Министерство науки и высшего образования Российской Федерации
Лысьвенский филиал федерального государственного автономного образовательного учреждения высшего образования
«Пермский национальный исследовательский политехнический университет»

Факультет профессионального образования

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

на тему «Разработка многофункционального светодиодной строки для ИП Кобзарев О.О.»
студента группы КСК-9-18-1спо по специальности 09.02.01 Компьютерные системы и комплексы
Кобзарева Андрея Константиновича _____

Руководитель работы: _____М.Н. Апталаев

Консультант по
экономической части: _____К.В.Кондратьева

Консультант по промышленной экологии
и охране труда: _____А.К. Торощин

Рецензент: _____(_____)

Допуск к защите: _____М.Н. Апталаев

Лысьва, 2022 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном обществе маркетинг выходит на совершенно новый уровень, ещё в начале 21 века реклама эволюционировала в нечто новое, сейчас же старые маркетинговые трюки не дают того же эффекта, что раньше, и даже находятся далеко от своего прежнего коэффициента полезного действия. Сейчас и старая вывеска или билборд не привлекут должного внимания, будь даже они красиво выполнены. Прежние статичные приемы рекламы не содержат в себе «изюминки» и потенциала привлекать людской взор как раньше, люди все больше обращают внимание лишь на динамичный, яркий маркетинг, на фоне которого старые вывески блекнут.

Всё больший процент маркетинга подминает под себя сфера информационных технологий: взаимодействие со всемирной паутиной, использование электронных средств массовой информации, рекламы на веб-страницах и даже новые, яркие, динамичные электрические билборды и вывески.

Современные вывески и билборды забрали себе наибольшую долю рынка. Теперь же, став доступнее в стоимости и легче в монтаже и управлении такие технологические приспособления легко пришли в обиход даже самых обычных, небольших предпринимателей и сферу малого бизнеса, позволяя сосредотачивать на своих магазинах и предприятиях больше людского внимания.

Светодиодная матрица также является средством пассивного маркетинга, содержание её рабочей области с легкостью информирует и привлекает пеший людской трафик, а также пользуется высоким спросом даже в сферах малого бизнеса в угоду простоте, низкой стоимости и технологичности.

Разместив такую матрицу на своей торговой точке или предприятии предприниматель будет выделяться на фоне однотипных старых вывесок, больше привлекать людские взгляды к своему бизнесу, при этом, не неся значимых финансовых затрат.

Объектом исследования ВКР является уличный маркетинг.

Предметом исследования ВКР является проектирование светодиодной строки, предназначенной для воспроизведения изображения.

Целью выпускной квалификационной работы является разработка многофункциональной светодиодной строки.

Задачи выпускной квалификационной работы можно обозначить следующими аспектами:

– провести анализ предметной области и изучение требований для проектируемого устройства;

– выполнить подбор элементной базы, используемой в проекте, комплектация принципиальной схемы, печатной платы, программного кода;

– провести расчёт экономических параметров разработанного устройства;

– рассмотреть основы охраны труда и промышленной экологии.

1 ИССЛЕДОВАТЕЛЬСКИЙ РАЗДЕЛ

1.1 Понятие уличный маркетинг и его виды

«Наружная реклама – графическая, текстовая, либо иная информация рекламного характера, которая размещается на специальных временных или стационарных конструкциях, расположенных на открытой местности, а также на внешних поверхностях зданий, сооружений, на элементах уличного оборудования, над проезжей частью улиц и дорог или на них самих.

Средства наружной рекламы весьма разнообразны. Применительно к городской среде – это различные носители рекламных сообщений, размещаемые на территории города и рассчитанные на визуальное восприятие из городского пространства, а именно: крышные установки, электронные табло, панно, рекламные щиты, мультивизионные установки, кронштейны, маркизы, штендеры, перетяжки и т. п. [20].

