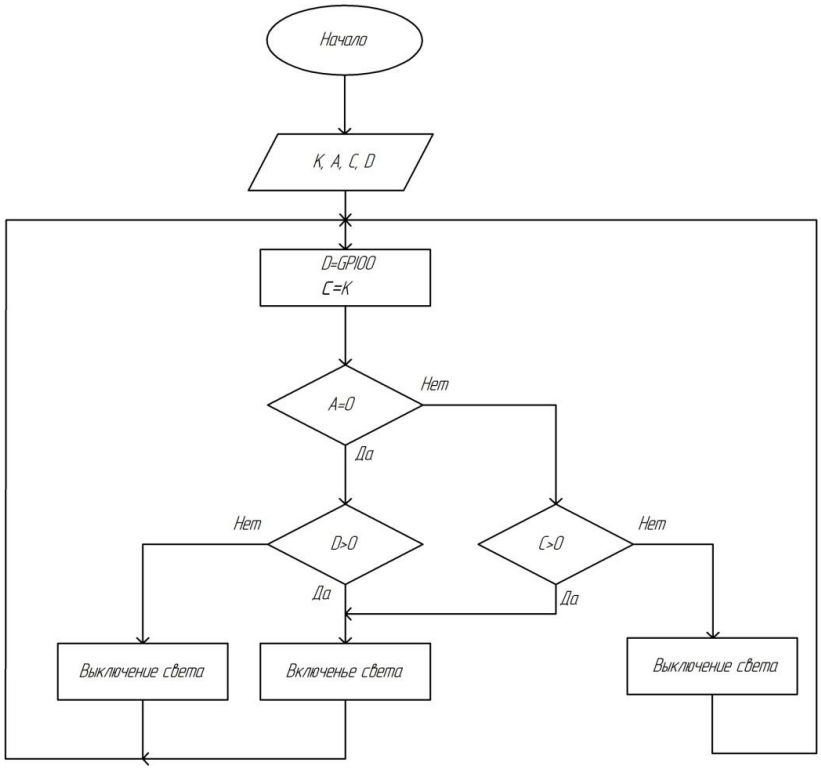

Рисунок 57 - Блок схема модуля А3, К-это ручной контроль, А- автомат, D-это движение

Программный код на языке Ladder для блока А3 примет следующий вид

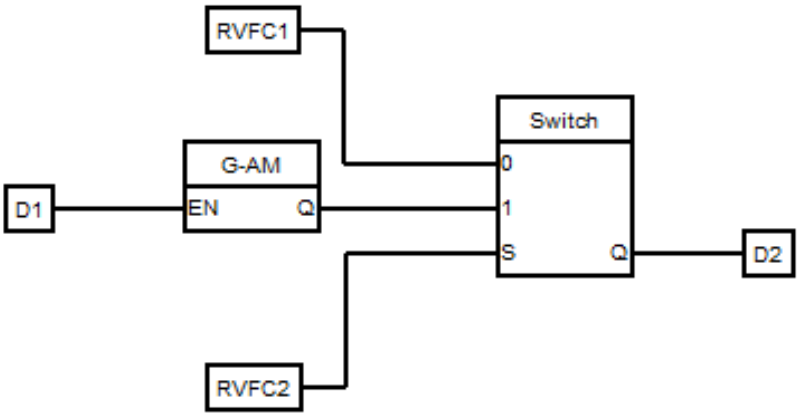

Рисунок 58 - программный код модуля A3 на языке ladder

Данный блок состоит из D1 - вход GPIO2, D2 - выход GPIO0, G-AM асинхронный мультивибратор, RVFC 1 - приходящий сигнал ручного управления, а RVFC 2 - приходящий сигнал автоматического управления.

Данный код при компиляции C# будет иметь следующий вид:

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
extern "C" {
#include "user_interface.h";
}
int _resiveModeCommunication = 0;
String _resiveVariableStringCommunication;
int _resiveVariableIndexCommunication = 0;
String _resiveVariableValueCommunication;
bool ESP8266ControllerWifiClient_HRD = 0;
bool ESP8266ControllerWifiClient_status = 1;
bool ESP8266ControllerWifiClient_isDHCP = 0;
bool ESP8266ControllerWifiClient_IsNeedReconect = 0;
bool ESP8266ControllerWifiClient_workStatus = 1;
char ESP8266ControllerWifiClient_SSID[40] = "StandHome";
char ESP8266ControllerWifiClient_password[40] = "123456789";
IPAddress ESP8266ControllerWifiClient_ip(192, 168, 1, 2);
 IPAddress  ESP8266ControllerWifiClient_dns (192, 168, 1, 1);
 IPAddress  ESP8266ControllerWifiClient_gateway (192, 168, 1, 1);
IPAddress ESP8266ControllerWifiClient_subnet (255, 255, 255, 0);
 uint8_t ESP8266ControllerWifiClient_mac[6] = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
WiFiUDP Udp;
char _udpPacketBuffer [UDP_TX_PACKET_MAX_SIZE];
bool _swi1;
bool _RVFC1 = 0;
bool _RVFC2 = 0;
bool _gen1I = 0;
bool _gen1O = 0;
unsigned long _gen1P = 0UL;
```

```
void setup()
{
WiFi.mode(WIFI_STA);
_esp8266WifiModuleClientReconnect();
Udp.begin(8888);
pinMode(5, INPUT);
pinMode(4, OUTPUT);
}
void loop()
{int _udpPacketSize = Udp.parsePacket();
 if (_udpPacketSize)  {
Udp.read(_udpPacketBuffer, UDP_TX_PACKET_MAX_SIZE);
 for(int i=0; i <= _udpPacketSize; i++) {
_ressiveByteFromCommunication(_udpPacketBuffer[i]);}
}
if(ESP8266ControllerWifiClient_IsNeedReconect)
{_esp8266WifiModuleClientReconnect(); ESP8266ControllerWifiClient_IsNeedReconect = 0; }
ESP8266ControllerWifiClient_status = WiFi.status() == WL_CONNECTED;
if (ESP8266ControllerWifiClient_status) { if ( ! ESP8266ControllerWifiClient_HRD) {
ESP8266ControllerWifiClient_ip =  WiFi.localIP();
ESP8266ControllerWifiClient_subnet =  WiFi.subnetMask();
ESP8266ControllerWifiClient_gateway =  WiFi.gatewayIP();
ESP8266ControllerWifiClient_dns =  WiFi.dnsIP();
WiFi.macAddress(ESP8266ControllerWifiClient_mac);
ESP8266ControllerWifiClient_HRD = 1;
}} else {ESP8266ControllerWifiClient_HRD = 0;}
if ( (digitalRead (5))) {if (! _gen1I) { _gen1I = 1; _gen1O = 1; _gen1P =  millis(); } } else
{ _gen1I = 0 ;  _gen1O= 0;  } if (_gen1I ) { if (_gen1O) { if ( _isTimer( _gen1P , 5000 )) { _gen1P
= millis(); _gen1O = 0; } } else  { if ( _isTimer( _gen1P , 0 )) {  _gen1P = millis(); _gen1O = 1;  } }
}
if(_RVFC2)
{_swi1=_gen1O;}
else
{_swi1=_RVFC1;}
digitalWrite(4, _swi1);
}
bool _isTimer(unsigned long startTime, unsigned long period )
  {
  unsigned long currentTime;
currentTime = millis();
if (currentTime>= startTime) {return (currentTime>=(startTime + period));} else {return
(currentTime >=(4294967295-startTime+period));}
  }
void _ressiveByteFromCommunication(byte resiveByte)
{
  if (_resiveModeCommunication==0) {
    if (resiveByte == 1) {
      _resiveModeCommunication=1;
     _resiveVariableStringCommunication="";
      return;
    }
  }
```

