МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ

Кафедра web-технологий и компьютерного моделирования

ВЗАИМОДЕЙСТВИЕ WEB-ПРИЛОЖЕНИЯ И КОНТРОЛЛЕРА ARDUINO UNO С ИСПОЛЬЗОВАНИЕМ ПЛАТЫ РАСШИРЕНИЯ WIFI SHIELD

Курсовая работа

Головчик Ксении Геннадьевны
студентки 4 курса 2 группы

Научный руководитель:
кандидат физ.-мат. наук,
доцент кафедры web-технологий
и компьютерного моделирования
Суздаль Станислав Валерьевич

Минск, 2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В наше время высоких технологий уже сложно представить жизнь без интернета. Мы общаемся в социальных сетях, мы находим информацию, покупаем продукты и вещи. Благодаря всему этому веб-технологии занимают огромное место в нашей жизни, развиваясь и изменяясь.

Однако кроме веб-технологий нас окружает много техники, мобильных устройств, телевизоров, бытовых устройств, машин. Всё это делает нашу жизнь проще и удобнее.

Но что если мы объединим две эти сферы, пользовательский интерфейс и технические устройства. Мы получим востребованное в наше время направление управления умным домом, техникой. По средствам понятного для пользователя интерфейса, он может управлять сложными приборами.

Однако такие взаимодействия по большинству своему направлены на бытовые и повседневные потребности. А что если обычному человеку позволить управлять светодиодной панелью, которая установлена на улице города. Это давало бы возможность необычным образом поздравить близкого человека с праздником или же просто сделать приятное себе и окружающим. Такая возможность привлекла бы туристов и стала бы достопримечательностью города.

Поэтому я поставила перед собой ряд задач, которые помогут достичь мне главной цели: “Создать удобное веб-приложение для взаимодействия с контроллером Arduino, который управляет светодиодами, зажигая их в заданной последовательности.”

Для этого были решены ряд задач:

1. Выяснить какие веб-технологии востребованы на данный момент.
2. Понять принцип их работы и применение на реальном проекте.
3. Понять принцип работы контроллера Arduino.
4. Взаимодействовать с ним используя определённые программы и технологии.
5. Выяснить и применить на практике каким образом может происходить взаимодействие между веб-приложением и контроллером.

ГЛАВА 1
ПОСТАНОВКА ЗАДАЧИ

1.1 Web-приложения аукционы

Первый шаг к написанию приложения было изучение уже существующих ресурсов со схожей тематикой. Распространены сервисы-аукционы по продаже автомобилей, вещей, коллекций. (Рисунок 1.1 Приложение А)

Единственный отличный от обычных аукционных сайтов для Беларуси это сайт maesens.by[1]. **МаеСэнс** — созданная в Беларуси социальная интернет-платформа в формате благотворительного аукциона встреч. Сервис МаеСэнс предлагает зарегистрированным пользователям нестандартный формат вовлечения в благотворительность, в основе которого лежат механизмы аукционной торговли, социальной сети, виртуальной службы знакомств и краудфандинга.[2] (Рисунок 1.2 Приложение А)

Так же существует множество аукционов в социальных сетях. Главная направленность которых за репост или отметку “мне нравится” записи в определённое время разыграть приз, тем самым привлечь больше публики на свой ресурс. (Рисунок 1.3 Приложение А)

Поэтому веб-приложение аукцион “Lucidum” для развлечения будет довольно новым для Беларуси. Отличие его в том, что он направлен на улучшение развлекательной составляющей города, сервис не направлен на получение огромной прибыли, скорее как благотворительность. Идея заключается в написании веб-приложения аукциона на который позволит изобразить собственно придуманный узор на светодиодной панели. Люди участвуют в аукционе предлагая свой узор и ставя сумму, которую они готовы за это заплатить. Контролировать цензурность узора будет администратор приложения.

1.2 Основные направления использования контроллеров

Так же был изучен белорусский рынок на предмет использования контроллеров в развлекательных или бытовых целях.

Самое востребованное и всеобще известное направление это умный дом. **Умный дом** — жилой дом современного типа, организованный для проживания людей при помощи автоматизации и высокотехнологичных

устройств. Под «умным» домом следует понимать систему, которая обеспечивает безопасность и ресурсосбережение для всех пользователей. В простейшем случае она должна уметь распознавать конкретные ситуации, происходящие в доме, и соответствующим образом на них реагировать: одна из систем может управлять поведением других по заранее выработанным алгоритмам.[3]

Конечно же контроллеры используются в развлекательных целях, наиболее известный пример это поющие и танцующие фонтаны, которые работают за счёт управления клапанами и системой подачи воды главным контроллером. Однако явного управления через веб-приложение мной не было встречено.

Поэтому управление контроллером по средствам протокола передачи данных через веб-приложение показалась интересной темой для курсовой работы.

1.3 Бизнес анализ основных возможностей сервиса

Бизнес-анализ — основа успешного маркетингового планирования. Достоверные данные, организованные и упорядоченные в осмысленную структуру, обеспечат чрезвычайно глубокое понимание целевого рынка и покупательского поведения потребителей.[4]

Воспользовавшись данным определением, был составлен краткий анализ сервиса:

1. Описание. Веб-приложение “Lucidum” создано для возможности удалённого управления светодиодной панелью. Для этого всего лишь нужно участвовать в аукционе на котором разыгрывается возможность изобразить свой собственный узор в определённое время. Победивший в аукционе узор передаётся со страницы администратора на сервер, который в свою очередь общается с сервером контроллера и запускает необходимую программу отображения.
2. Категория использования. Данное веб-приложение предназначено для любой категории людей, от детей до взрослых. Пользоваться приложением могут как частные лица, так и юридические компании в целях рекламы. Все средства собранные от участия в аукционе будут направлены в благотворительные фонды.
3. Аналоги. Существует много аналогов светодиодных экранов с возможностью размещения рекламы, однако в Беларуси нет сервиса, который позволяет взаимодействовать с контроллером напрямую через веб интерфейс.

Так же почти все экраны имеют рекламную направленность, “Lucidum” в свою очередь имеет развлекательный характер.

4. Минусы проекта. Минусами данного приложения является сторона монетизации. Т.к. проект рассматривается как развлекательный для города, громадных прибылей он приносить не может. Так же тот факт, что светодиодные экраны довольно развитое направление в городе, имеется много конкурентов, которые могут помешать развитию сервиса.

5. Плюсы проекта. Развиваясь предполагается развитие сервиса, будут внеднены не только светодиодные панели, но возможно и светодиодные кубы, которые представляют совершенно иной уровень 3D отображения. Расположение в городе такого развлечения привлечёт туристов и возможно станет отличительной чертой города.

1.4 Структура и функциональные возможности сервиса

Чтобы в полной мере оценить возможности ресурса нужно описать его структуру и основной функционал.

Структура сервиса будет состоять из двух составляющий: веб-приложение и контроллер. Приложение будет давать возможность удобно управлять контроллером с помощью веб-интерфейса. Часть сервера на стороне контроллера будет задавать определённые программы в зависимости от заданных параметров пользователя и передавать их на контроллер. Более детально взаимодействие описано в Главе 4.

Функциональные возможности сайта предполагают деление пользователей на две категории: обычные пользователи и администраторы. В дальнейшем возможно появится разделение на частных и юридических лиц.

Для обычных пользователей функционал позволяет:

- Задание узора;
- Демо уже заданных узоров;
- Участие в аукционе;
- Просмотр профайла;

Для администратора возможно:

- Создание аукциона;
- Отслеживание активности аукциона;
- Запуск программы на контроллере;
- Просмотр списка пользователей;

ГЛАВА 2
ТЕХНОЛОГИИ WEB-ПРИЛОЖЕНИЯ

Для создания веб-приложения важно использовать востребованные технологии на рынке разработки. Поэтому было выбрано написать приложение на MEAN стеке технологий в соответствии нынешним стандартам ES6, используя методологию single page application и шаблон проектирования MVC. Передача данных при этом будет проходить по http в соответствии с RestAPI. Опишем каждую технологию более подробно.

