Пермский филиал федерального государственного автономного образовательного учреждения высшего образования   
«Национальный исследовательский университет   
«Высшая школа экономики»

*Факультет экономики, менеджмента и бизнес-информатики*

Чепоков Елизар Сергеевич

**РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ПОДКЛЮЧЕНИЯ К ПЛАТФОРМЕ EMS INSYTE С ПОМОЩЬЮ КОНТРОЛЛЕРА ARDUINO**

*Курсовая работа*

студента образовательной программы «Программная инженерия»

по направлению подготовки *09.03.04 Программная инженерия*

Руководитель:

к.т.н, доцент кафедры информационных технологий в бизнесе

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

А. В. Кычкин

Пермь, 2020 год

**Аннотация**

Название: Разработка программы для подключения к платформе EMS INSYTE с помощью контроллера Arduino.

Автор: Чепоков Елизар Сергеевич, студент второго курса образовательной программы «Программная инженерия».

Руководитель: Кычкин Алексей Владимирович, доцент кафедры информационных технологий в бизнесе.

Данная курсовая работа посвящена разработке программы для контроллера Arduino

Работа включает 45 страниц формата А4, из них в основной части 28 страниц.

Основная часть работы включает в себя 10 иллюстраций и 7 таблиц.

Библиографический список состоит из 6 публикаций.

Работа включает в себя 5 приложений.

# Оглавление

[Оглавление 3](#_Toc40042744)

[Введение 5](#_Toc40042745)

[Глава 1. Анализ 7](#_Toc40042746)

[1.1. Изучение технологии MQTT протокола 7](#_Toc40042747)

[1.2. Обзор брокеров и клиентов 8](#_Toc40042748)

[1.3. Требования к разрабатываемой программе 10](#_Toc40042749)

[Глава 2. Проектирование 12](#_Toc40042750)

[2.1. Проектирование алгоритмов 12](#_Toc40042751)

[2.1.1. Создание диаграммы классов 17](#_Toc40042752)

[2.1.2. Создание диаграмм взаимодействия информационной системы 18](#_Toc40042753)

[2.2. Проектирование контроллера Arduino 19](#_Toc40042754)

[Глава 3. Разработка и тестирование. 21](#_Toc40042757)

[3.1. Выбор инструментов разработки 21](#_Toc40042764)

[3.2. Кодирование микроконтроллера. 23](#_Toc40042765)

[3.3. Тестирование программного продукта. 25](#_Toc40042766)

[3.3.1. Планирование тестирования. 25](#_Toc40042767)

[3.3.2. Планирование процедур контроля качества. 25](#_Toc40042768)

[3.3.3. Интеграционное тестирование. 25](#_Toc40042769)

[3.3.4. Системное тестирование. 25](#_Toc40042770)

[Заключение 27](#_Toc40042771)

[Список сокращений и условных обозначений 28](#_Toc40042772)

[Библиографический список 29](#_Toc40042773)

[Приложение А 30](#_Toc40042774)

[Приложение Б 34](#_Toc40042775)

[Приложение В 36](#_Toc40042776)

[Приложение Г 37](#_Toc40042777)

[Приложение Д 43](#_Toc40042778)

[Приложение Е 45](#_Toc40042779)

# Введение

В современном мире человек предъявляет все больше требований для удобной и комфортной жизни, в том числе и дом должен удовлетворять ряду требований:

* Климат – системы контроля температуры, влажности и качества воздуха;
* Безопасность – безопасность дома и жильцов;
* Бытовые требования – вода, газ, электричество;
* Комфорт – стиральные и посудомоечные машины, компьютеры и др.

С каждым годом темп жизни ускоряется, а вместе с ним увеличивается количество требований для комфортной жизни. В результате сложность техники и инженерное оснащение в домах растет. Таким образом появилось понятие «умный дом» - комплекс решений или устройств, облегчающие и автоматизирующие повседневные действия, для уменьшения затраченного времени и увеличения продуктивности. Системы «умный дом» чаще всего устанавливаются для всего жилья, с большим количеством устройств, из-за чего стоимость данных систем крайне велика.

Создание своего умного дома, с небольшим количеством устройств решает эту проблему. Для таких задач возможно использование программируемых контроллеров «Arduino». Данная тема актуальна, так как легкость в изучении и низкая стоимость контроллеров «Arduino» могут позволить сделать систему умного дома дешевле, что позволит более массово внедрять умные дома в повседневную жизнь.

Объектом исследования является система умного дома. Предметом исследования являются методы работы контроллера «Arduino» с онлайн брокером «EMS INSYTE».

Целью данной работы является создание программы для подключения к платформе «EMS INSYTE» с помощью контроллера «Arduino»

Задачи, для достижения поставленной цели:

1. Анализ взаимодействия клиента с брокером;
2. Изучение технологии MQTT;
3. Выявление преимуществ и недостатков систем аналогов;
4. Выбор инструментов разработки;
5. Формирование требований к разрабатываемому приложению;
6. Проектирование приложения;
7. Разработка и тестирование приложения.

При исследовании программ-аналогов будут использованы следующие методы исследования: анализ и сравнение, ранжирование, моделирование, дискретизация. При разработке приложения – объектно-ориентрованное программирование.

По итогу, после завершения данного проекта будет создана программа для контроллера «Arduino», которая сможет взаимодействовать с облачным брокером «EMS INSYTE», для управления устройствами умного дома. Так же данная работа сможет показать учащимся НИУ «ВШЭ», что можно настроить или реализовать «умный дом» своими руками.

# Анализ

В данной главе будут рассмотрены особенности работы и взаимодействия клиента и MQTT брокера, а также особенности MQTT протокола. Будут изучены аналоги брокеров и клиентов для работы с брокерами, выявлены их преимущества и недостатки. Результатом первой главы будут требования к программному продукту, а также диаграмма прецедентов.

## Изучение технологии MQTT протокола

Для связи контроллера с облачным брокером, потребуется научить контроллер выходить в сеть и сообщать о любых проблемах, возникших при подключении. Для данной задачи требуется выбрать сетевой протокол, который будет задавать направление в подключении.

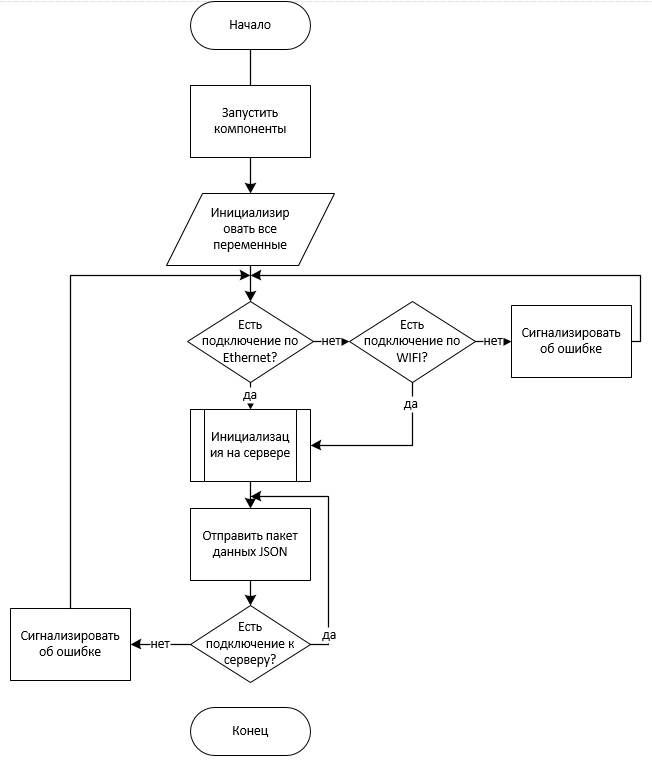
Сетевой протокол – это набор правил и действий в очередном порядке, позволяющий осуществить соединение и обмен данными между включенными в сеть устройствами. Минусом большинства протоколов является постоянная поддержка связи[5].

MQTT протокол – это упрощённый сетевой протокол работающий по принципу издатель-подписчик. Протокол этого типа ориентируется на простоту использования, легкую встраиваемость в любую систему и работу в условиях постоянной потери связи.

Так как планируется создать контроллер, который будет прост в создании и настройке и подключаться периодически, а не все время, то использоваться будет MQTT протокол[6].