```
if (_resiveModeCommunication==1) {
  if((resiveByte == 1 )||(resiveByte == 3)|| (resiveByte == 4)) {
   _resiveModeCommunication=0;
  return;
  }
  else
  {
    if(resiveByte==2) {
      _resiveModeCommunication=2;
     _resiveVariableIndexCommunication=_resiveVariableStringCommunication.toInt();
      _resiveVariableStringCommunication="";
      return;
    }
    else {
      _resiveVariableStringCommunication+= char(resiveByte);
   return;
    }
  }
}
if (_resiveModeCommunication==2) {
  if((resiveByte == 1 )||(resiveByte == 2)|| (resiveByte == 4)) {
    _resiveModeCommunication=0;
  return;
  }
  else
  {
    if (resiveByte==3) {
       _resiveModeCommunication=3;
     _resiveVariableValueCommunication =_resiveVariableStringCommunication;
      _resiveVariableStringCommunication="";
   return;
    }
    else {
       _resiveVariableStringCommunication+= char(resiveByte);
     return;
    }
  }
}
if (_resiveModeCommunication==3) {
  if((resiveByte == 1 )||(resiveByte == 2)|| (resiveByte == 3)) {
    _resiveModeCommunication=0;
  return;
  }
  else
  {
    if (resiveByte==4) {
       _resiveModeCommunication=0;
    _fillRessiveVariableasFromCommunication();
      _resiveVariableStringCommunication="";
   return;      }
   else {
      _resiveVariableStringCommunication+= char(resiveByte);
```