2.1 Реализация клиентской части

Для реализации клиентской части была использована одна составляющая MEAN стека технологий. Данная аббревиатура расшифровывается как MongoDB Expressjs Angular Nodejs. Клиентская часть написана на фреймворке Angular с использованием сборщика webpack для SPA на es6 с транспилятором Babel, т.к. многими браузерами старых версий не поддерживается стандарт es6.

2.1.1 Принцип устройства Single page application

Single Page Application – сокращенно SPA, что означает “приложение одной страницы”. Другими словами SPA – это web-приложение, размещенное на одной страницу, которая передаётся на сторону клиента и загружает весь необходимый код с загрузкой страницы.[5] Схематично это можно было бы представить так:

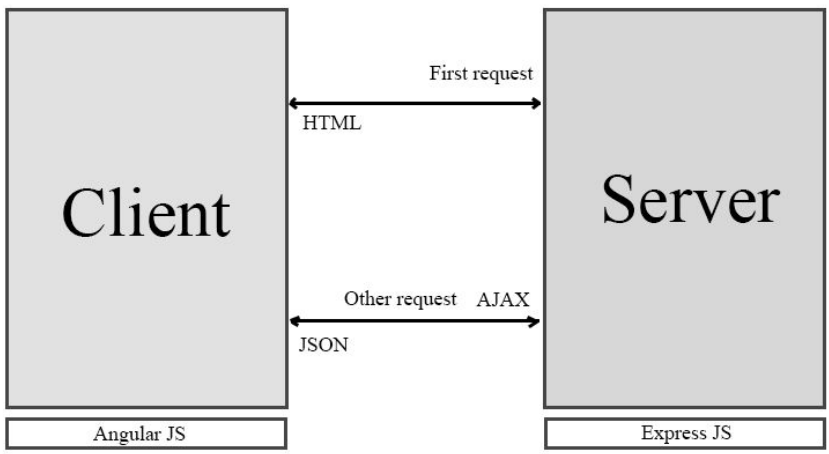


Рисунок 2.1 Схема SPA

Получается, что при первом запросе на сервер, мы получаем полноценную страницу вместе со всем кодом, который нам понадобится, а далее с помощью остальных запросов мы лишь передаём или просим интересующую нас информацию.

В таком способе построения приложения конечно же есть свои плюсы и минусы. Среди плюсов можно выделить тот факт, что SPA удобен для приложений, которые будут использовать на разных устройствах, компьютерах, мобильных устройствах. Ведь пользователю не придётся ждать при запросе получения всей страницы, он лишь будет получать информацию для вывода на странице. Так же плюс в том, что при обновлении информации на странице вы перезагружаете не всю страницу целиком, а лишь обновляете часть её, которая была изменена.

Среди минусов наверное можно выделить лишь тот факт, что клиентская сторона написана полностью на js. Но в наше время это почти совсем не проблема.

Но главное правило при выборе способа отображения и обработки данных нужно задуматься, подходит ли этот принцип разработки для проекта. Если проект большой и тяжёлый, информации много и она часто изменяется, то возможно принцип одностраничного приложения не самый лучший выход. Потому что приложение будет нагромождаться функционалом и появится путаница. Но если же приложение относительно небольшое и не загружено функциональность, то SPA это хороший выбор.

2.1.2 Шаблон проектирования MVVM

MVVM или model-view-viewmodel - шаблон проектирования, на котором строится один из возможных принципов построения проекта. Следуя данному паттерну, мы делим наше приложение на три составляющие части model, view, viewmodel. Каждая часть отвечает за свою логику.[6]

MVVM удобно использовать вместо классического MVC и ему подобных в тех случаях, когда в платформе, на которой ведётся разработка, присутствует «связывание данных».

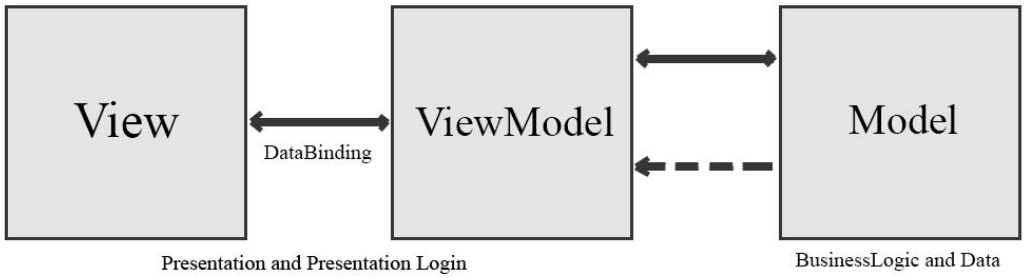


Рисунок 2.2 Схема MVVM

Шаблон MVVM делится на три части:

Модель, модель представляет собой фундаментальные данные, необходимые для работы приложения, это сущности системы, с которыми ведётся работа.

Представление — это графический интерфейс, то есть окно, кнопки и т.д.. Представление является подписчиком на событие изменения значений свойств или команд, предоставляемых Моделью представления. В случае, если в Модели представления изменилось какое-либо свойство, то она оповещает всех подписчиков об этом, и Представление, в свою очередь, запрашивает обновленное значение свойства из Модели представления. В случае, если пользователь воздействует на какой-либо элемент интерфейса, Представление вызывает соответствующую команду, предоставленную Моделью представления.

Модель представления является, с одной стороны, абстракцией Представления, а с другой, предоставляет обёртку данных из Модели, которые подлежат связыванию. То есть, она содержит Модель, которая преобразована к Представлению, а также содержит в себе команды, которыми может пользоваться Представление, чтобы влиять на Модель.

На принципах этого паттерна строится разработка с использованием AngularJS. Однако нельзя сказать однозначно, что MVVM полностью описывает структуру приложения на Angular. MVC(model-view-controller) частично так же укладывается в разработку на Angular. Controller из MVC, который в коде может быть модулем controller или service, отвечает за обработку данных, принятие, отправку запросов, что не укладывается в паттерн MVVM.

2.1.3 Сборщик модулей Webpack

Webpack – один из самых мощных и гибких инструментов для сборки frontend. Чтобы понять принцип его работы, нужно иметь примерное представление последовательности действий работы с ним.

Для начала с помощью команды npm install -g webpack, мы должны установить этот модуль на наше устройство, где ведётся разработка. В корне нашего проекта у нас есть конфигурационный файл webpack.conf.js, описывая его как объект с определёнными свойствами, мы задаёт полную работу сборщика. В этом файле для начала надо задать точки входа(entry points), т.е. файл с которого начинается сборка. Webpack загружает этот файл и начинает обрабатывать его, как только он встречает функцию require(), он оборачивается содержимое вызываемого файла в функцию, тем самым на выходе мы получим один сконфигурированный js файл, который хранит в себе все модули.

Многие пытаются сравнивать webpack и gulp. Однако это разные вещи. Если webpack это сборщик, который заточен на сборку js файлов в один файл, если какой-то из файлов обновляется, он не пересобирает все файлы, а лишь изменяет тот кусок, который был обновлён. **Gulp** - это система управления задачами. Сама по себе система ничего не умеет и лишь после того, как мы опишем для неё задачи и добавим их запуск, она отработает нужным нам образом. Конечно же одна из задач может быть сборка модулей.

Кроме сборки js файлов webpack может собирать и многие другие расширения .html, .jade, .jpeg, .svg, .ttf и т.д.. Для этот нужно подключить loaders в метод plugins, которые при встрече с require() в который мы передали пусть к файлу правильным образом обработать его.

В этом смысле Webpack очень удобен в SPA приложениях, потому что он собирает все модули в один файл, тем самым позволяет сразу отдать этот файл на сторону клиента.

2.1.4 Angular

Angular - javascript-фреймворк разработанный компанией Google. На данный момент последней стабильной версией является angular 1.5. К выходу уже довольно долгое время готовится Angular 2. Различия между двумя версиями колосальные, но основные понятия будут даны по Angular 1.

Angular построен на принципах MVVM, т.е. при изменениях в модели, изменяется и представление. Однако паттерн MVC так же подходит для описания разработки с использованием Angular.