Для правильной работы контроллера протокол должен выполнять следующие функции:

* + 1. Подключение к брокеру;
    2. Оповещение об отсутствии подключения;
    3. Подписка на топики;
    4. Отправка данных;
    5. Перепроверка подключения.

На рисунке 1.1 представлена упрощенная версия действий протокола для подключения к брокеру.

**Рисунок 1.1 – Протокол MQTT**

## Обзор брокеров и клиентов

Брокер является посредником для передачи данных между пользователем и устройством, такая система позволяет управлять предметами умного дома из любого места, при наличии подключения к сети «Интернет».

Целью данной курсовой работы является реализация подключения контроллера устройства, к удаленному брокеру «EMS INSYTE». Исходя из этого брокером будет являться система «EMS INSYTE»**.**

Написание отдельного брокера является крайне сложным и долгим проектом и качество результата будет низким и дорогим, так как будет требовать вложений для выхода в online. По этой причине данная курсовая работа не будет рассматривать этот вариант.

EMS INSYTE – автоматизированная система по управлению техническими устройствами, созданная Пермской компанией.

Данная система создана преимущественно для модерирования умным домом на практике. Так как система является коммерческим продуктом, она не приспособлена для легкого взаимодействия и обучения, и для отправки сообщения нужно точно знать куда и что отправлять. Данный подход обеспечивает безопасность умного дома и передаваемых данных, а так же сложность взаимодействия с ним.

Из клиентов для подключения к брокерам, самыми используемыми являются:

* MQTTspy;
* OpenHAB;
* IoT MQTT Panel.

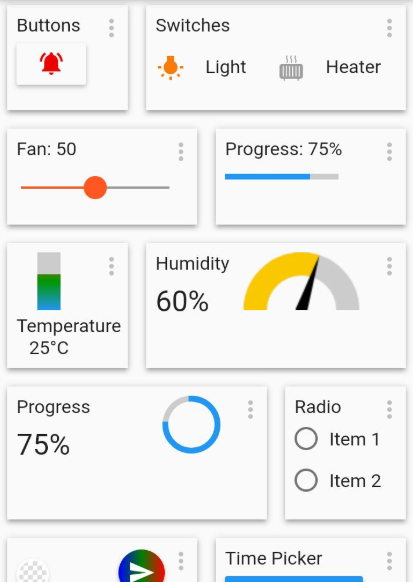
**MQTTspy**

MQTTspy – программа с открытым исходным кодом, предназначенная для мониторинга активности по MQTT протоколам. MQTTspy является одним из клиентов для обучения взаимодействия с брокером, из-за чего интерфейс программы является обычной консолью. Помимо обучения из плюсов стоит выделить большое количество документации, которая позволит разобраться, как работает подписка и публикация сообщений. Данный клиент в основном предназначен для локальной работы с брокером, который должен находится на том же компьютере, что и клиент, это является большим минусом для данной темы курсовой, так как будет сложно выстраивать взаимодействие с онлайн брокером.

**OpenHAB**

OpenHAB – система предоставляющая услуги брокера и клиента с открытым исходным кодом. Из плюсов стоит выделить легкость в использовании и наличие программы под любое устройство с любой операционной системой. Из минусов, отсутствует взаимодействие с неподдерживаемыми брокерами или устройствами, а также брокер должен быть локальным, хоть и с подключенным интернетом.

**IoT MQTT Panel**

IoT MQTT Panel – клиент для подключения к онлайн брокеру с мобильного устройства. Плюсами данной программы является подключение к любому MQTT брокеру, легкость в управлении и работа с мобильного устройства. Минусом клиента является поддержка только мобильных телефонов на платформе «Android». На рисунке 1.2 представлен интерфейс программы.

**Рисунок 1.2 – Интерфейс IoT MQTT Panel**

## Требования к разрабатываемой программе

Основываясь на вышеперечисленном разрабатываемая программа должна выполнять следующие требования:

* + 1. Программа должна работать автоматически с минимальным вмешательством пользователя;
    2. Программа должна подключаться по сети Internet;
    3. Программа должна выполнять протокол MQTT для стабильного и защищенного подключения;
    4. Программа должна подписываться на топики в облачном MQTT брокере;
    5. Программа должна отправлять и принимать данные.

Так же можно выделить входные и выходные данные:

**Входные данные:**

1. Данные для передачи измерений;
2. Значения для подключения к брокеру;
3. Сетевые данные для подключения к сети Internet.

**Выходные данные:**

1. Измерения для передачи в MQTT брокер;
2. Команды для подписки на топики брокера.

# Проектирование

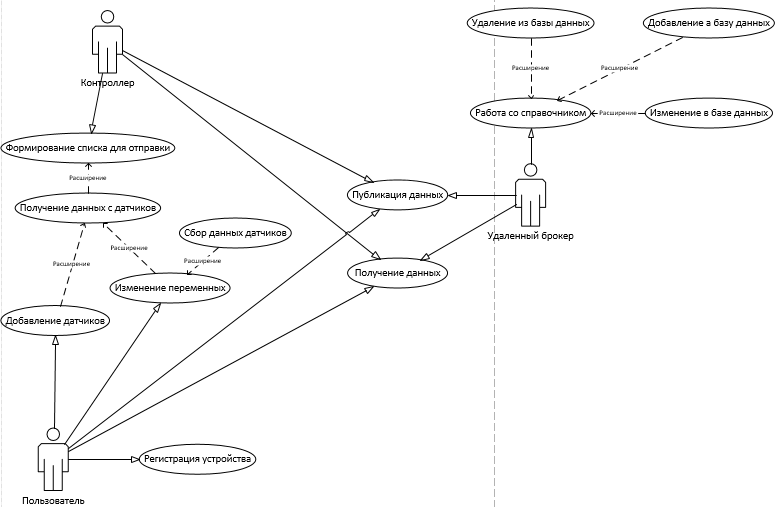
В данной главе будут рассмотрены алгоритмы и диаграммы, которые пoтребуются для создания программы и решения выдвинутых требований. Результатом данной главы является спроектированная программа.

## Проектирование алгоритмов

Проектирование алгоритма является одним из важнейших этапов создания любого программного продукта. Алгоритмы существуют для того чтобы иметь представление, как должна работать программа. Для создания алгоритмов могут использоваться как письменные алгоритмы, так и алгоритмы из рисунков, но чаще всего используют блок-схемы.

Для того чтобы представить и адаптировать стабильную работу контроллера, для начала нужно разобрать алгоритм полного подключения и работы программы. Данный алгоритм представлен в приложении Б.

**Рисунок 2.1 – Диаграмма прецедентов**



Так же, для четкого понимания работы системы была создана диаграмма прецедентов, представленная на изображении 2.1, созданная при помощи средств моделирования UML.

В таблицах 2.1 – 2.7 представлено расширенное описание всех прецедентов в формате таблиц. Полная диаграмма активностей представлена в приложении В.

**Таблица 2.2 – Прецедент «Изменение переменных»**

**Таблица 2.1 – Прецедент «Добавление датчиков»**

|  |  |
| --- | --- |
| Название прецедента: | Добавление датчиков |
| Актеры: | Пользователь |
| Краткое описание: | Пользователь добавляет разные датчики, переменные которых хочет отслеживать, с помощью клиента на телефоне. |
| Предусловия: | Желание пользователя добавить датчики для отслеживания. |
| Основной поток | |
| Пользователь: | Система: |
| Пользователь добавляет название датчику. | Система сохраняет датчик. |
| Пользователь выбирает какие переменные он хочет отслеживать.. |  |
| Подпотоки: | |
| нет | |
| Альтернативные потоки: | |
| Е1: Такой датчик уже существует. | |

|  |  |
| --- | --- |
| Название прецедента: | Изменение переменных |
| Актеры: | Пользователь. |
| Краткое описание: | Пользователь изменяет значения переменных, например, повышает влажность в доме. |
| Предусловия: | Желание изменить влажность в доме. |
| Основной поток: | |
| Пользователь | Система |
| Пользователь изменяет переменную. | Система записывает новую переменную для отправки. |
| Подпотоки: | |
| S1: Сбор данных датчиков. | |
| Альтернативные потоки: | |
| Нет. | |

|  |  |
| --- | --- |
| Название прецедента: | Регистрация устройств |
| Актеры: | Пользователь. |
| Краткое описание: | Пользователь вводит данные для регистрации контроллера на брокере. |
| Предусловия: | Необходимо добавить контроллер. |
| Основной поток: | |
| Пользователь: | Система: |
| Пользователь добавляет уникальный id устройства. | Система открывает форму для регистрации. |
| Пользователь вводит ip сервера. | Система проверяет данные. |
| Пользователь вводит имя контроллера. | Система сохраняет данные. |
| Подпотоки | |
| нет | |
| Альтернативные потоки | |
| E1: ошибка подключения к брокеру | |
| E2: данное устройство уже зарегистрировано | |