```
        return;     }    }  }}
      void _fillRessiveVariableasFromCommunication()
      {
      long  _tempId= _resiveVariableStringCommunication.toInt();
      if ((_resiveVariableIndexCommunication == 1) && (_tempId == 122532102)) {_RVFC1
= ((_resiveVariableValueCommunication).toInt());}
      if ((_resiveVariableIndexCommunication == 2) && (_tempId == 122532102)) {_RVFC2
= ((_resiveVariableValueCommunication).toInt());}
      }
      int hexStrToInt(String instring)
      {
       byte len = instring.length();
       if  (len == 0) return 0;
       int result = 0;
       for (byte i = 0; i < 8; i++)   // только первые 8 цыфар влезуть в uint32
       {
         char ch = instring[i];
         if (ch == 0) break;
         result <<= 4;
         if (isdigit(ch))
           result = result | (ch - '0');
         else result = result | (ch - 'A' + 10);
       }
       return result;
      }
       void _esp8266WifiModuleClientReconnect()
      {
       if( _checkMacAddres(ESP8266ControllerWifiClient_mac)) {  wifi_set_macaddr(0,
const_cast<uint8*>(ESP8266ControllerWifiClient_mac));}
      if( ESP8266ControllerWifiClient_isDHCP) { WiFi.config(0U, 0U, 0U, 0U, 0U); } else {
      WiFi.config(ESP8266ControllerWifiClient_ip, ESP8266ControllerWifiClient_gateway,
ESP8266ControllerWifiClient_subnet, ESP8266ControllerWifiClient_dns ,
ESP8266ControllerWifiClient_dns );}
       WiFi.begin(ESP8266ControllerWifiClient_SSID,
ESP8266ControllerWifiClient_password);
      }
      bool _checkMacAddres(byte array[])
      {
       bool result = 0;
       for (byte i = 0; i < 6; i++)
       {
         if (array[i] == 255) {
           return 0;
         }
         if (array[i] > 0 ) {
           result = 1;
         }  }
       return result;}
```

Модуль А4 осуществляет контроль систем климат контроля, благодаря датчику температуры и влажности.

Данный модуль имеет представленный на рисунке выше.

Блок схема программного модуля позволяет как принимать сигналы с модуля A1, так и самостоятельно реагировать на изменение температуры включать и выключать сплит систему. данного модуля имеет небольшую схожесть с блок-схемой модуля A3.

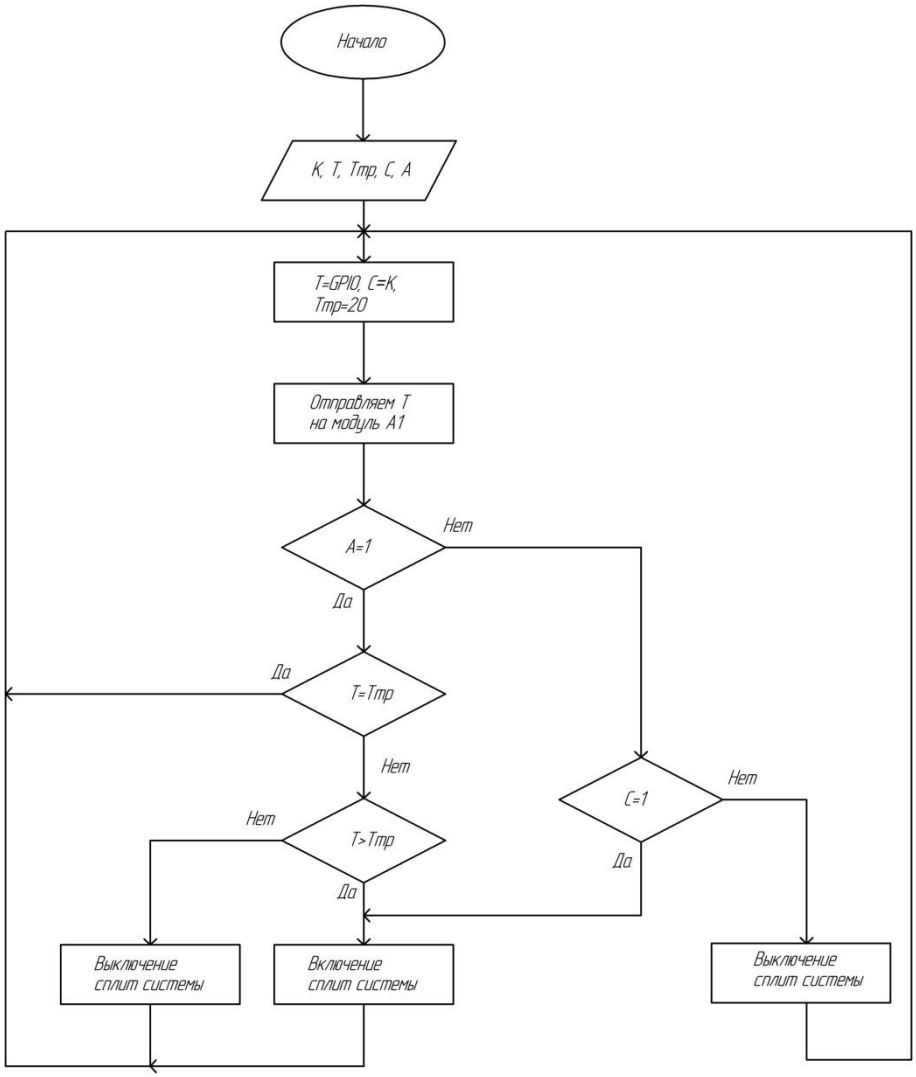

Рисунок 59 – Блок-схема программы модуля A4, C- сигнал контроля, T - температура показываемая модулем DHT11, Tmp - установленная температура спокойствия A- сигнал автоматического режима

Блок программы на рисунке 60 состоит из D2 - выхода GPIO0 управляющего реле, переключателя Switch, RVFC 1 - приходящего сигнал ручного управления, и RVFC 2 - приходящий сигнал автоматического управления, Заданная пользователем требуемая температура помещения, dht -

блок датчика температуры, передающий показания на логический блок сравнения.

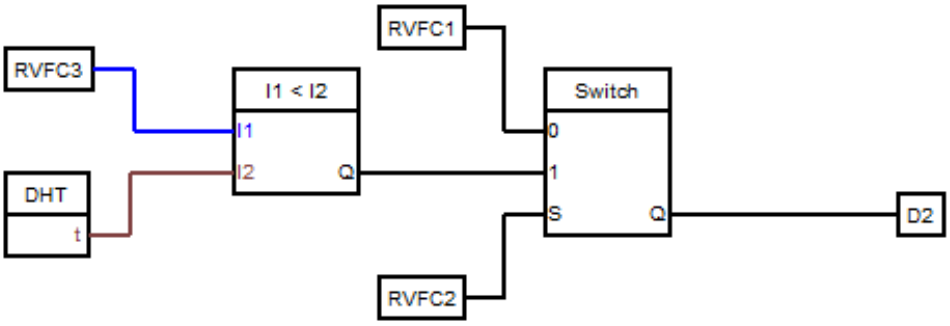

Рисунок 60 – программа применяющаяся в модуле A4

Последний модуль A5 отвечает за контроль уличного освещения, иными словами при малом уровне освещения вне помещения, будут загораться светодиоды отвечающие за внешнее освещение.

Схема данного модуля представлена выше на рисунке 55. Блок схема модуля A5 схожа с модулем A3 и имеет следующий вид (рисунок 61).

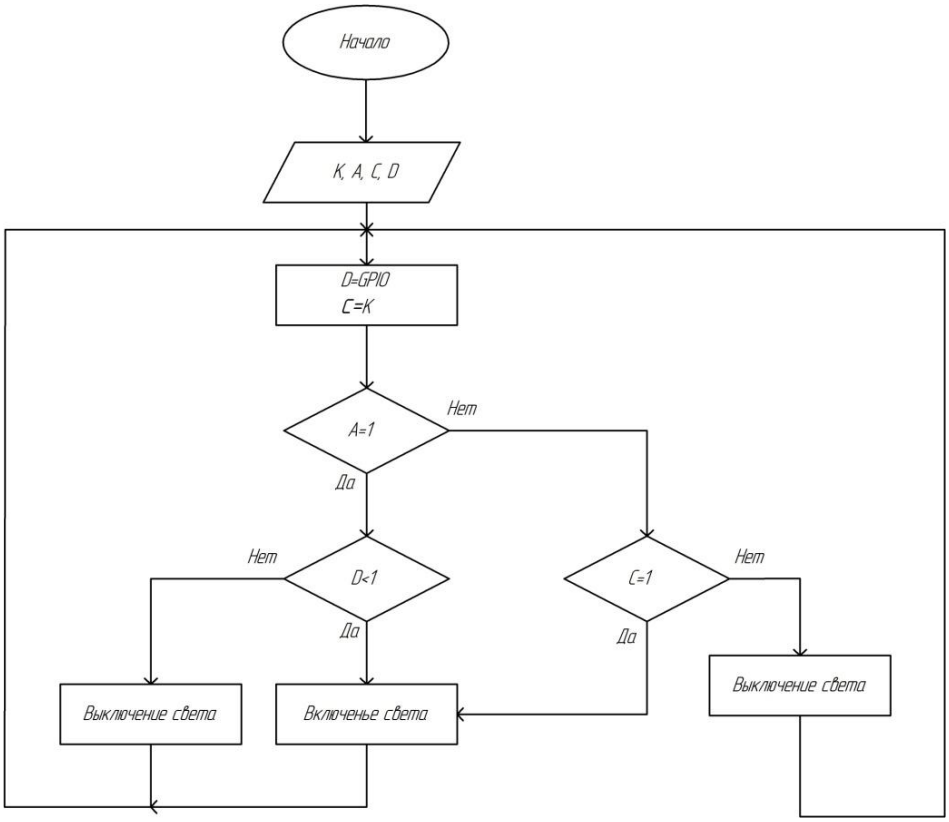

Рисунок 61 - Блок схема программного обеспечения на модуле A5

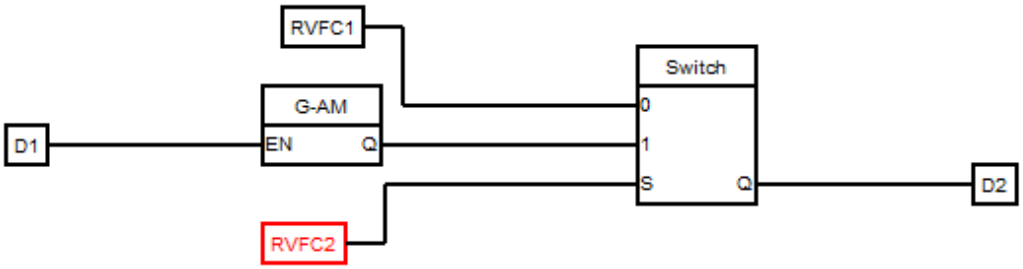

Рисунок 62 – Программа, применяющаяся в модуле A5

Данный блок состоит из D1 - вход GPIO2, D2 - выход GPIO0, G-AM асинхронный мультивибратор, RVFC 1 - приходящий сигнал ручного управления, а RVFC 2 - приходящий сигнал автоматического управления. Программный код при компиляции будет иметь схожее значение с кодом на блоке A3.

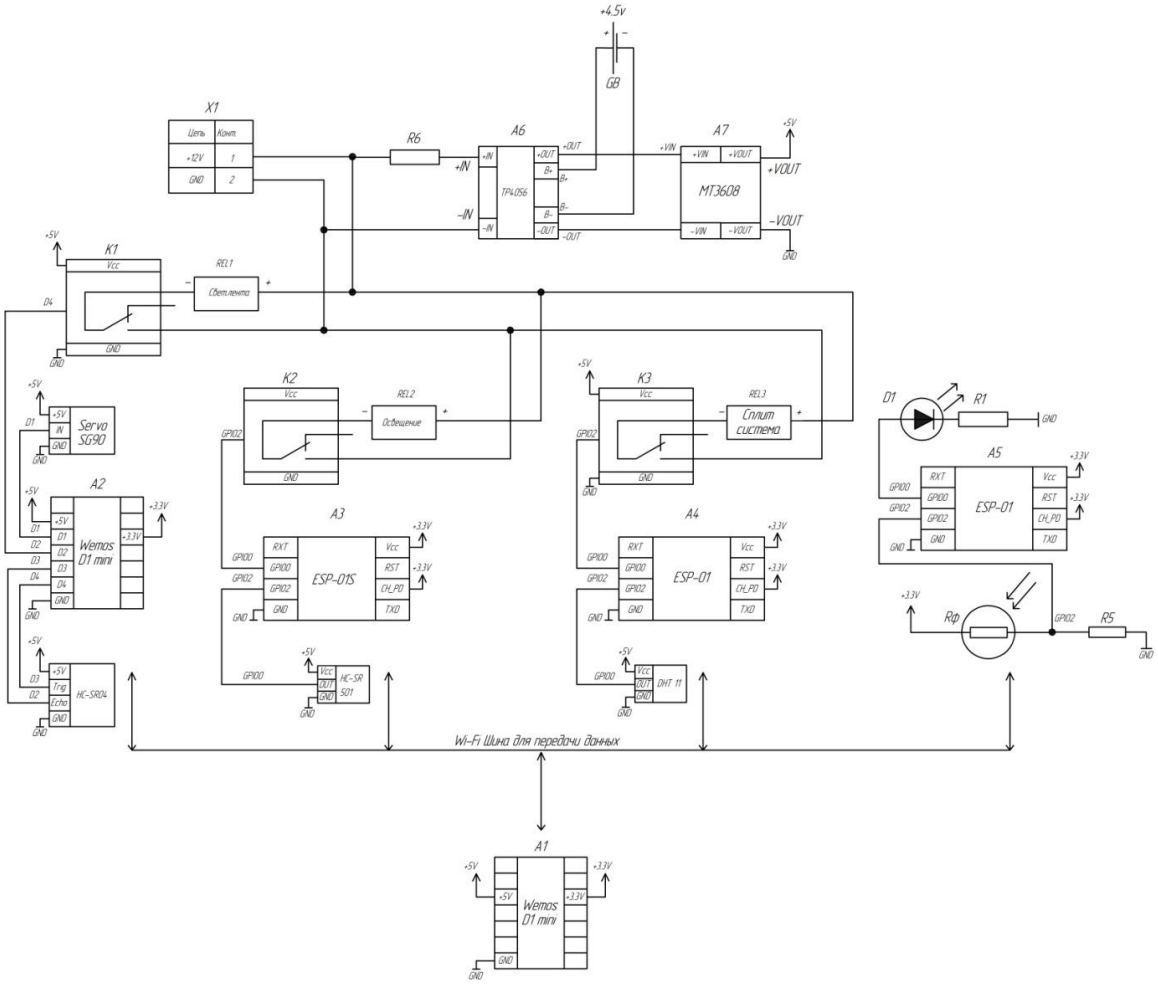

Рисунок 63 – Принципиальная схема стенда умный дом

В итоге будет разработан стенд децентрализованной системы типа «Умный дом» с независимыми модулями, рабочим Web-интерфейсом, использующий для передачи данных локальную сеть на базе WI-FI, с сетевой моделью TCP/IP. Структурная схема данного стенда имеет следующий вид.

Заключение

Ключевые выводы и итоги магистерской диссертации сводятся к нижеперечисленному:

- выполнен анализ современных решений систем умный дом;
- проанализированы существующие на данный момент типы передачи данных, а так же протоколы применяющиеся в них;
- предложен способ работы с новыми микроконтроллерами привычным методом без использования дополнительного ПО;
- подобрано новое ПО для работы с платами, основанными на микроконтроллерах ATmega и ESP8266;
- исследована возможность построения локальных сетей на базе микроконтроллеров, а так же проведена настройка и создана рабочая функциональная сеть, позволяющая не только передавать данные но и сохранять их в долгосрочную память;
- исследована возможность развертывания Web-серверов и Web-интерфейсов на печатных платах на базе ESP8266 и ATmega;
- по итогу магистерской диссертации был разработан и запрограммирован на работу стенд системы «Умный дом», а так же программное обеспечение позволяющее строить локальные сети на базе Wi-Fi сетей для обучения студентов.

Список используемой литературы

1. Архипов Г.В. «Системы для «интеллектуального» здания» - "СтройМаркет", № 45 .:Госэнергоиздат 1999 г. 218 с.

2. Майк Райли «Programming Your Home Automate with Arduino, Android, and Your Computer» - « The Pragmatic Bookshelf Dallas, Texas • Raleigh, North Carolina ».: LLC, 2012 г. 216с.

3. NetPing: конструктор для администратора и досуг для программиста [Электронный ресурс] URL: http://habrahabr.ru/post/118817/ (Дата обращения: 17.04.2017)

4. Умный дом своими руками [Электронный ресурс] URL: http://ab-log.ru/ (Дата обращения: 17.04.2017)

5. Умный дом на Arduino [Электронный ресурс] URL: http://a-bolshakov.ru/index/0-163 (Дата обращения: 13.04.2017)

6. Контроллер управления температурой [Электронный ресурс] URL: https://xively.com/?from_cosm=true (Дата обращения: 17.04.2017)

7. Каталог элементов управления [Электронный ресурс] URL: http://www.tesli.com/ru/service/automation/smart-house/ (Дата обращения: 17.04.2017)

8. Электронные компоненты fibaro [Электронный ресурс] URL: http://www.fibaro.com (Дата обращения: 20.03.2017)

9. Ю. Королев «УМНЫЙ ДОМ: приятная неизбежность» [Электронный ресурс] URL https://artelectronics.ru/posts/umnyj-dom-priyatnaya-neizbezhnost(Дата обращения: 17.05.2017)

10. Смирнов И.Г. «Должны ли кабельные системы быть структурированными?» [Электронный ресурс] URL https://www.dltens.ru/sks1.html

11. Петрушин С.А. Глушков В.А. «Источники энергии для индивидуальных домов» .: LAP Lambert Academic Publishing 2014г. 196 стр.

12. Тесля Е. ««Умный дом» своими руками. Строим интеллектуальную

цифровую систему в своей квартире» .:Питер. 2008 г. 196 стр.

13. Роберт Элсенпитер, Тоби Джордж Велт «Умный Дом строим сами» .:КУДИЦ-Образ 2016г. 619 с.

14. Кашкаров А. «Электронные схемы для «умного дома»» .:НТ Пресс 2007г 255с.

15. Дементьев А. ««Умный» дом XXI века» : Издательские решения 2016г 196 стр.

16. Сибикин М.Ю «Альтернативные источники энергии» .:ИП РадиоСофт 2014г. 248с.

17. Германович В., Турилин А.. «Альтернативные источники энергии. Практические конструкции по использованию энергии ветра, солнца, воды, земли, биомассы».: Наука и Техника 2014г. 196с.

18. «Концепция умного дома» [Электронный ресурс] URL https://nauchforum.ru/node/3560 (Дата обращения: 17.05.2017)

19. Гололобов В.Н. «Умный дом своими руками» .:НТ Пресс 2007г. 416с.

20. Марк Эдвард Сопер «Решения Умного дома» .:НТ Пресс 2007г. 432с.

21. Islanding the power grid on the transmission level: less connections for more security, 2016, Mario Mureddu, Guido Caldarelli, Alfonso Damiano, Antonio Scala & Hildegard Meyer-Ortmanns, 34797, http://www.nature.com/articles/srep34797

22. How dead ends undermine power grid stability, 2014, Peter J. Menck, Jobst Heitzig, Jürgen Kurths, Hans Joachim Schellnhuber, 3969, http://www.nature.com/articles/ncomms4969

23. Abruptness of Cascade Failures in Power Grids, 2014, Sakshi Pahwa, Caterina Scoglio & Antonio Scala, 3694, http://www.nature.com/articles/srep03694

24. Enhancing robustness of coupled networks under targeted recoveries, 2015, Maoguo Gong, Lijia Ma, Qing Cai, Licheng Jiao, 8439, http://www.nature.com/articles/srep08439

25. Enhancing synchronization stability in a multi-area power grid, 2016, Bing Wang, Hideyuki Suzuki& Kazuyuki Aihara, 26596, http://www.nature.com/articles/srep26596

26. А.С. Ковбасенко, И.А. Шипицин, Н.С. Семыкин «Обзор архитектур, применяющихся в системах «умный дом»» СТУДЕНЧЕСКИЕ ДНИ НАУКИ В ТГУ Тольятти, (2–27 апреля 2018 года) Сборник студенческих работ Часть 1 Научно-практическая конференция с 167-169.

27. А.С. Ковбасенко, И.В. Лаврова «проектирование высокоэффективной, масштабируемой корпоративной сети умного дома» / Сборник II Всероссийской научной конференции с международным участием "Информационные технологии в моделировании и управлении: подходы, методы, решения" / с 128-134.

28. Компьютерные сети. Настольная книга системного администратора 2016 – 912 с.

29. Компьютерные сети. Принципы, технологии, протоколы. 2017 – 992с.

30. Система «умный дом» — концепция умного дома — [Электронный Ресурс] — Режим доступа. — URL: http://energorus.com/sistema-umnyj-dom-koncepciya-umnogodoma/ (дата обращения 09.12.2018).

Приложение А

Листинг модуля создания сети и веб-сервера