MVC в AngularJS реализовано отдельным модулем и не играет важной роли в самой библиотеке.[8] Потребуется подключить angular-ui-router.js и в зависимости приложения добавить uiRoute:

```
var app = angular.module("tutorial", ["ui.route"])
```

uiRoute настраивается с помощью $stateProvider, например, так:

```
app.config(($stateProvider) => {
  $stateProvider.state('login', {
        url: '/login',
        controller: LoginCtrl,
        controllerAs: 'ctrl',
        templateUrl: './login.html'
      }})
```

Модели. Как таковых, моделей в AngularJS нет. Связывание данных происходит исключительно на области видимости.

Контроллеры. В отличие от моделей, концепция контроллеров в AngularJS есть, однако относятся контроллеры здесь непосредственно к DOM. В контроллер необходимо внедрить хотя бы $scope, в противном случае он будет вещью в себе и не сможет ничего сделать с областью видимости, а именно для этого предназначен контроллер. Контроллер всегда создаёт внутреннюю область видимости. У каждой области видимости есть свойства $parent (указывает на родительскую область видимости) и $root (указывает на глобальную область видимости).

uiView. Директива uiView позволяет динамически подгружать часть HTML в зависимости от значения templateUrl в настройках $stateProvider.

Выражения. Выражением в AngularJS считается практически любое выражение JavaScript. Это и математические операции, вызов функции и т.д. Выражение в AngularJS заключается в двойные фигурные скобки и может быть использовано непосредственно в DOM. AngularJS вычисляет выражение и подставляет его значение прямо в DOM. При этом используются все преимущества связывания данных: значение выражения обновляется с изменением входящих в него переменных (или результатов вычисления функции).

Шаблоны. Шаблоны в AngularJS это не более, чем обычный HTML, расширенный с помощью директив и выражений. Выражения используются для вывода данных, директивы — для расширения функциональности HTML для вашего конкретного приложения. Можно добавить директиву, создающую интерфейс с вкладками, можно добавить директиву, которая выводит в цикле

содержимое массива с данными с помощью ng-repeat, можно выводить HTML в зависимости от значения той или иной переменной и так далее. Подлинная мощь AngularJS заключена именно в директивах, а обычный , расширенный директивой может превратиться во что угодно.

Сервисы. Сервисы позволяют хранить данные Модели и бизнес-логику, например, общение с сервером через HTTP. сервис использует конструктор, поэтому, когда используете его в первый раз, он выполнит new Login(); для создания экземпляра объекта.

```
app.service('loginService', function($http) {
    return {
        login: () => {
            return $http.post('/login', {
                username: 'Lena',
                password: '1234'
            }}
        }
});
```

2.1.5 Стандарт языка JavaScript ECMAScript6

ECMAScript — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков Стандартизирован международной организацией ECMA в спецификации ECMA-262. Расширения языка, JavaScript, JScript и ActionScript, широко используются в вебе.[9]

В отличии от предыдущих версий ecmascript, es6 стал более лаконичный и легко поддерживаемый. Появилось много нововведений, которые облегчают и более структуризируют написание кода.

На данный момент некоторые браузеры в полной мере не поддерживают es6, поэтому к выше описанному webpack был подключён модуль babel.[10] Babel.JS – это транспайлер, переписывающий код на ES-2015 в код на предыдущем стандарте ES5.

Кратко опишем основные нововведения ES6.[11]

- **let вместо var.** Вместо того, чтобы писать var x = y теперь можно написать let x = y. **let** позволяет объявлять переменные с блочной областью видимости. Это особенно полезно для for или while циклов.
- **Строковые шаблоны вместо конкатенации.** В стандартной библиотеки JavaScript, составление строк всегда было более болезненным, чем следовало бы. Строковые шаблоны сделали встраивание выражений в строки

тривиальной операцией, также как и поддержку нескольких линий. Просто нужно заменить ‘ на `:

```
let str = `Hello, ${name}`;
```

- **Классы вместо прототипов.** Определение класса было громоздкой операцией и требовало глубокого знания внутреннего устройства языка. Даже несмотря на то, что, польза понимания внутреннего устройства очевидна, порог входа для новичков был неоправданно высоким. Class предлагает синтаксический сахар для определения функции конструктора, методов прототипа и геттеров / сеттеров. Он также реализует прототипное наследование со встроенным синтаксисом (без дополнительных библиотек или модулей).

```
class A extends B {
        constructor(){}
        method(){}
        get prop(){}
        set prop(){}
}
```

- **() => вместо function.** Не только потому что **(x, y) => {}** короче написать, чем **function (x,y) {}**, но поведение **this** в теле функции, скорее всего, будет ссылаться на то, что вы хотите.
- **Экспорт.** Ключевое слово export, стоящее перед объявлением переменной (посредством var, let, const), функции или класса экспортирует их значение в остальные части программы. Например:

```
export function square(x) {
   return x * x;
}
```

- **Импорт.** Импортирует значение из части программы прописанной в пути. Например:

```
import { square } from './lib';
```

2.2 Технологии взаимодействия клиентской и серверной сторон

2.2.1 Архитектура RestAPI

Одной из немало важной части проекта является правильно спроектированный REST API. Но прежде чем начать проектирование, нужно разобраться что же это такое.

REST API определяет набор функций, к которым разработчики могут совершать запросы и получать ответы. Взаимодействие происходит по протоколу HTTP. Преимуществом такого подхода является широкое распространение протокола HTTP, поэтому REST API можно использовать практически из любого языка программирования. [20]

Архитектура в стиле REST состоит из клиентов и серверов. Клиенты инициируют запросы к серверам; серверы обрабатывают запросы и возвращают подходящие ответы.

Основы REST и HTTP-методы. Существуют основные правила оформления URL, которых следует придерживаться для написания архитектурно правильного REST API.

Желательно все URL адреса API начинать с префикса, например, /api/ это поможет упростить сопровождение API в будущем. Хорошим вариантом может оказаться префикс в имени домена http://api.example.com/users/ae25b8

Существует 2 основных типа ресурса в архитектуре REST: коллекция и элемент коллекции.

Коллекция представляет собой набор самостоятельных и самодостаточных элементов. Например, /api/users. Элемент коллекции пользователей, или конкретный пользователь, в таком случае, может быть представлен в виде: /api/users/ae25b8.

Имена коллекций должны представлять сущности (существительные во множественном числе), и они должны быть максимально конкретными и понятными.

Каждым ресурсом в REST API управляет несколько определенных минимально необходимых глаголов.

Ресурс	*POST*	*GET*	*PUT*	*DELETE*
/users	*Создать пользователя*	*Показать список всех пользователей*	*Обновить список всех пользователей*	*Удалить всех пользователей*
/users/ae25b8	*Ошибка*	*Показать пользователя*	*Если есть, обновить*	*Удалить пользователя*

			пользователя, если нет Ошибка	

Таблица 1 Методы Rest

Ошибки. Следует различать 2 основных семейства статус кодов (HTTP Status Code):

4xx — проблема возникла на стороне пользователя и он сам может ее исправить, правильно введя необходимую для запроса информацию.

5xx — проблема возникла на сервере и для ее решения пользователь может отправить запрос к службе поддержки.

Ошибки должны быть четко описаны, чтобы не только пользователь знал, что ему необходимо сделать, но и сам разработчик легко ориентировался, если пользователь присылает запрос для решения проблемы.

Существует необходимый минимум статус кодов в приложении, которые стоит знать любому разработчику:

1. 200 OK
2. 400 Bad Request (некорректный запрос)
3. 500 Internal Server Error (внутренняя ошибка сервера)
4. 201 Created (успешно создан)
5. 304 Not Modified (данные не изменились)
6. 404 Not Found (не найден)
7. 401 Unauthorized (не авторизован)
8. 403 Forbidden (доступ запрещен)

2.2.2 Библиотека Socket.io

Socket.IO — JavaScript библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для node.js. Оба компонента имеют похожее API. Подобно node.js, Socket.IO событийно-ориентированная.[12]

Socket.IO — JavaScript библиотека для веб-приложений и обмена данными в реальном времени. Состоит из двух частей: клиентской, которая запускается в браузере и серверной для node.js. Оба компонента имеют похожее API.