**Таблица 2.4 – Прецедент «Формирование списка для отправки»**

**Таблица 2.3 – Прецедент «Регистрация устройств»**

|  |  |
| --- | --- |
| Название прецедента: | Формирование списка для отправки |
| Актеры: | Контроллер. |
| Краткое описание: | Формирует пакеты JSON для отправки на брокер. |
| Предусловия: | Сообщение от брокера с командой отправить данные. |
| Основной поток: | |
| Контроллер: | Система: |
| Контроллер объединяет все переменные. | Система присылает команду для отправки файла. |
| Контроллер формирует сообщение в формате JSON файла. | Система оповещает о неправильном отправлении. |
| Подпотоки: | |
| S1: Получение данных с датчика. | |
| Альтернативные потоки: | |
| E1: Ошибка при отправке данных. | |

|  |  |  |  |
| --- | --- | --- | --- |
| Название прецедента: | | Публикация данных | |
| Актеры: | | Контроллер, Удаленный брокер, Пользователь. | |
| Краткое описание: | | Публикация любых данных или команд на топике, отправка команд. | |
| Предусловия: | | Запрос данных контроллером. Инициализация контроллера на брокере.  Запрос или изменение данных пользователем. | |
| Основной поток: | | | |
| Контроллер: | Пользователь: | Удаленный брокер: | Система: |
| Контроллер обновляет данные о датчиках. | Пользователь изменяет значения для контроллера или датчиков. | Удаленный брокер запрашивает обновленные данные. | Отправляет данные на брокер или на контроллер. |
| Контроллер инициализируется на брокере. | Пользователь запрашивает обновленные данные. |  | Отправляет информацию пользователю. |
| Контроллер запрашивает разрешение на отправку данных. |  |  |  |
| Подпотоки: | | | |
| Нет. | | | |
| Альтернативные потоки: | | | |
| Е1: Отправленные данные имеют неправильный формат. | | | |
| Е2: Ошибка при получении данных. | | | |
| Е3: Ошибка при инициализации на сервере. | | | |
| Е4: Ошибка при подключении к интернету. | | | |
| Е5: Отсутствует подключение к брокеру. | | | |

**Таблица 2.5 – Прецедент «Публикация данных»**

**Таблица 2.6 – Прецедент «Получение данных»**

|  |  |  |  |
| --- | --- | --- | --- |
| Название прецедента: | | Получение данных: | |
| Актеры: | | Контроллер, Удаленный брокер, Пользователь. | |
| Краткое описание: | | Контроллер, удаленный брокер или пользователь получает данные или команды. | |
| Предусловия: | | Брокер запрашивает данные о датчиках  Пользователь запрашивает данные о датчиках. | |
| Основной поток: | | | |
| Контроллер: | Пользователь: | Удаленный брокер: | Система: |
| Контроллер получает команду на отправку данных. | Пользователь получает данные датчиков. | Удаленный брокер получает информацию о контроллере. | Отправляет данные на брокер или на контроллер. |
|  |  | Удаленный брокер получает данные датчиков | Отправляет информацию пользователю. |
| Подпотоки: | | | |
| нет | | | |
| Альтернативные потоки | | | |
| Е1: Ошибка при получении данных. | | | |
| Е2: Ошибка при инициализации на сервере. | | | |
| Е3: Ошибка при подключении к интернету. | | | |
| Е4: Отсутствует подключение к брокеру. | | | |

**Таблица 2.7 – Прецедент «Работа со справочником»**

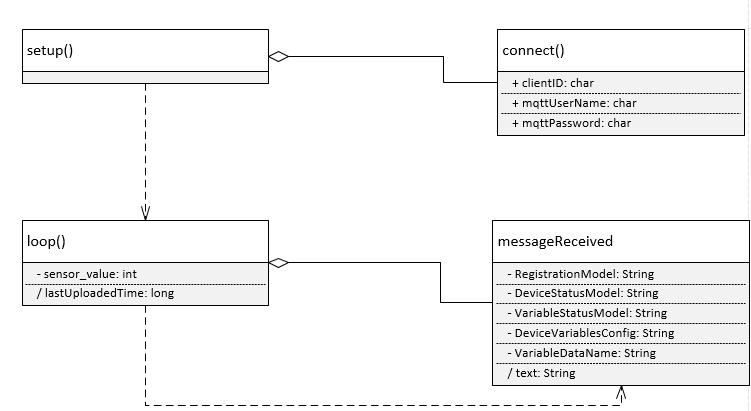
|  |  |
| --- | --- |
| Название прецедента: | Работа со справочником |
| Актеры: | Удаленный брокер |
| Краткое описание: | Брокер работает с полученными данными, изменяя переменные. |
| Предусловия: | Необходимость обновить или отправить данные. |
| Основной поток: | |
| Удаленный брокер: | Система: |
| Удаленный брокер вносит изменения в базу данных. | Система обрабатывает действия брокера. |
| Подпотоки: | |
| S1: Удаление из базы данных | |
| S2: Добавление в базу данных | |
| S3: Изменение в базе данных | |
| Альтернативные потоки: | |
| нет | |

По итогу спецификации требований было составлено техническое задание, текст которого приведен в приложении А.

### **Создание диаграммы классов**

Для проектирования модели данных использовались знания о предметной области, полученные на этапе анализа. В итоге для базы данных логистической компании было определено 4 сущности (см. Рисунок 2.2).

**Рисунок 2.2 – Диаграмма классов**



Так как система должна выполнять протокол MQTT для подключения к брокеру, то основными сущностями будут «connect» и «MessageReceived».

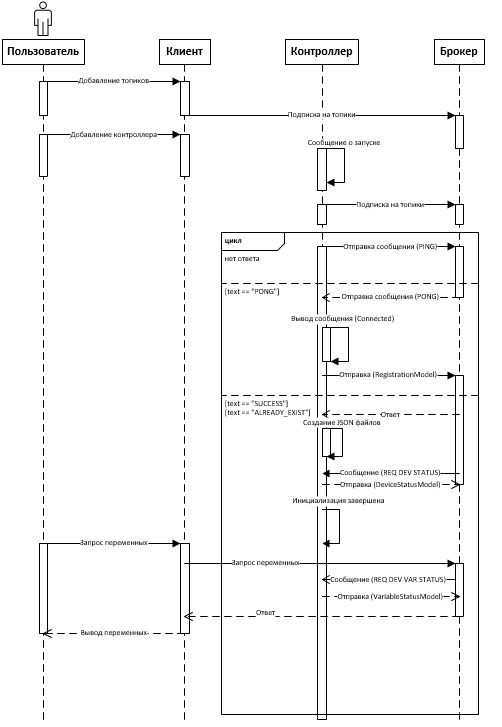
Сущности (loop и setup) необходимы для успешной инициализации всех переменных и запуска контроллера. Так как все переменные прописаны, как константы, сущность «Setup» будет лишь проводить включение контроллера и поочередно выполнять действия нужные при запуске. Сущность «Loop» является бесконечным циклом, где выход, это выключение контроллера, данная функция будет принимать данные и по необходимости подключать остальные участки кода.

Сущности (connect и messageReceived) предназначены для инициализации контроллера на брокере и получения сообщений. Сущность «connect» выполняется каждый раз, когда цикл перезапускается, для проверки подключения к сети «Интернет», а также проверки подключения к брокеру, данная сущность содержит в себе данные о пользователе, уникальный код контроллера, ip адрес и mac адрес, которые отправляются брокеру для инициализации. Сущность «messageReceived» задействуется при получении любого сообщения на контроллер и содержит JSON документы для отправки брокеру.

### **Создание диаграмм взаимодействия информационной системы**

Во время проектирования системы требуется построить модель поведения системы и её прецедентов, это необходимо для того, чтобы понимать, как будет работать система и в дальнейшем. Для построения данной модели была использована нотация UML. Чтобы охарактеризовать систему были выбраны диаграммы последовательностей, так как они помогают понять, как пользователь будет взаимодействовать с системой. Диаграмма последовательностей представлена на рисунке 2.3.