```
#include <ESP8266WiFi.h>
#include <WiFiUdp.h>
#include <EEPROM.h>
extern "C" {
#include "user_interface.h";
}
bool ESP8266ControllerWifiClient_status = 1;
char ESP8266ControllerWifiAP_SSID[40] = "StandHome";
char ESP8266ControllerWifiAP_password[40] = "123456789";
bool ESP8266ControllerWifiAP_IsNeedReconect = 0;
bool ESP8266ControllerWifiAP_workStatus = 1;
IPAddress ESP8266ControllerWifiAP_ip(192, 168, 1, 1);
IPAddress  ESP8266ControllerWifiAP_dns (192, 168, 1, 1);
IPAddress  ESP8266ControllerWifiAP_gateway (192, 168, 1, 1);
IPAddress ESP8266ControllerWifiAP_subnet (255, 255, 255, 0);
uint8_t ESP8266ControllerWifiAP_mac[6] = {0x0, 0x0, 0x0, 0x0, 0x0, 0x0};
bool  ESP8266ControllerWifi_isNeededRestsrt = 0;
bool ESP8266_freeParam_259986625 = 0 ;
bool ESP8266_freeParam_95464520 = 1 ;
WiFiServer ESP8266ControllerWifi_tspWebServer(80);
WiFiClient ESP8266ControllerWifi_tspWebServer_client;
String ESP8266ControllerWifi_tspWebServer_client_buffer = "";
WiFiUDP Udp;
IPAddress _destIp1(192, 168, 1, 2);
String webSettingPageHeader1 = "HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<meta charset=\"utf-8\">\r\n<html>\r\n";
String webSettingPageFooterString = "Разработал Ковбасенко А.С.";
String webSettingPageHeaderString = "Web-интерфейс стенда умный дом";
String _ESP8266WebInterfacePageButton_save = "<input type=\"submit\" class = \"buttonFlp\"
value=\"Сохранить\"></form>";
String _ESP8266WebInterfacePageButton_reset = "<form action=\"\" method=\"POST\"><input
type=\"hidden\" name=\"resetController\" value=\"reset\"><input type=\"submit\" class =
\"buttonFlp\"  value=\"Перезагрузить\"></form>";
String _ESP8266WebInterfacePageButton_exit =  "<form action=\"\" method=\"POST\"><input
type=\"hidden\" name=\"exitUser\" value=\"exit\"><input type=\"submit\" class = \"buttonFlp\"
value=\"Выход\"></form>";
String webSettingPageMainStyle = " .menu {float:left; padding: 1%; margin: 1%;width:16%;
border: 1px solid black;border-radius: 8px;} .menuItem { font-weight: 600;font-family: 'Times New
Roman', Times, serif;} .header { padding: 10px; left: 0px; right: 0px; top: 0px; background:
#00FFFF; text-align: center;font-weight: 600;font-family: 'Times New Roman', Times, serif;font-
size: 100%; } .content { float:right; width:78%; } .footer{ float:left; padding: 1%; width:98%;
background: #00FFFF; margin-top: 1%; text-align: center;font-weight: 600;font-family: 'Times
New Roman', Times, serif;font-size: 150%;}.buttonFlp {width:150px;border-
radius:20px;background:#459DE5;color:#fff;font-size:12px;cursor:pointer;float:left; padding: 1%;
margin: 1%;}.buttonFlp:hover{background:#358DE5;}.buttonFlp:focus{outline:none;}";
int ESP8266ControllerWifi_currenUser = -1;
unsigned long ESP8266ControllerWifi_currenUserSeesStTi;
```

Продолжение приложения А