Socket.IO главным образом использует протокол WebSocket, но если нужно, использует другие методы, например Adobe Flash сокеты, JSONP запросы или AJAX запросы, предоставляя тот же самый интерфейс. Помимо того, что Socket.IO может быть использована, как оболочка для WebSocket, она содержит много других функций, включая вещание на несколько сокетов,

хранение данных, связанных с каждым клиентом, и асинхронный ввод/вывод. Подключается как модуль на стороне сервера require(‘socket.io’) и на стороне клиента.

Связь между клиентом и сервером происходит с помощью событий .on, .emit, .broadcast. Событие .on является отправной точкой, которая передаёт данные событию .emit.

2.3 Реализация серверной части

2.3.1 Фреймворк Express JS

На текущий момент Express - наиболее подходящий фреймворк для большинства Node.js-разработчиков. Он относительно зрелый и построен на базе connect. Поддерживает такие возможности, как маршрутизация, конфигурация, шаблонный движок, разбор POST запросов и многое другое. Автором express является TJ Holowaychuk. TJ является автором 85 модулей для node.js.[13]

Небольшой пример написания кода:

```
var express = require('express');
var app = express.createServer();
app.get('/', function(req, res){
  res.send('Hello World');
});
app.listen(3000);
```

В этом примере создается web-сервер, «слушающий» порт 3000 и обрабатывающий запрос к /, ответом на который выводится строка Hello World.

В express встроена мощная система маршрутизации. Например:

```
app.get('/user/:id', function(req, res){
  res.send('user ' + req.params.id);
});
```

Данный код обрабатывает запросы /user/:id, для которых автоматически выставляется значение :id для переменной req.params.id. Для описания маршрутов так же можно использовать регулярные выражения.

Для того, чтобы обрабатывать POST запросы, приложению необходимо использовать специальный middleware — bodyParser. Подключается он очень легко: app.use(express.bodyParser()) (перед эти инсталируется в папку проекта при помощи пакетного менеджера npm). BodyParser обрабатывает тела application/x-www-form-urlencoded и application/json запросов и выставляет для них req.body. Вот пример:

```
app.use(express.bodyParser());
app.post('/', function(req, res){
    console.log(req.body.name);
    res.send('ok');
});
```

В данном примере на консоль выводится значение переменной name из тела запроса, а в ответ на запрос возвращается строка ok.

Кроме bodyParser доступно еще несколько middleware'ей:

- logger отвечает за логирование HTTP запросов,
- cookieParser — за обработку cookies,
- session — за работу с сессиями,
- static — за работу со статическим контентом (css, javascript, картинки),
- errorHandler — за обработку ошибок.

В конце приведу основные возможности express:

1. Гибкая система маршрутизации запросов;
2. Перенаправления(app.use(express.static()));
3. Динамические представления;
4. Обработка представлений и поддержка частичных шаблонов;
5. Поддержка конфигураций на основе окружений;
6. Максимальное покрытие тестами;

2.3.2 MongoDB

В своём проекте я использую документо-ориентированную базу данных MongoDB, не требующая описания схемы таблиц.

Документо-ориентированная СУБД - это база данных специально предназначенная для хранения иерархических структур данных (документов) и обычно реализуемая с помощью подхода NoSQL. В основе документо-ориентированных СУБД лежат документные хранилища, имеющие структуру дерева (иногда леса). API для поиска позволяет находить по запросу документы и части документов. В отличие от хранилищ типа ключ-значение, выборка по запросу к документному хранилищу может содержать части большого количества документов без полной загрузки этих документов в оперативную память.[14]

Формат данных в MongoDB. Одним из популярных стандартов обмена данными и их хранения является JSON (JavaScript Object Notation). JSON эффективно описывает сложные по структуре данные. Способ хранения данных в MongoDB в этом плане похож на JSON, хотя формально JSON не

используется. Для хранения в MongoDB применяется формат, который называется BSON (БиСон) или сокращение от binary JSON. BSON позволяет работать с данными быстрее: быстрее выполняется поиск и обработка. Хотя надо отметить, что BSON в отличие от хранения данных в формате JSON имеет небольшой недостаток: в целом данные в JSON-формате занимают меньше места, чем в формате BSON, с другой стороны, данный недостаток с лихвой окупается скоростью. [15]

Кроссплатформенность. MongoDB написана на C++, поэтому ее легко импортировать на самые разные платформы. MongoDB может быть развернута на платформах Windows, Linux, MacOS.

Специфика MongoDB. Если реляционные базы данных хранят строки, то MongoDB хранит документы. В отличие от строк документы могут хранить сложную по структуре информацию. Документ можно представить как хранилище ключей и значений. Ключ представляет простую метку, с которым ассоциировано определенный кусок данных.

Коллекции. Если в традиционном SQL есть таблицы, то в MongoDB есть коллекции. И если в реляционных БД таблицы хранят однотипные жестко структурированные объекты, то в коллекции могут содержать самые разные объекты, имеющие различную структуру и различный набор свойств.

Использование MongoDB. В своём проекте я не устанавливаю локально базу для хранения данных, а использую сторонние сервисы, одним из которых является MongoLab. Выбор пал именно на этот сервис, т.к. он позволяет на определённый объём данных использовать их хранилища бесплатно. Всё что нужно для создания своей базы, это зарегистрироваться и войти в систему, заполнив основные поля о себе и базе, которая будет создана.

После создания, сервисом будет дан специальный Url, который нужно прописать в проекте:

```
var url = 'mongodb://' + config.get('mongo:user') + ':' + config.get('mongo:password') + '@'
+ config.get('mongo:url') + ':' + config.get('mongo:port') + '/' + config.get('mongo:db');
```

и подключить используя модуль mongoose:

```
var mongoose = require('mongoose');
var autoIncrement = require('mongoose-auto-increment');
var db = mongoose.createConnection(url);
autoIncrement.initialize(db);
db.once('open', function callback() {
        console.info('Mongo db connected successfully');
```

```
});
```

Драйвер MongoDB. Конечно, мы можем работать и через консоль mongo, добавляя и отображая объекты в базу данных. Но было бы неплохо, если бы MongoDB взаимодействовала бы с нашим приложением, написанными на NodeJS. Как раз для этой цели нам потребуются специальный драйвер. Мы будем использовать драйвер mongoose.

Mongoose. Mongoose – это ODM для Node.js. Он имеет процветающий открытый исходный код и включает в себя передовые схемы на основе функций, таких как проверка асинхронности, управления жизненным циклом объекта, псевдо-соединения и широкая поддержка работы с запросами.
Установить его очень легко используя менеджер пакетов *npm*:

```
$ npm install mongoose
```

После установки нам нужно подключить нашу базу данных к проекту, это делается через прописывание в js файле:

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/my_database');
```

Теперь мы можем создавать схемы для нашей базы данных, например, создадим схему User:

```
var UserSchema = mongoose.Schema({
    login: String,
    password: String,
    age: Number
});
```

Теперь мы можем обратиться к схеме в нашем проекте:

```
var User = mongoose.model('User', UserSchema);
```

Далее можно сформировывать запросы и производить работу с базой данных. Примеры использования mongoose можно увидеть в (Приложении В).

ГЛАВА 3
ТЕХНОЛОГИИ КОНТРОЛЛЕРА ARDUINO

3.1 Понятие контроллера

Контроллер — это миниатюрный компьютер с набором входов и выходов, работающий по заранее написанной программе.[16]

Контроллер — вещь сама по себе универсальная. Ко входам можно подключить как обычные кнопки (пульт), так и температурные датчики (кондиционер), модули беспроводной связи (телефон) и даже электрогитару (цифровой процессор эффектов). Выходы также могут управлять чем угодно. Задача контроллера — измерять электрическое напряжение на входах и подавать напряжение на выходы в соответствии с программой.

На любом контроллере, со всеми выходами и входами находится “ядро” всей этой системы, микроконтроллер.

Микроконтроллер – это специальная микросхема, предназначенная для управления различными электронными устройствами. Он объединяет процессор, память, ПЗУ и периферию внутри одного корпуса, внешне похожего на обычную микросхему.

Самый главный элемент любого процессора – арифметико-логический узел (АЛУ). В нем-то происходят арифметические и логические операции над числами.

На схеме так же присутствует ОЗУ – оперативная память контроллера. Предназначена она для для того, чтобы хранить данные при выполнении программы.

Для того, чтобы контроллер обменивался данными с внешними устройствами существуют **порты ввода/вывода (ПВВ)**. **Порт** – это пачка однобитных каналов, каждый из которых может быть независимо настроен либо на ввод, либо на вывод. Любая программа для контроллера начинается

именно с настройки портов. Надо определить, какие каналы будут работать на ввод, какие – на вывод. По умолчанию, все каналы включены на ввод.[17]

3.2 Типы контроллеров Arduino

Arduino — это небольшая плата с собственным процессором и памятью. На плате также есть пара десятков контактов, к которым можно подключать всевозможные компоненты: лампочки, датчики, моторы, чайники, роутеры, магнитные дверные замки и вообще всё, что работает от электричества.[21]

В процессор Arduino можно загрузить программу, которая будет управлять всеми этими устройствами по заданному алгоритму. Таким образом можно создать бесконечное количество уникальных классных гэджетов, сделанных своими руками и по собственной задумке.

Существует несколько версий платформ Arduino. Последняя версия Leonardo базируется на микроконтроллере ATmega32u4. Uno, как и предыдущая версия Duemilanove построены на микроконтроллере Atmel ATmega328. Старые версии платформы Diecimila и первая рабочая Duemilanoves были разработаны на основе Atmel ATmega168, более ранние версии использовали ATmega8. Arduino Mega2560, в свою очередь, построена на микроконтроллере ATmega2560.[18]

Ниже представлены основные версии плат Arduino:

Leonardo — последняя версия платформы Arduno на ATmega32u4 микроконтроллере . Отличается разъемом microUSB, по размерам совпадает с UNO.

Uno — самая популяраня версия базовой платформы Arduino USB. Uno имеет стандартный порт USB. Arduino Uno во многом схожа с Duemilanove, но имеет новый чип ATMega8U2 для последовательного подключения по USB и новую, более удобную маркировку вход/выходов. Платформа может быть дополнена платами расширения, например, пользовательскими платами с различными функциями.

Arduino Ethernet — контроллер со встроенной поддержкой работы по сети и с опциональной возможностью питания по сети с помощью модуля POE (Power over Ethernet).

Nano — это компактная платформа, используемая как макет. Nano подключается к компьютеру при помощи кабеля USB Mini-B.

Mega ADK - версия платы Mega 2560 с поддрежкой USB host интерфейса для связи с телефонами на Android и другими устройствами с USB интерфейсом.

Большую популярность плата Arduino приобрела не только из-за низкой стоимости, легкости разработки и программирования, но, главным образом, благодаря наличию плат расширения (так называемых шилдов), добавляющих Arduino дополнительную функциональность. Шилды (кроме маленьких модулей и платы LilyPad) подключаются к Arduino с помощью имеющихся на них штыревых разъемов.

И многие другие типы контроллеров Arduino. В моём же распоряжении есть Arduino Uno, работу с которым я и буду вести. Но для начала краткое описание.

Arduino Uno — это базовый контроллер семейства Arduino, идеально подходящий для решения большинства задач. У него есть 14 контактов, которые могут служить и входами, и выходами, serial-интерфейс для подключения к компьютеру, USB-порт. Для более специфических задач могут подойти другие платы.[16]

3.3 Плата расширения Arduino WiFi Shield

Плата расширения Arduino WiFi позволяет контроллерам Arduino осуществлять сетевое соединение, используя беспроводную сеть. Плата построена на базе чипа HDG104 Wireless LAN 802.11b/g System in-Package.

Физически плата WiFi, как и большинство плат расширения, соединяется с платой контроллера Arduino посредством контактных колодок, расположенных по краям платы.

На плате WiFi имеется слот для micro-SD карт, которые могут быть использованы для хранения и передачи файлов по сети.

Соединение Arduino контроллера с процессором платы расширения WiFi и встроенной картой SD осуществляется по SPI шине на разъеме ICSP.