**Рисунок 2.3 – Диаграмма последовательностей**



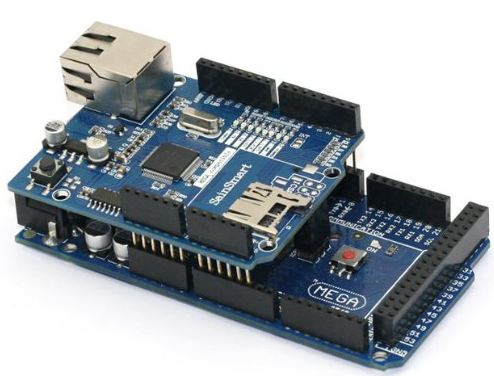
На данной диаграмме виден процесс работы и взаимодействия всех членов системы между друг другом

## Проектирование контроллера Arduino

Для полноценной автоматизированной работы контроллера требуется заранее выбрать компоненты, которые будут подключатся к сети и сигнализировать о выполнении протокола MQTT.

Контроллер данной курсовой работы будет состоять из двух частей, представленных на рисунке 2.4.

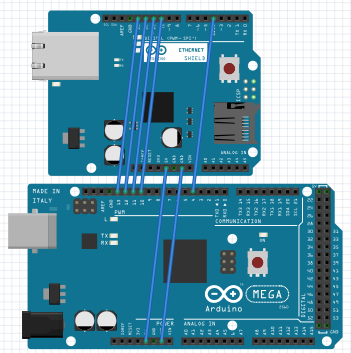
**Рисунок 2.4 – Arduino Mega и Ethernet shield**



Нижняя часть контроллера является полноценной платой «Arduino Mega 2560», построенной на микроконтроллере «ATmega2560». Данная часть будет выполнять исходный код и обрабатывать любые действия, проводимые пользователем.

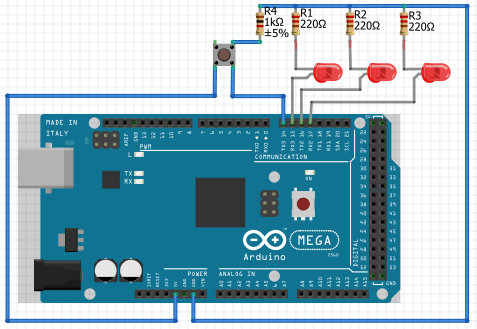
Верхняя часть контроллера, «Ethernet shield», поддерживает интернет соединение через канал «ethernet». Данная часть будет производить подключение к сети «Интернет» и отправлять данные брокеру. Подключение данной платы к «Arduino Mega 2560» представлено на рисунке 2.5.

**Рисунок 2.5 – схема подключения Ethernet shield**



Так же стоит добавить несколько диодов, которые будут сигнализировать о подключении или успешном отправлении данных, для этого будут использоваться обычные светодиоды, с прикрепленными к ним резисторами и кнопка.(см. рисунок 2.6).

**Рисунок 2.6 – Подключение диодов**








# Разработка и тестирование.

Данная глава обозревает реализацию и тестирование конечного продукта. Здесь рассматриваются все принципы создания программы. Результатом третьей главы является конечный продукт.

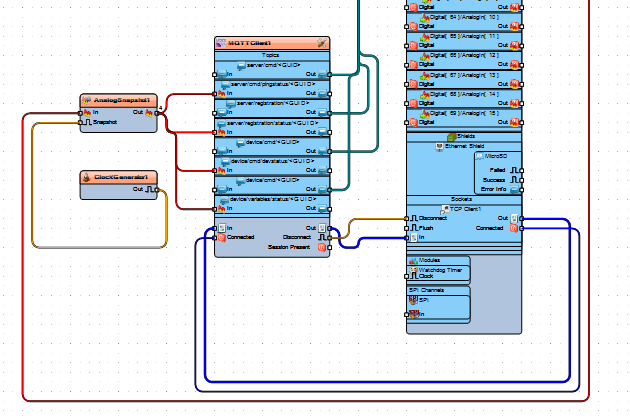


## Выбор инструментов разработки

Для начала работы требуется выбрать программное обеспечение, на котором будет выполнятся написание кода и загрузка кода на плату. Языком программирования для написания структуры кода будет статически типизированный язык программирования общего назначения С++, так как все контроллеры «Arduino» могут распознавать только этот язык, вставки из других языков программирования осуществляются при помощи дополнительных библиотек, расширяющих структуру кода С++.

Для программирования и последующей записи на память контроллера существует не так много вариантов, из которых стоит выделить:

1. Virtuino;
2. Visuino;
3. Arduino IDE;

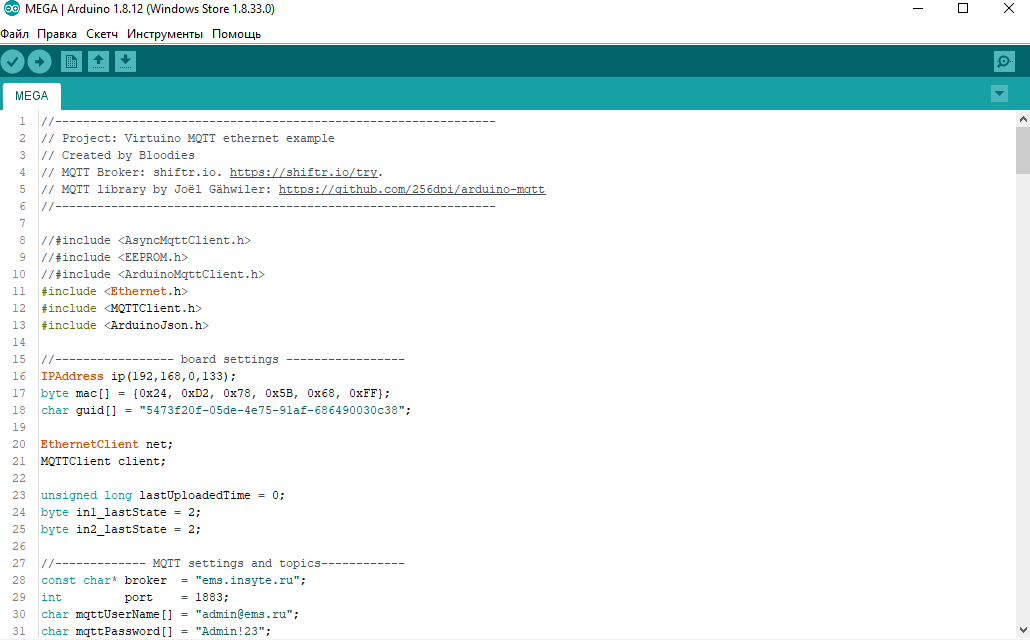
Virtuino – среда визуального программирования, использующая модульные блоки (см. Рисунок 3.1) для последующей конвертации в структурированный код.

**Рисунок 3.1 – Работа программы Virtuino**

Virtuino является неплохой средой разработки, но функционал данной программы оставляет желать лучшего. Данная программа подойдет только для схематичной визуализации взаимосвязи структуры кода между друг другом, но для полноценной отлаженной работы потребуется изменить созданный код, так как данная среда разработки забивает память ненужными элементами, которые мешают работе. Так же минусом является использование отдельного приложения Arduino IDE для программирования микроконтроллера.

Visuino – расширение для «Visual Studio», позволяющее работать с библиотеками «Arduino».

Данное расширение удобно тем, что не приходится скачивать дополнительные приложения для создания кода, но на этом плюсы заканчиваются. Из минусов стоит выделить сложную в использовании и не всегда работающую систему добавления библиотек «Arduino», из-за которой невозможно подключение отдельных библиотек для контроллера. Так же стоит выделить плохую работу программатора и компилятора, которые работают не со всеми структурами кода.

Arduino IDE - интегрированная среда разработки используемая ля написания и загрузки программ на Arduino-совместимые платы, а также, с помощью ядер сторонних производителей, на платы разработки других производителей. Интерфейс программы представлен на рисунке 3.2.

**Рисунок 3.1 – Интерфейс программы «Arduino IDE»**

Данное программное обеспечение создано специально для программирования микроконтроллеров, из-за чего существует большое количество документации и справочников, помогающие разбираться в программировании микроконтроллеров. Так же из плюсов стоит выделить компилятор, который позволяет работать вместе с загрузчиком, из-за чего можно проверить работоспособность микроконтроллера, подключив его по кабелю «USB».