```
String ESP8266User_255215165_password = "admin";
String ESP8266User_255215165_login = "admin";
String ESP8266User_225118120_password = "123123";
String ESP8266User_225118120_login = "study";
String ESP8266User_148424023_password = "123123";
String ESP8266User_148424023_login = "techer";
bool isNeededCommitESP8266EEprom = 0;
bool _SVFC2 = 0;
bool _SVFC2needSend = 0;
bool _SVFC1 = 0;
bool _SVFC1needSend = 0;
void setup()
{
  EEPROM.begin(2);
  if (((readByteFromEEPROM(0, 0, 0x0))) != 179) {
    (updateByteToEEPROM(0, 0, 0x0, (179)));
    (updateBooleanToEEPROM(1, 0, 0x0, (0)));
    (updateBooleanToEEPROM(1, 1, 0x0, (1)));
    EEPROM.commit();
  }
  ESP8266_freeParam_259986625 = readBooleanFromEEPROM(1, 0, 0x0);
  ESP8266_freeParam_95464520 = readBooleanFromEEPROM(1, 1, 0x0);
  WiFi.mode(WIFI_AP);
  _esp8266WifiModuleApReconnect();
  ESP8266ControllerWifi_tspWebServer.begin();
  Udp.begin(8888);
}
void loop()
{ if (! (ESP8266ControllerWifi_currenUser == -1)) {
    if (_isTimer(ESP8266ControllerWifi_currenUserSeesStTi, 900000 )) {
      ESP8266ControllerWifi_currenUser = -1;
    }
  }
  ESP8266ControllerWifi_tspWebServer_client =
ESP8266ControllerWifi_tspWebServer.available();
  if (ESP8266ControllerWifi_tspWebServer_client) {
    String _WSCReqest = "";
    int _WSCPageNumber = 0;
    delay(5);
    while (ESP8266ControllerWifi_tspWebServer_client.available())
    {
      _WSCReqest.concat(char(ESP8266ControllerWifi_tspWebServer_client.read()));
    }
    ESP8266ControllerWifi_tspWebServer_client.flush();
    _WSCPageNumber  = _parseWebServerReqest(_WSCReqest);
    _sendWebServerPage(_WSCPageNumber);
  }
  if (isNeededCommitESP8266EEprom) {
    EEPROM.commit();
    isNeededCommitESP8266EEprom = 0;
  }
  if (ESP8266ControllerWifi_isNeededRestsrt) {
```

Продолжение приложения А