SPI (Serial Peripheral Interface), или последовательный периферийный интерфейс, был разработан компанией Motorola для организации быстрого и простого в реализации обмена данными между компонентами системы — микроконтроллерами и периферийными устройствами. На шине может быть одно ведущее устройство (master) и несколько ведомых (slave). При этом задействованы следущие выходы: 11, 12 и 13 на UNO. На обоих платах выход 10 используется для выбора HDG104 и выход 4 для SD карты. Эти вход/выходы

не могут быть задействованы для других целей. Порт 3 используется для синхронизации (handshake) между платой WiFi и Arduino и не может быть задействован для других целей.

Для использования данной платы, мной была установлена библиотека Adafruit_CC3000. Которую я положила в папку libraries, папки arduino. В будущем её можно будет подключить. При помощи класса Adafruit_CC3000 реализуется интерфейс для работы wifi.

3.4 Реализация клиента на контроллере Arduino

Для написания клиента для контроллера Arduino потребуется ознакомится с рядом инструментов, которые нужны для разработки под Arduino.

3.4.1 Среда разработки Arduino IDE

ArduinoIDE - бесплатная среда разработки в которой происходит написание,компиляция и загрузка программ в микроконтроллер. Данная среда разработки минимально наполнена функционалом, однако этого достаточно, для программирования на Arduino. Интерфейс очень понятен и просто.

Sketch (скетч) — это программа, написанная в среде Arduino. Расширение данного файла .ino.

Разрабатываемым скетчам дополнительная функциональность может быть добавлена с помощью библиотек, оторые можно подключить к создаваемому проекту. Специализированных библиотек существует множество. Обычно библиотеки пишутся так, чтобы упростить решение той или иной задачи и скрыть от разработчика детали программно-аппаратной реализации.

Bootloader (загрузчик) — специальная программа в микроконтроллере позволяющая с минимальными сложностями загружать скетчи — просто через USB-кабель или как в моём случае через wifi.[19]

Так же для начала работы с IDE её нужно настроить.

Выберем тип Arduino в меню “Tools ->Boards->Arduino UNO”. Далее нужно проверить, видит ли компьютер Arduino, т.е. виден ли порт, через который мы его подключили (“Tools->Serial Port”)

Для запрограммирования микроконтроллера, есть тип файла .ino. Файлы компилируются, к библиотекам шлются обращения. Всё это написано на языке C++, который оперирует функциями, переменными, после чего, чтобы

написанная на языке C++ программа стала понятна микроконтроллеру, она переводится в машинный код — компилируется и затем загружается в контроллер.

3.4.2 Scketch для контроллера Arduino

Данный раздел уделён описанию программы написанной для работы контроллера. Программа описывает работу светодиодной панели, по заданным входным данным происходит загорание и затухание светодиодов. Программа имеет 4 встроенные динамические программы для отработки и может получать на вход данные для статичного загорания определённых диодов.

Опишем вкратце методы и их значение использованные в программе.

Порты платформы Arduino могу работать как входы или выходы. Т.к. для моей программы порты работают только для вывода, с помощью метода pinMode() передадим номер порта и значение OUTPUT. Метод ирменяющий режим работы порта описан в приложении В

Arduino программируется на языке Wiring, которого на самом деле не существует, как не существует и компилятора Wiring — написанные на Wiring программы преобразуются в программу на языке C/C++ и затем компилируются компилятором AVR-GCC. Фактически используется специализированный для микроконтроллеров AVR вариант C/C++.

Базовая структура программы для Arduino состоит, по меньшей мере, из двух обязательных частей: функций **setup()** и **loop()**. Перед функцией setup() идет объявление переменных, подключение библиотек. Функция setup() запускается один раз после каждого включения питания или сброса платы Arduino. Она используется для инициализации переменных, установки режима работы портов и прочих подготовительных для основного цикла программы действий. Она обязательно должна быть включена в программу, даже если не выполняет никаких действий.

Функция loop() в бесконечном цикле последовательно исполняет команды, которые описаны в ее теле. Эта функция выполняется циклически, она выполняет основную работу.[22]

Так же важным методом в написании программы является digitalWrite(), на вход которому подаётся значение HIGH или LOW. Т.к. режим работы портов имеет значение OUTPUT, HIGH или LOW будут означать, что напряжение на порту 5В и 0В соответственно, т.е. диод загорается или затухает.

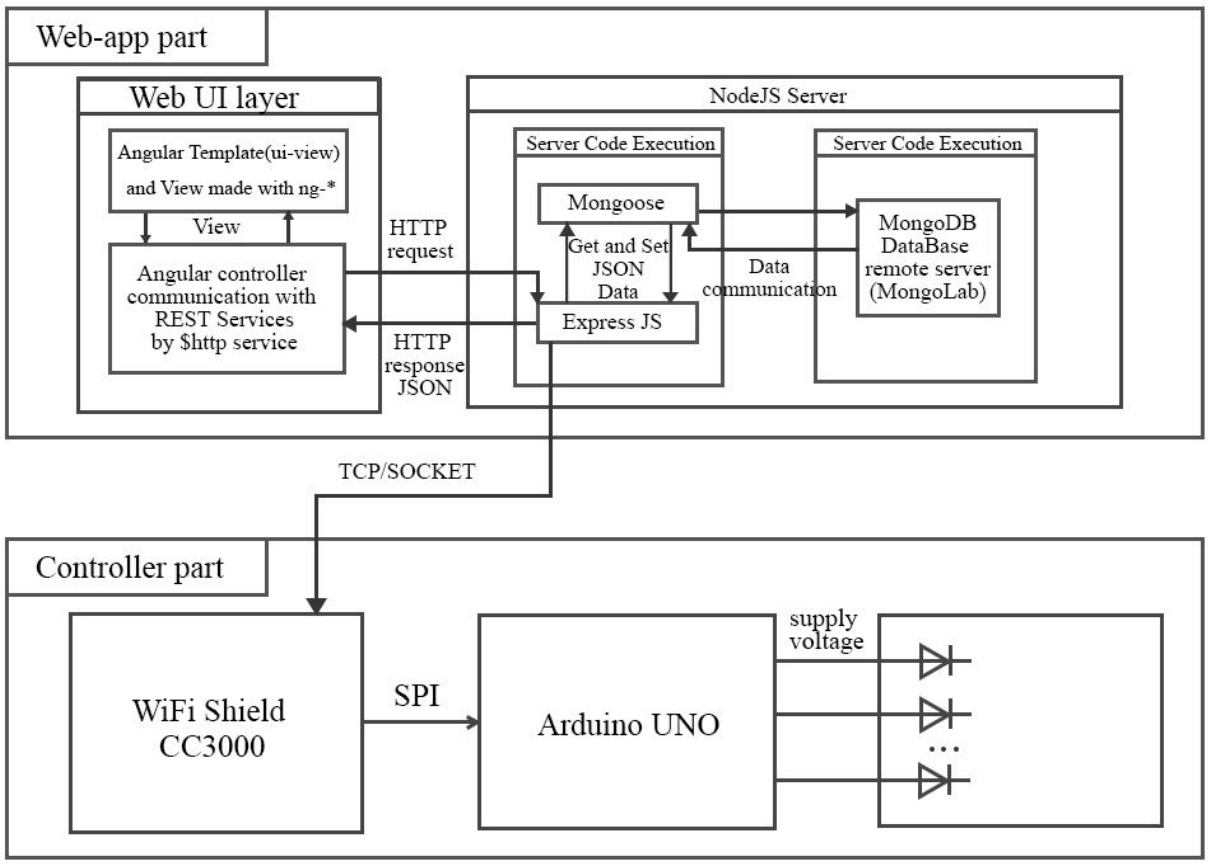
В программе имеется подключенная библиотека Adafruit_CC3000, которая позволяет работать с wifi shield. Инициализация происходит засчёт

метода Adafruit_CC3000(), в который будут передаваться порты с помощью которых происходит подключение. Инициализация wifi shield происходит в функции initWiFiShield(), которая представлена в Приложении В.

ГЛАВА 4
ВЗАИМОДЕЙСТВИЕ ВЕБ-ПРИЛОЖЕНИЯ И КОНТРОЛЛЕРА

4.1 Схема и принцип взаимодействия

Взаимодействие между частями приложения происходит по следующему принципу.



Есть две составляющие части сервиса. Сторона Web-приложения и работа с контроллером Arduino. Сторона приложения состоит из клиентской части, написанной на Angular и серверной части, написанной на Express JS.