На основе всего вышеперечисленного было выбрано программное обеспечение Arduino IDE, на основе языка программирования C++, по критериям, представленным ниже:

1. Большое количество документации и справочников;
2. Лёгкость и удобность программного обеспечения;
3. Лицензия программного обеспечения является бесплатной;
4. Работоспособность программатора и отладчика;

## Кодирование микроконтроллера.

Так как код в обычных программах на языке C++ и код для контроллера Arduino различается следует разобраться как работает код для контроллеров Arduino[1].

Для запуска самого контроллера служит функция setup() которая инициализирует все данные и начинает изначальный запуск.

Код для функции setup() представлен ниже:

void setup()

{

Serial.begin(9600);

Serial.println("Setup");

pinMode(MQTT\_CONNECTION\_LED\_PIN, OUTPUT);

pinMode(OUT1\_PIN, OUTPUT);

pinMode(OUT2\_PIN, OUTPUT);

pinMode(IN1\_PIN, INPUT);

Serial.begin(9600);

Ethernet.begin(mac);

client.begin(broker, net);

client.onMessage(messageReceived);

connect();

}

Для более удобного изменения данных все статичные переменные инициализируются как константы и не изменяются в будущем (см. Приложение Г).

Для запуска основного бесконечного цикла используется функция loop(). В данной функции программа раз за разом повторяет все, что написано в данной функции, пока контроллер не будет отключен от питания[2].

Остальные функции добавляются расширениями, вместе с добавлением библиотек. В данной программе были использованы библиотеки «MqttClient» и «Ethernet» для подключения «Ethernet Shield» и работы с протоколом MQTT.

Данные брокеру отправляются с помощью JSON сообщения, пример сообщения представлен ниже:

{"GUID": "5473f20f - 05de - 4e75 - 91af - 686490030c38",

"name": "Arduino - test",

"brand": "arduino",

"model": "MEGA 2560",

"serial\_number": "153255825568557862",

"last\_changed": "2020 - 05 - 05T13: 04: 04Z",

"start\_time": "2020 - 05 - 05T13: 04: 04Z",

"ip": "192.168.1.100",

"gateway": "192.168.1.100",

"mac": "24: D2: 78: 5B: 68: FF",

"timestamp": "2020 - 05 - 05T13: 04: 04Z",

"Internet\_channel": "ETH"}

Но так как обычный язык программирования не воспринимает данные формата JSON, было принято решение отправлять данные в формате String следующим образом:

String RegistrationModel = "{";

RegistrationModel += "\"GUID\":\"";

RegistrationModel += "5473f20f-05de-4e75-91af-686490030c38";

RegistrationModel += "\",\"name\":\"";

RegistrationModel += "Arduino - test";

RegistrationModel += "\",\"brand\":\"";

RegistrationModel += "arduino";

RegistrationModel += "\",\"model\":\"";

RegistrationModel += "MEGA 2560";

RegistrationModel += "\",\"serial\_number\":\"";

RegistrationModel += "153255825568557862";

RegistrationModel += "\",\"last\_changed\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"start\_time\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"ip\":\"";

RegistrationModel += "192.168.0.133";

RegistrationModel += "\",\"gateway\":\"";

RegistrationModel += "192.168.0.133";

RegistrationModel += "\",\"mac\":\"";

RegistrationModel += "24:D2:78:5B:68:FF";

RegistrationModel += "\",\"timestamp\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"Internet\_channel\":\"";

RegistrationModel += "ETH";

RegistrationModel += "\"}";

## Тестирование программного продукта.

Квалификационное тестирование программного обеспечения выполняется разработчиком в присутствии заказчика по всем разделам требований. Проверяется соответствие документации требованиям и ее адекватность разработанным компонентам[3]. Этот этап является завершающим этапом создания программного продукта, который позволяет выявить проблемы и в последующем решить их при помощи отладки.

### **Планирование тестирования.**

В программе есть 3 типа пользователей, из-за чего стратегию придется планировать исходя из входных данных трех акторов.

Так как программный продукт подразумевает собой программу для контроллера «Arduino», то и входных данных будет мало, так как единственными входными данными являются команды от пользователя брокеру и от брокера контроллеру. Следовательно, тестирование должно пройти в два этапа: интеграционное и системное тестирование.

### **Планирование процедур контроля качества.**

В результате тестирования должны быть исправлены все ошибки допущенные при разработке программы, исправлены и, если требуется оптимизированы.

### **Интеграционное тестирование.**

В данном тестировании необходимо проверить как модули взаимодействуют друг с другом.

Базой данных в данном тестировании будет являться брокер, который будет передавать настраиваемые переменные пользователем, так как все данные передаются в одном формате и максимально упрощены для пользователя, все ошибки на данном этапе исключены и могут возникнуть только при ошибках в системе контроллера или брокера, или при неправильно написанном файле программистом.

### **Системное тестирование.**

На этом этапе тестирования проверяется соответствует ли продукт заявленным требованиям. На этапе анализа были установлены требования и итоговый продукт им удовлетворяет.

По окончанию разработки системы, было составлено руководство пользователя, оно находится в приложении Д, и программиста в приложении Е. Листинг программы приведен в приложении Г.

# Заключение

По итогу данной работы, была разработана программа для подключения к «EMS INSYTE» с помощью контроллера «Arduino». Для этого были изучены основы создания контроллеров «Arduino» и рассмотрена литература на эту тему. Произведен разбор современного программного обеспечения, на основе которого делался вывод, как лучше написать программное обеспечение.

Положительным результатом проведенной работы является усовершенствование навыков написания программ на статически типизированном языке программирования C++ и разработка системы умного дома. Изученный материал по разработке программы для контроллера «Arduino», возможно пригодится в будущем, так как рынок «Умного дома» с каждым годом растет и развивается.

# Список сокращений и условных обозначений

Брокер – посредник для передачи данных между пользователями или устройствами.

# Библиографический список

1. alexgyver.ru [Электронный ресурс]. – Режим доступа: https://alexgyver.ru/ – Заглавие с экрана. – (Дата обращения: 20.01.2020)
2. wikihandbk.com [Электронный ресурс]. – Режим доступа: http://wikihandbk.com/wiki/Arduino – Заглавие с экрана. – (Дата обращения: 10.02.2020)
3. Плаксин М.А. Тестирование и отладка программ для профессионалов будущих и настоящих / М.А. Плаксин - М. : Лаборатория знаний, 2015. - 170 с.
4. Соснин О.М., «Основы автоматизации технологических процессов и производств: учебное пособие для студентов высших учебных заведений», г. Москва 2009, издательский центр Академия, 239с.
5. Bryan Boyd et al. Building Real-time Mobile Solutions with MQTT and IBM MessageSight. IBM Redbooks, 2014
6. Jeff Mesnil. Mobile and Web Messaging. O’Reilly Media, Inc., 2014 ISBN 978-1-4919-4480-6 — II. MQTT

# Приложение А

**Техническое задание**

1. **Введение**
   1. **Наименование программы**

«Программа для подключения к платформе ems insyte с помощью контроллера arduino».

* 1. **Краткая характеристика области применения**

Программный продукт представляет собой запрограммированный контроллер Arduino с возможностью подключаться к удаленному онлайн брокеру. Данная система создается как программное обеспечение для взаимодействия с умным домом для юридических или частных лиц.

Области применения программного обеспечения: программный продукт предназначен для:

1. Передачи данных с разных датчиков или устройств, подключенных к нему;
2. Принятия данных для датчиков или устройств, подключенных к нему;
3. Управления датчиками или устройствами, подключенными к нему с помощью удаленного доступа;
4. Обработки информации;

Программный продукт рекомендуется использовать системным администратором на коммерческих объектах или рядовым пользователем в домашней среде.

1. **Основания для разработки**

Программа создается в рамках написания курсовой работы по теме «Разработка программы для подключения к платформе EMS INSYTE с помощью контроллера Arduino» (факультет экономики, менеджмента и бизнес-информатики), в соответствии с учебным планом подготовки бакалавров по направлению подготовки 09.03.04 «Программная инженерия». Приказ № 8.2.2.1-38-25/04 от 07.10.2019.

1. **Назначение разработки**

Функциональное назначение:программный продукт осуществляет доступ к удаленному использованию системы «Умный дом», для взаимодействия как с отдельными устройствами, так и со всеми устройствами в целом.