```
    ESP.restart();
  }
  if (ESP8266ControllerWifiAP_IsNeedReconect) {
    _esp8266WifiModuleApReconnect();
    ESP8266ControllerWifiAP_IsNeedReconect = 0;
  }
  //Плата:1
  bool _SVFC1temp;
  _SVFC1temp = ESP8266_freeParam_259986625;
  if (_SVFC1temp != _SVFC1) {
    _SVFC1 = _SVFC1temp;
    _SVFC1needSend = 1;
  }
  bool _SVFC2temp;
  _SVFC2temp = ESP8266_freeParam_95464520;
  if (_SVFC2temp != _SVFC2) {
    _SVFC2 = _SVFC2temp;
    _SVFC2needSend = 1;
  }
  if (_SVFC1needSend || _SVFC2needSend) {
    if (_SVFC1needSend || _SVFC2needSend) {
      Udp.beginPacket( _destIp1, 8888);
      if (_SVFC1needSend) {
        _SVFC1needSend = 0;
        Udp.write (1);
        Udp.print (1);
        Udp.write (2);
        Udp.print (ESP8266_freeParam_259986625);
        Udp.write (3);
        Udp.print (122532102);
        Udp.write (4);
      }
      if (_SVFC2needSend) {
        _SVFC2needSend = 0;
        Udp.write (1);
        Udp.print (2);
        Udp.write (2);
        Udp.print (ESP8266_freeParam_95464520);
        Udp.write (3);
        Udp.print (122532102);
        Udp.write (4);
      }
      Udp.endPacket();
    };
  };

}
bool _isTimer(unsigned long startTime, unsigned long period )
{
  unsigned long currentTime;
  currentTime = millis();
```

Продолжение приложения А