По средствам Angular мы делаем первый запрос на сервер, в ответ мы получаем html страницу с подгруженными к ней js файлами. При помощи роутера uiRouter мы имеем возможность перемещаться по state и изменять контент страниц не перезагружая их. Последующие запросы на сервер обрабатываются с помощью маршрутизаторов и отдают на сторону клиента JSON файлы с интересующими нас данными. Если же мы делаем POST запрос в соответствии rest, данные могут или при помощи модуля mongoose передаться на удалённый сервер базы данных MongoDB и сохраниться в ней, или же запрос был направлен на работу с контроллером.

При запуске сервера на Express запускается net server, перед этим подключив модуль net: require(‘net’) на порту 6969, который работает с сокетами и ждём соединения со стороны Arduino. Когда соединение было установлено, при помощи TCP протокола используя сокеты мы передаём данные на сторону контроллера.

На стороне контроллера на этот момент запущена программа, которая используя wifi библиотеку позволяет нам установить соединением с net сервером на стороне web-приложения. Получив данные по tcp, wifi shield передаёт их по SPI на Arduino, где они обрабатываются и на основе их подаётся напряжение на соответствующие порты светодиодной схемы, тем самым включая или выключая диоды.

ЗАКЛЮЧЕНИЕ

Ни для кого не секрет, что мы живём во время быстро развивающихся технологий. Люди пытаются создать технику и технологии, которые упрощали бы им жизнь во всех её аспектах. Особенную актуальность на данный момент имеют веб-технологии. JavaScript за последние пару лет стал из почти неприметного языка в один из самых востребованных на рынке разработчиков. На нём можно написать всё, начиная от обычного веб-приложения, до полноценной игры. В таком же направлении двигаются и контроллеры Arduino, которые мало того, что дешёвые, на них так же можно написать довольно полезные вещи, такие например как систему для умного дома.

Поэтому мне захотелось объединить эти два направления в полноценное приложение, которое использовалось бы в развлекательных целях.

По окончанию написания курсовой работы и создания приложения взаимодействующего с контроллером, я пришла к заключению, что написание веб-приложений используя стек технологий MEAN это удобный и красивый способ написать веб-приложение, которое удовлетворяла бы всем возможностям нынешней разработки. Так же это удобный способ для связи веб части и контроллера. Достаточно всего лишь подключить модуль, чтобы передавать данные на контроллер.

Так же за время написания курсовой, я познакомилась с совершенно новым для себя направлением - контроллеры. Оказалось, что используя лишь самые основные знания в C++ и небольшие познания в физике, дают широкий спектр возможностей для разработки программ контроллера.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. maesens.by URL: https://maesens.by/
2. МаеСэнс // Wikipedia URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%B5%D0%A1%D1%8D%D0%BD%D1%81
3. Умный дом // Wikipedia URL: https://ru.wikipedia.org/wiki/%D0%A3%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B4%D0%BE%D0%BC
4. Как провести бизнес-анализ // barmashovks.ru URL: http://www.barmashovks.ru/page206/page212/index.html
5. Что такое SPA или одностраничный портал // Codenet URL: http://www.codenet.ru/webmast/js/spa/
6. Model-View-ViewModel // Wikipedia URL: https://ru.wikipedia.org/wiki/Model-View-ViewModel
7. Webpack URL: https://webpack.github.io/
8. Angular.JS: введение и основные концепции // Makeomatic URL: https://makeomatic.ru/blog/2013/08/14/AngularJSIntro/
9. ECMAScript // Wikipedia URL: https://ru.wikipedia.org/wiki/ECMAScript
10. ES-2015 сейчас // learn.javascript URL: https://learn.javascript.ru/es-modern-usage
11. Модули в ECMAScript 6: будущее уже сейчас // frontender URL: http://frontender.info/es6-modules/
12. Socket.IO // Wikipedia URL: https://ru.wikipedia.org/wiki/Socket.IO
13. Руководство для начинающих по Node.js от Felix’a // nodeguide.ru URL: http://nodeguide.ru/doc/felix/beginner/#express

14. Документо-ориентированная СУБД // Википедия URL: https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94
15. 1. Введение в MongoDB // metanit.com URL: http://metanit.com/nosql/mongodb/1.1.php
16. Конструктор Arduino: что это и как работает // Popmech URL: http://www.popmech.ru/diy/12982-konstruktor-arduino-chto-eto-i-kak-rabotaet/#full
17. Что такое контроллер, и с чем его едят? // Radiokot URL: http://radiokot.ru/start/mcu_fpga/avr/01/
18. Аппаратная часть платформы Arduino // Arduino URL: http://arduino.ru/Hardware
19. Arduino, термины, начало работы // Robocraft URL: http://robocraft.ru/blog/arduino/1050.html
20. REST. Введение // blogger.sapronov.me URL: http://blogger.sapronov.me/2014/02/rest.html
21. Что такое Arduino // Amperka URL: http://amperka.ru/page/what-is-arduino
22. Викто Петин Проекты с использование контроллера Arduino. СПб.: БХВ-Петербург, 2014.

ПРИЛОЖЕНИЕ А

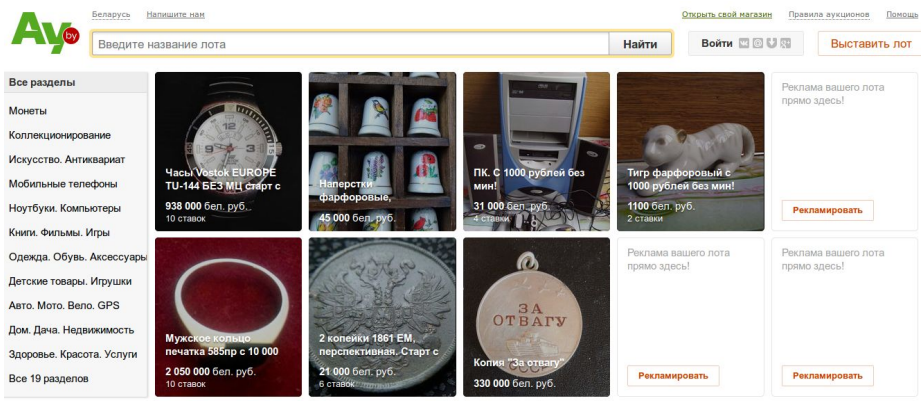


Рисунок 1.1 Сайт аукционов

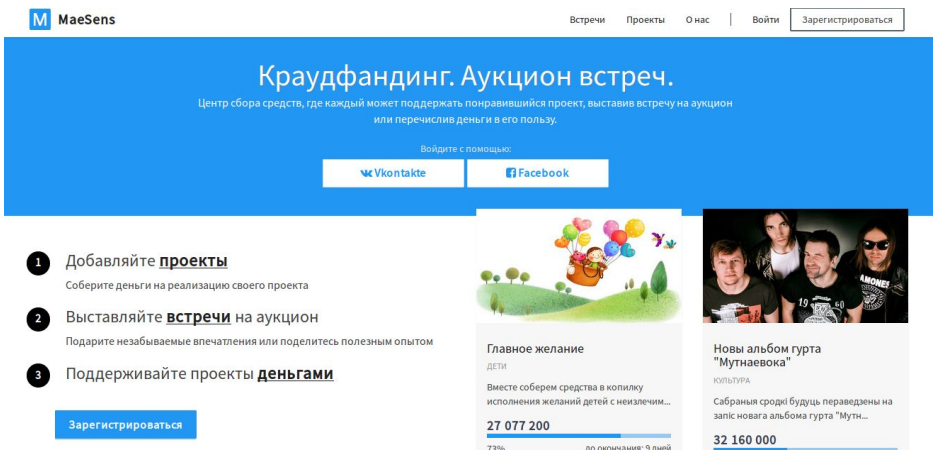


Рисунок 1.2 Сайт maesens.by

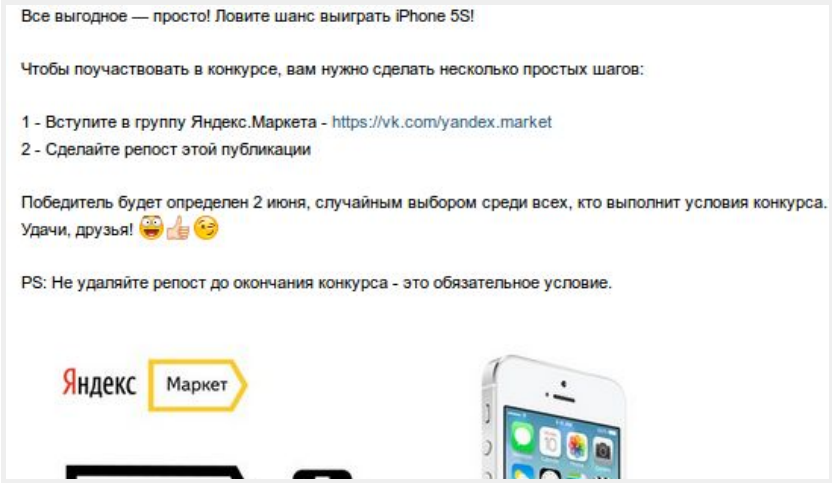


Рисунок 1.3 Пример аукциона в соц.сетях

Рисунок 3.1 Главная страница приложения

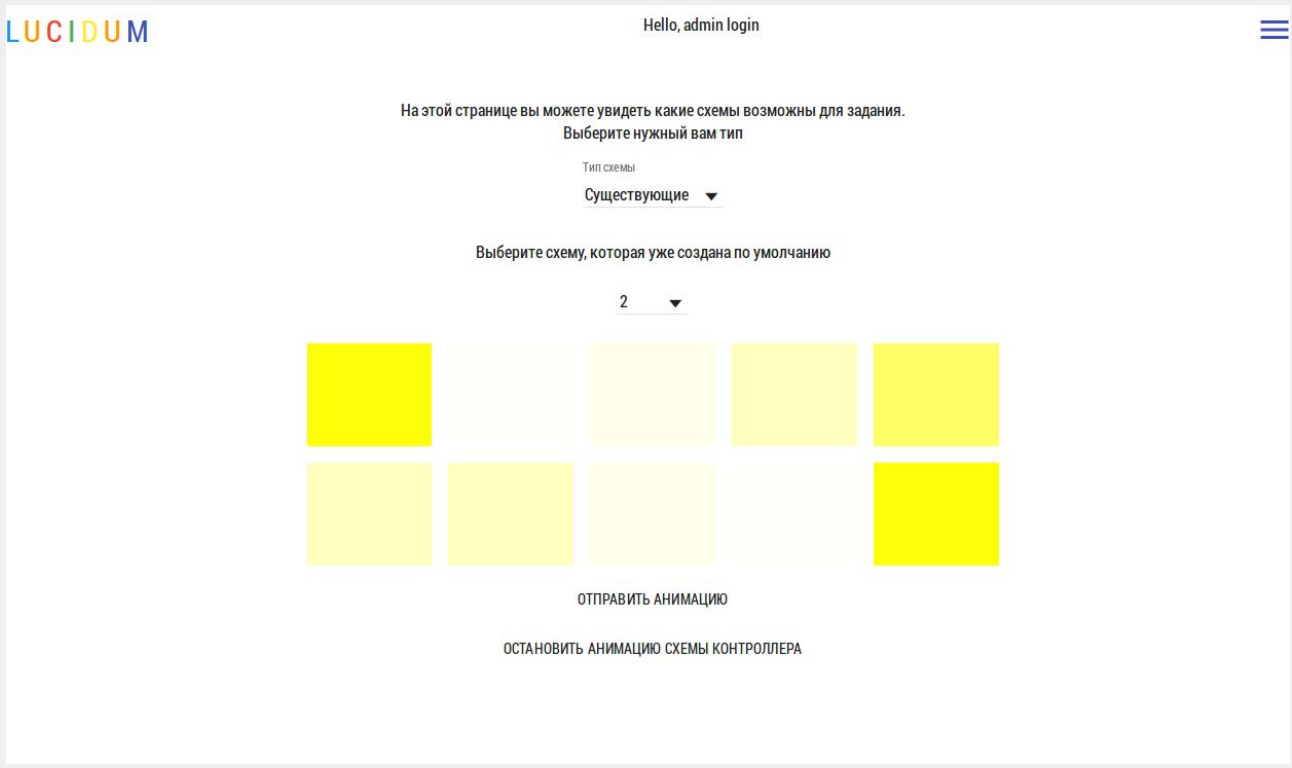


Рисунок 3.2 Страница задания схемы диодов

ПРИЛОЖЕНИЕ В

Ниже приведен код конфигурационного файла webpack