Эксплуатационное назначение: программный продукт должен эксплуатироваться как в бытовой среде, рядовыми пользователями, так и в рабочей среде на объектах заказчика.

1. **Требования к программе**
   1. **Требования к функциональным характеристикам** 
      1. **Состав выполняемых функций:**

* Контроллер подключается к сети «Интернет»;
* Контроллер инициализируется на удаленном брокере «EMS INSYTE», отправляя данные о контроллере и подключенных к нему устройствах;
* Контроллер отправляет данные об устройствах каждый, установленный промежуток времени;
* Контроллер проверяет соединение с удаленным брокером;
* Контроллер подписывается на топики (каналы передачи данных MQTT протокола) для получения данных;
* Контроллер отправляет данные с помощью каналов отправления данных MQTT протокола;
* Пользователь подписывается на топики (каналы передачи данных MQTT протокола) для получения данных;
* Пользователь может запросить данные об устройствах;
* Пользователь может изменять данные устройств или отправлять команды на выполнение чего-либо;
  + 1. **Организация входных и выходных данных:**
* Ввод данных осуществляется с помощью изменения данных на устройствах, подключенных к контроллеру, либо с помощью панели управления на сайте брокера, либо с помощью приложения, подключенного к удаленному брокеру по системе издатель-подписчик;
* Входные и выходные данные передаются между брокером и контроллером с помощью MQTT протокола;
* Входные и выходные данные передаются в формате JSON сообщения;
  1. **Требования к обеспечению надежного функционирования программы**

Программный продукт не должен завершаться аварийно. Единственными способами разорвать связь в системе является отключение контроллера от сети «Интернет» или от питания, а также отключение удаленного брокера от сети «Интернет» или от питания.

1. **Условия эксплуатации**

Для управления системой необходима установка устройств и подключение их к контроллеру мастером. Для эксплуатации после установки достаточно мобильного телефона с доступом к сети «Интернет» или любого другого устройства с доступом к сети «Интернет».

* 1. **Требования к составу и параметрам технических средств**

Необходимо любое устройство, подключенное к сети «Интернет», оснащенный (32/64)-разрядным (x32/x64) процессором с тактовой частотой 1 ГГц или выше, 1 ГБ оперативной памяти (ОЗУ) и выше; имеющий минимум 500 МБ свободного пространства на жестком диске и предустановленным браузером или приложением для администрирования.

* 1. **Требования к информационной и программной совместимости**
     1. **Требования к информационным структурам и методам решения**

Требования к информационным структурам, а также к методам решения не предъявляются.

* + 1. **Требования к исходным кодам и языкам программирования**

Исходные коды программы должны быть написаны для контроллера Arduino на языке программирования C++

* + 1. **Требования к программным средствам, используемым программой**

Должен быть установлен любой браузер или любое приложение, выполняющее протокол MQTT.

* 1. **Требования к маркировке и упаковке**

Требования к маркировке и упаковке не предъявляются.

* 1. **Требования к транспортированию и хранению**

Специальные требования к транспортировке и хранению не предъявляются.

1. **Требования к программной документации**

Требования устанавливаются на основе правил написания и оформления курсовых работ студентов образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04 Программная инженерия, Пермь 2017.

1. **Технико-экономические показатели**

В рамках данной работы расчёт экономической эффективности не предусмотрен.

1. **Стадии и этапы разработки**

Предусматривается:

Анализ

Проектирование

Разработка

Тестирование

1. **Порядок контроля и приемки**
   1. **Виды испытаний**

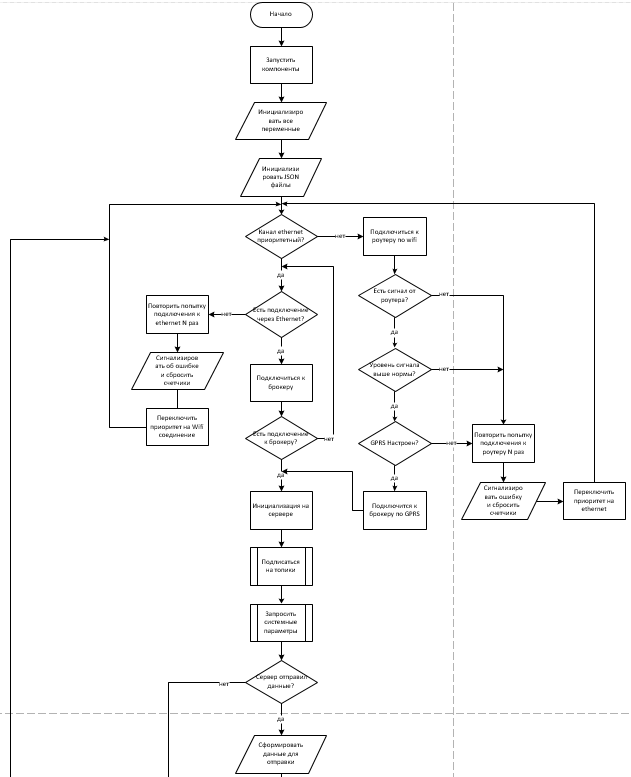
Производится проверка корректного выполнения программой заложенных в нее функций.

* 1. **Общие требования к приемке работы**

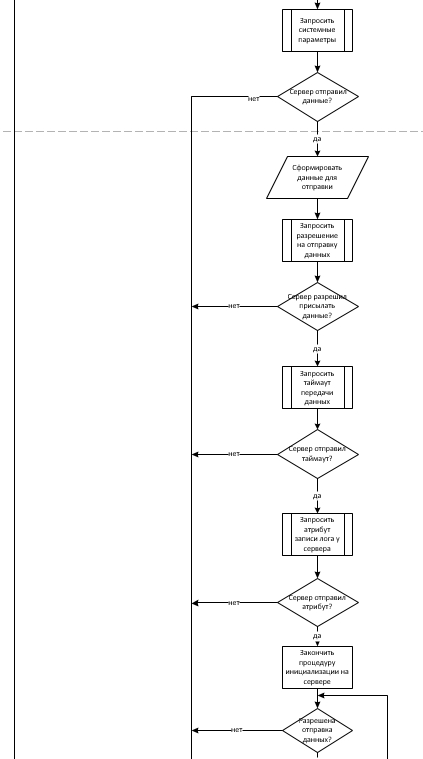
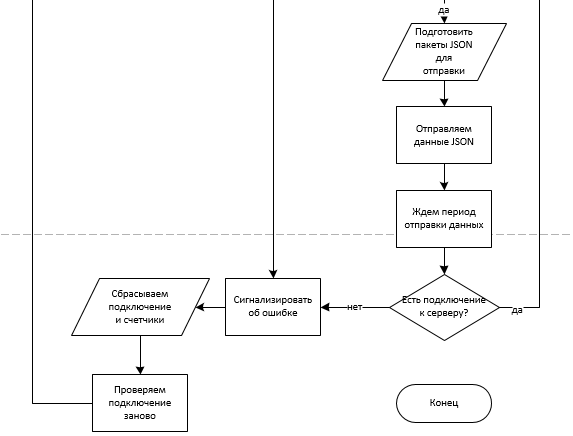
Общие требования к проверке работ указаны в правилах написания и оформления курсовых работ студентов образовательной программы бакалавриата «Программная инженерия» по направлению подготовки 09.03.04 Программная инженерия, Пермь 2017.