```
  if (currentTime >= startTime) {
    return (currentTime >= (startTime + period));
  } else {
    return (currentTime >= (4294967295 - startTime + period));
  }
}
void _sendWebServerPage(int sendPageNumber)
{
  if (sendPageNumber == -2) {
    _sendPasswordWebIntefacePage();
    return;
  }
  if (sendPageNumber == -1) {
    return;
  }
  if (sendPageNumber == 1) {
    _sendWebServerPage1();
    return;
  }
  _sendWebServerSend404Page();
}
void _sendWebServerSend404Page(void)
{
  ESP8266ControllerWifi_tspWebServer_client.print("HTTP/1.0 404 Not Found\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML><html><body><p>404 - Page not
fond</p></body></html>\r\n");
  delay(1);
  ESP8266ControllerWifi_tspWebServer_client.flush();
  ESP8266ControllerWifi_tspWebServer_client.stop();
}
int _parseWebServerReqest(String reqestAddres)
{
  int index;
  int result = 0;
  if ((reqestAddres.indexOf("POST")) == 0) {
    _parsePostWebServerReqest(reqestAddres);
    return -1;
  }
  index = reqestAddres.indexOf("/");
  reqestAddres = reqestAddres.substring(index + 1);
  index = reqestAddres.indexOf(" ");
  reqestAddres = reqestAddres.substring(0, index);
  if (reqestAddres == "start") {
    result = 1;
  }
  if ((result > -1) && (ESP8266ControllerWifi_currenUser == -1)) {
    result = -2;
  }
  if ((result > -1) && (! ( ESP8266ControllerWifi_currenUser == -1))) {
    ESP8266ControllerWifi_currenUserSeesStTi = millis();
  }
```

```
  return result;
}
void  _parsePostWebServerReqest(String reqestAddres)
{
  int index = reqestAddres.indexOf("\r\n\r\n");
  String paramsValue;
  String params = reqestAddres.substring(index + 4);
  while (params.length() > 0) {
    index = params.indexOf("&");
    if (index == -1) {
      paramsValue = params;
      params = "";
    } else {
      paramsValue = params.substring(0, index) ;
      params = params.substring(index + 1) ;
    }
    _processingPostWebServerReqestData(paramsValue);
  }
  index = reqestAddres.indexOf("Referer:");
  paramsValue = reqestAddres.substring(index + 8);
  index = paramsValue.indexOf("\r");
  paramsValue = paramsValue.substring(0 , index);
  params = "HTTP/1.0 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE
HTML><html><head> <script type=\"text/javascript\">document.location.replace(\"";
  params = params + paramsValue;
  params = params + "\");</script></head><body></body></html>\r\n";
  ESP8266ControllerWifi_tspWebServer_client.print(params);
  delay(1);
  ESP8266ControllerWifi_tspWebServer_client.flush();
  ESP8266ControllerWifi_tspWebServer_client.stop();
}
void _processingPostWebServerReqestData(String paramData)
{
  byte index = paramData.indexOf("=");
  if (index < 0) {
    return;
  }
  String key = paramData.substring(0, index);
  String value = convertPostData(paramData.substring(index + 1));
  if (key.equals(String("login_password"))) {
    _updateWebInterfaceUserWithData(key, value);
    return;
  }
  if (key.equals(String("exitUser"))) {
    ESP8266ControllerWifi_currenUser = -1;
    return;
  }
  if (key.equals(String("resetController"))) {
    ESP8266ControllerWifi_isNeededRestsrt = 1;
    return;
  }
```

Продолжение приложения А

```
  if (key.equals(String("parSetPage32861949E"))) {
    ESP8266_freeParam_95464520 = value.equals(String("on"));
    updateBooleanToEEPROM(1, 1, 0x0, ( value.equals(String("on"))));
  }
  if (key.equals(String("parSetPage103240179E"))) {
    ESP8266_freeParam_259986625 = value.equals(String("on"));
    updateBooleanToEEPROM(1, 0, 0x0, ( value.equals(String("on"))));
  }
}
String convertPostData(String value)
{
  String result = "";
  char symbol;
  byte code;
  int i = 0;
  String codeString;
  while (i < value.length()) {
    symbol = value.charAt(i);
    if (symbol == '+') {
      result.concat(' ');
    } else {
      if (symbol == '%')
      {
        codeString = "";
        i++;
        codeString.concat(value[i]);
        i++;
        codeString.concat(value[i]);
        code = hexStrToInt(codeString);
        Serial.println(codeString);
        Serial.println(code, HEX);
        result.concat(char(code));
      } else {
        result.concat(symbol);
      }
    }
    i++;
  }

  return result;
}
void _sendWebServerPage1(void)
{
  printToClient(webSettingPageHeader1, &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<head>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<head><style>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient((webSettingPageMainStyle + " "), &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
```

Продолжение приложения А

```
  printToClient("\"</style>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("</head>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);

  printToClient("<body>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<div class=\"header\">", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient(webSettingPageHeaderString, &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("</div>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<div class=\"content\">", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<form action=\"\" method=\"POST\">",
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( (_eSP8266SettingWebPageCheckBoxVisualElement("", "Shelk",
ESP8266_freeParam_259986625, "parSetPage103240179E")) ,
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( (_eSP8266SettingWebPageCheckBoxVisualElement("", "Auto",
ESP8266_freeParam_95464520, "parSetPage32861949E")) ,
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient(_ESP8266WebInterfacePageButton_save,
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient(_ESP8266WebInterfacePageButton_reset,
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient(_ESP8266WebInterfacePageButton_exit,
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("</div>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<div class=\"footer\">", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient(webSettingPageFooterString, &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("</div>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("</body></html>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  commitClient(&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
}
void _sendPasswordWebIntefacePage(void)
{
```

Продолжение приложения А

```
  printToClient( webSettingPageHeader1, &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( "<body><script type=\"text/javascript\">function wotkIn_form( )
{document.getElementById('lp1').value =
(document.getElementById('l1').value)+'*'+(document.getElementById('p1').value); return
true;}</script>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( "<div style=\"width:100%; text-align: center;\"><h2>Вы не
авторизованны!</h2></div> <div style=\"width:100%; text-align: center;\"><form action=\"\"
method=\"POST\" onsubmit=\"wotkIn_form( );\">",
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<h3> <label>логин:</label><br/><input  name=\"login\" type=\"text\" size=\"15\"
id=\"l1\" maxlength=\"50\" ><br/>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( "<label>пароль:</label><br/> <input   name=\"password\" type=\"password\"
size=\"15\"  id=\"p1\" maxlength=\"50\" ><br/><br/>",
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<input  name=\"login_password\"  id=\"lp1\" type=\"hidden\"  ><br/>",
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient( "<input   type=\"submit\"
value=\"Войти\"><br/><br/><h3></form></div></body></html>",
&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  commitClient(&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
}
void _updateWebInterfaceUserWithData(String key, String value)
{ byte index = value.indexOf("*");
  if (index < 0) {
    ESP8266ControllerWifi_currenUser = -1;
    return;
  }
  String templogin = value.substring(0, index);
  String tempPassword = value.substring(index + 1);
  if ((ESP8266User_255215165_password.equals(tempPassword)) &&
(ESP8266User_255215165_login.equals(templogin))) {
    ESP8266ControllerWifi_currenUser = 0;
    ESP8266ControllerWifi_currenUserSeesStTi = millis();
    return;
  }
  if ((ESP8266User_225118120_password.equals(tempPassword)) &&
(ESP8266User_225118120_login.equals(templogin))) {
    ESP8266ControllerWifi_currenUser = 1;
    ESP8266ControllerWifi_currenUserSeesStTi = millis();
    return;
  }
  if ((ESP8266User_148424023_password.equals(tempPassword)) &&
(ESP8266User_148424023_login.equals(templogin))) {
```