```
module.exports = function (_path) {
  return {
    entry: {
      vendor: [
        "./node_modules/angular/angular.js",
        "./node_modules/angular-ui-router/release/angular-ui-router.min.js"
      ],
      app: './public/src/index'
    },
    externals: {
      'angular': 'angular'
    },
    resolve: {
      modulesDirectories: ['node_modules']
    },
    output: {
      path: _path + '/public/dist',
      filename: 'app.js'
```

```
        },
        module: {
          loaders: [{
            test: /\.js$/,
            exclude: /node_modules/,
            loader: 'babel-loader',
            query: {
              presets: ['es2015'],
              cacheDirectory: true
            }
          }, {
            test: /\.html$/,
            loader: 'html'
          }]
        },
        plugins: [
          new webpack.optimize.CommonsChunkPlugin('vendor', 'vendor.js'),
          new ExtractTextPlugin('index.css'),
          new HtmlPlugin({
            title: 'Test APP',
            chunks: ['app', 'vendor'],
            filename: 'index.html',
            template: path.join(_path, 'public', 'src', 'layout.jade')
          })
        ]
      };
    }
```

Ниже приведён код основного js файла на стороне клиента

```
import angular from 'angular';

import landing from './landing/landing';
import AuthService from './common/service/auth-service';
import SessionStorageService from './common/service/session-storage-service';
import CookiesService from './common/service/cookies-service';

export default angular.module('app', [
    'ui.router',
    'ngCookies',
    'ngStorage',
    'ngMaterial',
    'ngMdIcons',
    landing.name
  ])
```

```
.service('authService', AuthService)
.service('sessionStorageService', SessionStorageService)
.service('cookiesService', CookiesService)
.config(($locationProvider, $httpProvider) => {
  var interceptor = ($q, $injector) => {
    return {
      'response': response => {
        return response.data;
      },
      'responseError': rejection => {
        if (rejection.status == '401') {
          $injector.get('$state').transitionTo('landing');
        }
        return $q.reject(rejection.data);
      }
    };
  };
  $httpProvider.interceptors.push(interceptor);
  $locationProvider.html5Mode(true);
})
.run(function ($rootScope, $state, authService,cookiesService) {
  $rootScope.$on('$stateChangeStart', function (event, stateInfo, current) {
      if (stateInfo.name != 'landing' && stateInfo.name != 'login' && stateInfo.name !=
'register') {
      if(stateInfo.name.search('admin') == 0 && cookiesService.get('type') != 'admin'){
        event.preventDefault();
        $state.go('landing');
      } else {
        authService.isAuth()
          .then(() => {

          })
          .catch(() => {
            event.preventDefault();
            $state.go('landing');
          });
      }
    }
  });
});
```

Ниже приведён код основного js файла на стороне сервера

```
var express = require('express');
var http = require('http');
```

```
var path = require('path');
var passport = require('passport');
var session = require('express-session');
var bodyParser = require('body-parser');
var config = require('./config');
var app = express();
app.use(bodyParser.text());
app.use(cookieParser());
app.use(session({
  secret: 'cuepath',
  resave: false,
  saveUninitialized: true
}));
app.use(passport.initialize());
app.use(passport.session());
var index = require('./routes/index.js');
var api = require('./routes/api.js');
app.set('superSecret', config.get('secret'));
// for first load /admin
app.use('/',express.static(path.join(__dirname, '/public/dist')));
// for load /admin/*
app.use('/',index);
// for api app
app.use('/api', api);
var port = config.get('port');
var server = http.createServer(app).listen(port, function () {
  console.log('Express server listening on port ' + port);
});
var io = require('socket.io').listen(server);
global.io = io;
module.exports = app;
```

Ниже приведён код scketch файла на стороне контроллера

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "utility/debug.h"
#include "utility/socket.h"
#include <stdio.h>
#include <stdlib.h>

#define CC3000_TINY_DRIVER
```

```
// These are the interrupt and control pins
#define ADAFRUIT_CC3000_IRQ   3  // MUST be an interrupt pin!
// These can be any two pins
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
// Use hardware SPI for the remaining pins
// On an UNO, SCK = 13, MISO = 12, and MOSI = 11
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
                                SPI_CLOCK_DIVIDER); // you can change this clock speed

Adafruit_CC3000_Client  client;

////////////////////////////////////////////////////////////////////////////
//  LEDs settings
////////////////////////////////////////////////////////////////////////////

#define LED_RECT_H  2
#define LED_RECT_W  5
const int ports_mapping[LED_RECT_H][LED_RECT_W] = {
  0,  1,  2,  7,  8,
  19, 18, 17, 16, 15
};

///////  Animations classes  ////////////////////////////////////////////////////

class Animation {
        public:
          Animation() {}
          virtual ~Animation() {}

          virtual void init() {}
          virtual void run(int ms_delay) {}

          void clear_rect()
          {
                for(int i = 0; i < LED_RECT_H; ++i)
                {
                for(int j = 0; j < LED_RECT_W; ++j)
                {
                digitalWrite(ports_mapping[i][j], LOW);
                }
                }
          }
```

```
};

class Animation_1 : public Animation{
        public:
          void init()
          {
                pos_1 = 0;
                old_pos_1 = 0;
          }

          void run(int ms_delay)
          {
                for(int i = 0;i < LED_RECT_H; ++i)
                {
                digitalWrite(ports_mapping[i][pos_1], HIGH);
                digitalWrite(ports_mapping[i][old_pos_1], LOW);
                }
                old_pos_1 = pos_1;
                pos_1 = (((pos_1 + 1) == LED_RECT_W) ? 0 : (pos_1 + 1));
                delay(ms_delay);
          }
        private:
          int pos_1;
          int old_pos_1;
};

//////////////  End  ////////////////////////////////////////////////////////////////

Animation * animations[5];
Animation * current;

void setup(void)
{
          initWiFiShield();

          initLEDsRect();
          initAnimations();

          connect2Host();
}

void loop(void)
{
          int count = 0;
```

```
	int num_buf[20];

	if(client) {
		if(client.available() > 0) {
			char buf[RECV_BUF_SIZE];
			int result = client.read(buf, RECV_BUF_SIZE);
			parseMsg(buf, num_buf, &count);
			if(count == 1)
			{
				setAnimation(num_buf[0]);
			}
			else {
				current->clear_rect();
				current = animations[5];
				StaticImage *st = (StaticImage *)current;
				st->setStates(num_buf);
				current->init();
			}
		}
	}
	current->run(100);
}

void initWiFiShield(void){
	/* Initialise the module */
	if (!cc3000.begin())
	{
		while(true);
	}

	if (!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
		while(true);
	}

	while (!cc3000.checkDHCP())
	{
		delay(100); // ToDo: Insert a DHCP timeout!
	}
}

void connect2Host(void){
	// Host settings
	uint32_t host_ip = cc3000.IP2U32(192,168,1,5);
	uint16_t host_port = 6969;
```

```
        // Attempt connect to server
        do {
                client = cc3000.connectTCP(host_ip, host_port);
                delay(100);
                //Serial.println("Another one\n");
        } while(!client);
}

void parseMsg(const char * str, int * arr, int * n){
        int pos = 0;
        int count = parseInt(str, &pos);

        for(int i = 0; i < count; ++i)
        {
                arr[i] = parseInt(str, &pos);
        }
        *n = count;
}
```