# Приложение Б

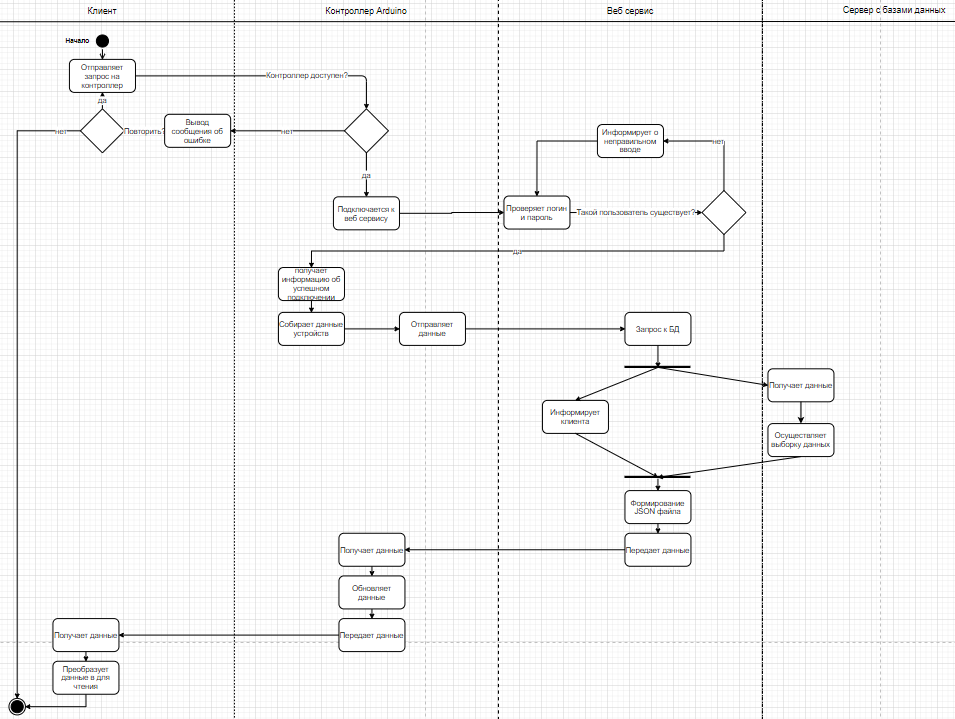
**Полный MQTT протокол**



**Рисунок Б.1. Полный MQTT протокол**



# Приложение В



**Диаграмма активностей**

**Рисунок В.1. Диаграмма отслеживания переменных контроллером**

# Приложение Г

**Листинг программы**

//---------------------------------------------------------------

// Project: Virtuino MQTT ethernet example

// Created by Bloodies

// MQTT Broker: shiftr.io. https://shiftr.io/try.

// MQTT library by Joël Gähwiler: https://github.com/256dpi/arduino-mqtt

//---------------------------------------------------------------

//#include <AsyncMqttClient.h>

//#include <EEPROM.h>

//#include <ArduinoMqttClient.h>

#include <Ethernet.h>

#include <MQTTClient.h>

#include <ArduinoJson.h>

//----------------- board settings -----------------

IPAddress ip(192,168,0,133);

byte mac[] = {0x24, 0xD2, 0x78, 0x5B, 0x68, 0xFF};

char guid[] = "5473f20f-05de-4e75-91af-686490030c38";

EthernetClient net;

MQTTClient client;

unsigned long lastUploadedTime = 0;

byte in1\_lastState = 2;

byte in2\_lastState = 2;

//------------- MQTT settings and topics------------

const char\* broker = "ems.insyte.ru";

int port = 1883;

char mqttUserName[] = "admin@ems.ru";

char mqttPassword[] = "Admin!23";

const char\* topic\_pub\_pingdev = "device/cmd/pingstatus/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_pingdev = "device/cmd/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_pingserv = "server/cmd/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_pingserv = "server/cmd/pingstatus/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_reg = "server/registration/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_reg = "server/registration/status/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_devstatus = "device/cmd/devstatus/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_devstatus = "device/cmd/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_variables = "device/variables/status/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_variables = "device/cmd/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_config = "device/variables/config/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_config = "device/variables/status/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_sub\_vardata = "server/status/5473f20f-05de-4e75-91af-686490030c38";

const char\* topic\_pub\_vardata = "device/variables/5473f20f-05de-4e75-91af-686490030c38";

const unsigned long mqttPostingInterval = 5L \* 1000L; // Post sensor data every 5 seconds.

//--------------------- JSON ---------------------

//------------------ PIN settings ------------------

#define MQTT\_CONNECTION\_LED\_PIN 4 // MQTT connection indicator

#define OUT1\_PIN 5 // Led or relay

#define OUT2\_PIN 6 // Led or relay

#define IN1\_PIN 7 // Button

void connect() {

digitalWrite(MQTT\_CONNECTION\_LED\_PIN, LOW); // Turn off the MQTT connection LED

Serial.print("\nConnecting.");

//--- create a random client id

char clientID[] = "ARDUINO\_0000000000"; // For random generation of client ID.

for (int i = 9; i < 19 ; i++) clientID[i] = char(48 + random(10));

while (!client.connect("Arduino", mqttUserName, mqttPassword)) {

Serial.print(".");

digitalWrite(MQTT\_CONNECTION\_LED\_PIN, !digitalRead(MQTT\_CONNECTION\_LED\_PIN));

delay(1000);

}

Serial.println("\nconnected!");

digitalWrite(MQTT\_CONNECTION\_LED\_PIN, HIGH);

client.subscribe(topic\_sub\_pingdev);

client.subscribe(topic\_sub\_pingserv);

client.subscribe(topic\_sub\_reg);

client.subscribe(topic\_sub\_devstatus);

client.subscribe(topic\_sub\_variables);

client.subscribe(topic\_sub\_config);

client.subscribe(topic\_sub\_vardata);

// client.unsubscribe(topic\_sub\_out2);

}

//----------------- board settings -----------------

// put your setup code here, to run once:

void setup()

{

Serial.begin(9600);

Serial.println("Setup");

pinMode(MQTT\_CONNECTION\_LED\_PIN, OUTPUT);

pinMode(OUT1\_PIN, OUTPUT);

pinMode(OUT2\_PIN, OUTPUT);

pinMode(IN1\_PIN, INPUT);

Serial.begin(9600);

Ethernet.begin(mac);

client.begin(broker, net);

client.publish(topic\_pub\_pingserv, "PING");

Serial.println("output -> PING");

client.onMessage(messageReceived);

connect();

}

//------------------ main program ------------------

// put your main code here, to run repeatedly:

void loop()

{

client.loop();

if (!client.connected())connect();

//---- MQTT upload

//if (millis() - lastUploadedTime > mqttPostingInterval)

//{

// int sensor1\_value = random(100); // replace the random value with your sensor value

// client.publish(topic\_pub\_sensor1, String(sensor1\_value), true, 1);

// int sensor2\_value = analogRead(A0);

// client.publish(topic\_pub\_sensor2, String(sensor2\_value), true, 1); // upload the analog A0 value

// lastUploadedTime = millis();

//}

//---- check if button is pushed

byte input1\_state = digitalRead(IN1\_PIN);

//if (input1\_state != in1\_lastState)

//{

// client.publish(topic\_pub\_in1, String(input1\_state), true, 1);

// delay(100);

// in1\_lastState = input1\_state;

//}

}

//------------------ main program ------------------

void messageReceived(String &topic, String &payload)

{

char RegistrationModel = "{";

RegistrationModel += "\"GUID\":\"";

RegistrationModel += "5473f20f-05de-4e75-91af-686490030c38";

RegistrationModel += "\",\"name\":\"";

RegistrationModel += "Arduino - test";

RegistrationModel += "\",\"brand\":\"";

RegistrationModel += "arduino";

RegistrationModel += "\",\"model\":\"";

RegistrationModel += "MEGA 2560";

RegistrationModel += "\",\"serial\_number\":\"";

RegistrationModel += "153255825568557862";

RegistrationModel += "\",\"last\_changed\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"start\_time\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"ip\":\"";

RegistrationModel += "192.168.0.133";

RegistrationModel += "\",\"gateway\":\"";

RegistrationModel += "192.168.0.133";

RegistrationModel += "\",\"mac\":\"";

RegistrationModel += "24:D2:78:5B:68:FF";

RegistrationModel += "\",\"timestamp\":\"";

RegistrationModel += "2020-05-05T13:04:04Z";

RegistrationModel += "\",\"Internet\_channel\":\"";

RegistrationModel += "ETH";

RegistrationModel += "\"}";

float battary = 100.00;

String DeviceStatusModel = "{";

DeviceStatusModel += "\"state\":\"";

DeviceStatusModel += "enabled";

DeviceStatusModel += "\",\"heath\":\"";

DeviceStatusModel += "ok";

DeviceStatusModel += "\",\"power\":\"";

DeviceStatusModel += "battery";

DeviceStatusModel += "\",\"battery\":\"";

DeviceStatusModel += battary;

DeviceStatusModel += "\",\"Internet\_channel\":\"";

DeviceStatusModel += "ETH";

DeviceStatusModel += "\"}";

int id\_var = 1;

bool enabled\_var = true;

int timeout\_var = 3000;

String VariableStatusModel = "{";

VariableStatusModel += "\"id\":\"";

VariableStatusModel += id\_var;