```
    ESP8266ControllerWifi_currenUser = 2;
```
Продолжение приложения А
```
    ESP8266ControllerWifi_currenUserSeesStTi = millis();
    return;
  }
  ESP8266ControllerWifi_currenUser = -1;
}
void  _sendNotAccesDenaidedPage(void)
{
  printToClient( webSettingPageHeader1, &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  printToClient("<body> <div style=\"width:100%; text-align: center;\"><h1>Доступ
запрещён!</h1></div></body></html>", &ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
  commitClient(&ESP8266ControllerWifi_tspWebServer_client,
&ESP8266ControllerWifi_tspWebServer_client_buffer);
}
void printToClient(String value, WiFiClient* sendClient, String* sendBuffer)
{
  for (int i = 0; i < value.length(); i++)
  {
    if (sendBuffer->length() > 800)
    {
      sendClient->print(*sendBuffer);
      *sendBuffer = "";
    }
    sendBuffer->concat(value.charAt(i));
  }
}

void commitClient(WiFiClient* sendClient, String* sendBuffer)
{
  if ((sendBuffer->length()) > 0) {
    sendClient->print(*sendBuffer);
  }
  *sendBuffer = "";
  sendClient->flush();
  sendClient->stop();
}
byte readByteFromEEPROM(int addres, byte bitAddres, byte chipAddres)
{
  return EEPROM.read(addres);
}
void updateByteToEEPROM(int addres, byte bitAddres, byte chipAddres, byte value)
{
  isNeededCommitESP8266EEprom = 1;
  return EEPROM.write(addres, value);
}
bool readBooleanFromEEPROM(int addres, byte bitAddres, byte chipAddres)
{
  byte temp = readByteFromEEPROM( addres,  bitAddres,  chipAddres);
  return  bitRead(temp, bitAddres);
```

```
}
```
Продолжение приложения А
```
void updateBooleanToEEPROM(int addres, byte bitAddres, byte chipAddres, bool value)
{
  byte temp = readByteFromEEPROM( addres,  bitAddres,  chipAddres);
  bitWrite(temp, bitAddres, value);
  updateByteToEEPROM( addres,  bitAddres,  chipAddres, temp);
}
String _eSP8266SettingWebPageAllVisualElementsHeader (String style, String label)
{
  String result = "<div";
  if (style.length() > 0) {
    result.concat(" class=\"");
    result.concat(style);
    result.concat("\"");
  }
  result.concat("> <p>");
  result.concat(label);
  result.concat("<br>");
  return result;
}
String _eSP8266SettingWebPageATextVisualElement (String style, String label, String value)
{
  String result = _eSP8266SettingWebPageAllVisualElementsHeader (style,  label);
  result.concat(value);
  result.concat("</p></div>");
  return result;
}
bool _eSP8266SettingWebPageCanViewPage(int pageIndex)
{
  if (ESP8266ControllerWifi_currenUser == 0) {
    return 1;
  }
  if (pageIndex == 0) {
    return 1;
  }
  return 1;
}
String _eSP8266SettingWebPageCheckBoxVisualElement (String style, String label, bool value,
String name)
{
  String result = "<div";
  if (style.length() > 0) {
    result.concat("class=\"");
    result.concat(style);
    result.concat("\"");
  }
  result.concat("> <label><input type=\"checkbox\" id=\"");
  result.concat(name);
  result.concat("1\"");
  if (value) {
    result.concat(" checked ");
```

```
  }
```
Продолжение приложения А
```
  result.concat("/>");
  result.concat(label);
  result.concat("</label><input type=\"hidden\" name=\"");
  result.concat(name);
  result.concat("\" id=\"");
  result.concat(name);
  result.concat("2");
  result.concat("\" value=\"");
  if (value) {
    result.concat( "on");
  } else {
    result.concat( "off");
  }
  result.concat( "\"><script> var el");
  result.concat(name);
  result.concat( " = document.querySelector('#");
  result.concat(name);
  result.concat("1");
  result.concat("'); el");
  result.concat(name);
  result.concat( ".onclick = function() {if (el");
  result.concat(name);
  result.concat( ".checked) { document.getElementById('");
  result.concat(name);
  result.concat("2");
  result.concat("').value ='on'; } else { document.getElementById('");
  result.concat(name);
  result.concat("2");
  result.concat("').value ='off';}} </script> </div>");
  return result;
}
int hexStrToInt(String instring)
{
  byte len = instring.length();
  if  (len == 0) return 0;
  int result = 0;
  for (byte i = 0; i < 8; i++)   // только первые 8 цыфар влезуть в uint32
  {
    char ch = instring[i];
    if (ch == 0) break;
    result <<= 4;
    if (isdigit(ch))
      result = result | (ch - '0');
    else result = result | (ch - 'A' + 10);
  }
  return result;
}
void _esp8266WifiModuleApReconnect()
{
  if ( _checkMacAddres(ESP8266ControllerWifiAP_mac)) {
```

```
    wifi_set_macaddr(1, const_cast<uint8*>(ESP8266ControllerWifiAP_mac));
```
Продолжение приложения А
```
  }
  WiFi.softAPConfig(ESP8266ControllerWifiAP_ip, ESP8266ControllerWifiAP_gateway,
ESP8266ControllerWifiAP_subnet);
  WiFi.softAP(ESP8266ControllerWifiAP_SSID, ESP8266ControllerWifiAP_password);
  if ( ! (_checkMacAddres(ESP8266ControllerWifiAP_mac))) {
    WiFi.softAPmacAddress(ESP8266ControllerWifiAP_mac);
  }
}
bool _checkMacAddres(byte array[])
{
  bool result = 0;
  for (byte i = 0; i < 6; i++)
  {
    if (array[i] == 255) {
      return 0;
    }
    if (array[i] > 0 ) {
      result = 1;
    }
  }
  return result;
}
```