VariableStatusModel += "\",\"name\":\"";

VariableStatusModel += "light";

VariableStatusModel += "\",\"type\":\"";

VariableStatusModel += "IN";

VariableStatusModel += "\",\"data\_type\":\"";

VariableStatusModel += "STRING";

VariableStatusModel += "\",\"enabled\":\"";

VariableStatusModel += enabled\_var;

VariableStatusModel += "\",\"timeout\":\"";

VariableStatusModel += timeout\_var;

VariableStatusModel += "\"}";

int id\_conf = 1;

bool enabled\_conf = true;

int timeout\_conf = 3000;

bool section\_conf = true;

String DeviceVariablesConfig = "{";

DeviceVariablesConfig += "\"id\":\"";

DeviceVariablesConfig += id\_conf;

DeviceVariablesConfig += "\",\"enabled\":\"";

DeviceVariablesConfig += enabled\_conf;

DeviceVariablesConfig += "\",\"timeout\":\"";

DeviceVariablesConfig += timeout\_conf;

DeviceVariablesConfig += "\",\"section\":\"";

DeviceVariablesConfig += section\_conf;

DeviceVariablesConfig += "\"}";

int id\_data = 1;

String VariableDataName = "{";

VariableDataName += "\"id\":\"";

VariableDataName += id\_data;

VariableDataName += "\",\"timestamp\":\"";

VariableDataName += "2020-05-05T13:04:04Z";

VariableDataName += "\",\"value\":\"";

VariableDataName += "105.2";

VariableDataName += "\"}";

Serial.println("incoming: " + topic + " - " + payload);

if (topic == topic\_sub\_pingserv) //recieving from server/cmd/pingstatus/<GUID>

{

String text = payload;

if (text == "PONG")

{

Serial.println("Success");

client.onMessage(messageReceived);

client.publish(topic\_pub\_reg, RegistrationModel);

}

else

{

Serial.println("no answer");

delay(1000); //таймер на 10 мин

client.publish(topic\_pub\_pingserv, "PING");

client.onMessage(messageReceived);

}

}

if (topic == topic\_sub\_reg) //recieving from server/registration/status/<GUID>

{

String text = payload;

if (text == "SUCCESS")

{

Serial.println("Success");

client.onMessage(messageReceived);

}

else if (text == "ALREDY\_EXIST")

{

Serial.println("ALREDY\_EXIST");

client.onMessage(messageReceived);

}

else if (text == "ACCESS\_DENIED")

{

Serial.println("ACCESS\_DENIED");

client.publish(topic\_pub\_pingserv, "PING");

client.onMessage(messageReceived);

}

else if (text == "INTERNAL\_ERROR")

{

Serial.println("INTERNAL\_ERROR");

client.publish(topic\_pub\_pingserv, "PING");

client.onMessage(messageReceived);

}

else

{

delay(1000);

client.publish(topic\_pub\_pingserv, "PING");

client.onMessage(messageReceived);

}

}

if (topic == topic\_sub\_devstatus) //recieving from device/cmd/<GUID>

{

String text = payload;

if (text == "REQ DEV STATUS ")

{

client.publish(topic\_pub\_devstatus, DeviceStatusModel);

}

}

if (topic == topic\_sub\_variables) //recieving from device/cmd/<GUID>

{

String text = payload;

if (text == "REQ DEV VAR STATUS ")

{

client.publish(topic\_pub\_devstatus, VariableStatusModel);

}

}

if (topic == topic\_sub\_config) //recieving from device/variables/config/<GUID>

{

String text = payload;

if (text == "")

{

}

else

{

}

}

if (topic == topic\_sub\_vardata) //recieving from server/status/<GUID>

{

String text = payload;

if (text == "")

{

}

else

{

}

}

if (topic == topic\_sub\_pingdev) //recieving from device/cmd/<GUID>

{

String text = payload;

if (text == "")

{

}

else

{

}

}

}

# Приложение Д

**Руководство пользователя**

1. **Введение**
   1. **Область применения**

Областью применения программы является система умного дома для коммерческого или общего пользования. Основная цель программы – автоматизировать и облегчить использования разных устройств умного дома.

* 1. **Краткое содержание возможностей**

Данная система представляет собой контроллер на базе платформы Arduino с подключенными устройствами, управление которыми осуществляется с помощью мобильного приложения или личного кабинета на сайте «EMS INSYTE».

* 1. **Уровень подготовки пользователя**

Для управления системой необходима установка контроллера и устройств и подключение их к сети «Интернет», предоставляемая компанией.

* 1. **Перечень эксплуатационной документации**

Настоящее руководство пользователя и руководство программиста.

1. **Назначение и условия применения**
   1. **Виды деятельности, функции**

Программный продукт является контроллером для отправки и принятия изменений, подключенных устройств, от удаленного брокера компании «EMS INSYTE». Оно позволяет добавлять, модифицировать и удалять данные, выполнять запросы к данным. В целом программа служит инструментом для автоматизации устройств умного дома.

* 1. **Программные и аппаратные требования к системе**

Необходимо любое устройство с предустановленным браузером или приложением для отправки протоколов MQTT и выходом в интернет.

1. **Подготовка к работе**
   1. **Состав дистрибутива**

В состав дистрибутива входит контроллер Arduino и устройства умного дома.

* 1. **Запуск системы**

Запуск системы производит установщик, представленный от компании.

* 1. **Проверка работоспособности системы**

Программное обеспечение работоспособно, если в результате действий запуска горит красный светодиод.

1. **Описание операций**
   1. **Вход в систему**

Вход в систему производится автоматически при запуске системы с заранее выделенными логином и паролем.

* 1. **Внесение изменений в системе**

Условия выполнения: для добавления в базу данных информации о изменении переменных измените переменные с помощью мобильного приложения или на сайте «EMS INSYTE».

Подготовительные действия: авторизация в системе.

Основные действия: изменение переменных.

Заключительные действия не требуются.

Ресурсы, расходуемые на операцию: от секунды до минуты времени, ресурсы центрального процессора. Продолжительность выполнения операции зависит от интернет соединения и скорости отклика брокера.

* 1. **Просмотр информации об подключенных устройствах**

Условия выполнения: наличие подключения к сети «Интернет».

Подготовительные действия: авторизация в системе.

Основные действия: зайти в свой кабинет на сайте «EMS INSYTE», зайти в раздел устройства.

Заключительные действия: по завершении просмотра закрыть окно.

Ресурсы, расходуемые на операцию: от секунды до минуты времени, ресурсы центрального процессора. Продолжительность выполнения операции зависит от интернет соединения и скорости отклика брокера.

# Приложение Е

**Руководство разработчика**

1. **Общие сведения**
2. Документация: программная.
3. Предмет: программа для подключения к платформе ems insyte с помощью контроллера arduino.
4. Аудитория: С++ программисты, программисты контроллеров Arduino.
5. Задачи: обеспечить программисту возможность решать задачи, которые могут быть перед ним поставлены в отношении программы.
6. Стандарты: ГОСТ 19.504-79
7. **Цели и задачи**

Программный продукт по своему основному назначению является контроллером для отправки и принятия данных путем издатель-подписчик MQTT протокола.

1. **Назначение и условия применения программы**
   1. **Назначение программы**

Программный продукт создан для автоматизации и упрощения в управлении устройствами умного дома.

* 1. **Функции, выполняемые программой**

Программа работает с удаленным брокером по принципу издатель-подписчик MQTT протокола. Программный продукт позволяет изменять, добавлять или удалять данные устройств, подключенных к нему.

* 1. **Условия, необходимые для выполнения программы**

Необходимо любое устройство с предустановленным браузером или приложением для отправки протоколов MQTT и выходом в интернет.

1. **Характеристика программы**
   1. **Описание основных характеристик программы**

Скорость работы контроллера зависит от подключения к интернету и скорости ответа брокера.

* 1. **Особенности программы**

Программная система работает с удаленным брокером по сети «Интернет».

1. **Обращение к программе**

Запуск программы осуществляется сразу после установки и настройки контроллера.

1. **Входные и выходные данные**

Входные данные для контроллера от брокера и от контроллера к брокеру представлены в виде JSON списка.

1. **Сообщения**

При отсутствии какой-либо индикации проверить настройку контроллера.

При работе красного светодиода контроллер подключен к удаленному брокеру.

При работе желтого светодиода контроллер не подключен к удаленному брокеру либо отсутствует подключение к интернету.