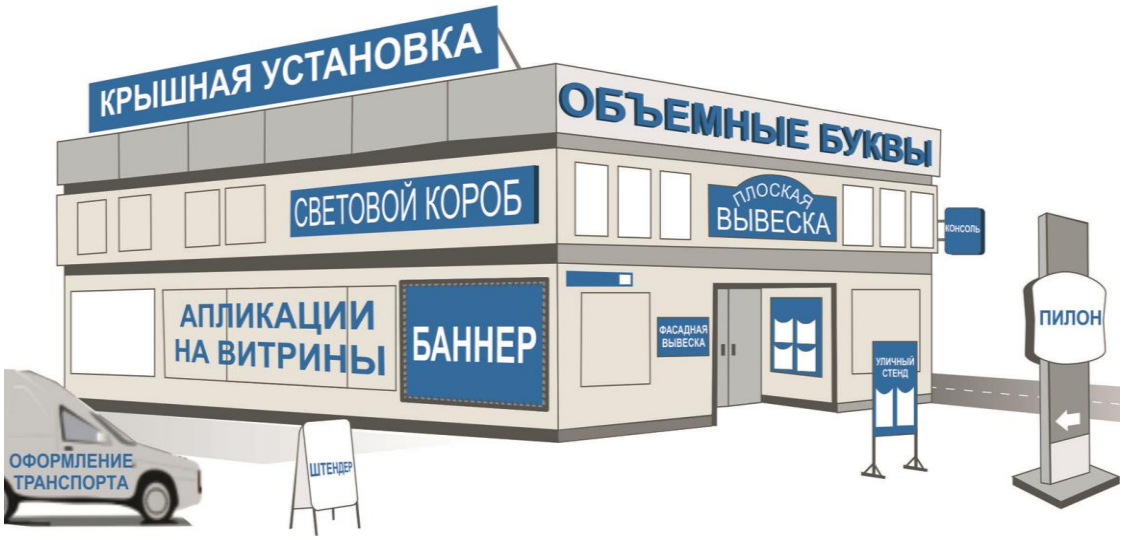

Рисунок 1 – Разновидности наружной рекламы

Все средства наружной рекламы делят на стационарные и временные и обязательно регистрируют. К стационарным средствам рекламы относят носители рекламных сообщений, имеющие постоянное место размещения. Временные средства наружной рекламы размещаются предприятиями в городской среде в часы их работы. Стационарные средства подразделяют на отдельно стоящие и размещаемые на зданиях и сооружениях. Отдельно

стоящие средства наружной рекламы, в отличие от установленных на зданиях, имеют самостоятельный фундамент или участок размещения.

Иногда к наружной рекламе относятся также рекламные сообщения, размещенные внутри магазинов/супермаркетов – конструкции POS (point of sale), экраны и другие, но чаще их выделяют в отдельный вид внутренней рекламы (indoor-реклама). Внутренняя и наружная реклама объединяются в рекламу Out-of-home (OOH).

Уличный маркетинг вбирает в себя все средства привлечения внимания к реализации товаров и услуг, которые расположены вне помещений.

Впервые уличный маркетинг заимел упоминания лишь в XIX веке, до этого предприниматели довольствовались лишь полями и строками в газетах. Начальные виды наружной рекламы совсем не были разнообразными, так как являлись только лишь плакатами разных величин, фиксируемыми на вертикальных поверхностях близ вокзала или на плацах и форумах.

Разновидности уличной рекламы:

Рисунок 2 – Рекламный щит

Рекламные щиты – очень простой способ предоставления различной информации. Они легко располагаются в чертах города, так же и вдоль дорог. Обычные билборды берут начало от стендов, которые появлялись повсеместно по дворам. Но уличные щиты на данный момент являют собой сразу множество различных вариаций форм уличного маркетинга. Они различны способами применения, типоразмерами и многими другим аспектами. [18].

Рисунок 3 – Вывеска

Вывеска – одна из прародительниц форм уличного маркетинга. Бывает в разных размерах и формах, зачастую имея собственную подсветку для акцентирования внимания. Располагаются только непосредственно на здании, которое рекламирует свои услуги.

Рисунок 4 – Световой короб

Световая реклама напоминает крышевые установки, так как в основном использует буквы и логотипы. Но подобные конструкции намного меньше крышного формата, поэтому используются в качестве светодиодной вывески, чтобы облегчить поиск посетителям в темное время суток[13].

Рисунок 5 – Светодиодная бегущая строка

Электронные табло и экраны – эта разновидность наружной рекламы отражает возможности человечества. Небольшие табло обычно отображают текстовую информацию бегущей строкой.

Светодиодной матрицей называется электронное устройство в виде табло, которое предназначено для вывода текстовых и графических данных. Бегущие строки активно применяются в наружной рекламе. Для привлечения внимания и интереса потребителей помимо рекламной информации на бегущую строку выводится полезная и актуальная информация о температуре окружающего воздуха, влажности, атмосферном давлении и т.д.

1.2 Понятие светодиодная строка и их виды

Светодиодной строкой называется электронное устройство в виде табло, которое предназначено для вывода текстовых и графических данных. Бегущие строки активно применяются в наружной рекламе. Для привлечения внимания и интереса потребителей помимо рекламной информации на бегущую строку выводится полезная и актуальная информация о температуре окружающего воздуха, влажности, атмосферном давлении и т.д.

Светодиодная строка работает совершенно автономно. Подключаться к ней нужно только для того, чтобы ввести необходимую и актуальную информацию, новый рекламный текст и прочее. В зависимости от модели способы подключения к бегущей строке могут быть совершенно разными. Какие-то бегущие строки требуют прямого подключения через USB или сетевой кабель, какие-то имеют возможность подключаться к ним удаленно, с помощью беспроводных способов связи [11].

Светодиодные строки используются как в помещении, так и на улице. Уличные модели, обычно располагаются на фасадах зданий и должны безотказно работать при любых климатических условиях, выдерживать перепады температур, сильный ветер, пыль, осадки и т.д.

Основное назначение бегущих строк – наружная реклама. Вывод информации о текущем времени, температуре воздуха и т.д. является второстепенным для разместившей устройство компании, но главной информацией для потребителей. Человек обращает внимание на бегущую строку во многом из-за того, что надеется обнаружить там нужную ему информацию. [17].

Для привлечения внимания к информации на табло различные модели этих устройств имеют разнообразные спецэффекты. Текст может выводиться различными нестандартными способами, с различной скоростью, слова могут “бежать” по одной букве и т.д. Некоторые модели позволяют формировать на экране не только текст, но и графические изображения, в том числе

анимированные. Яркие бегущие строки особенно заметны и эффективны в темное время суток.

Светодиодная матрица может состоять из нескольких модулей, поэтому длина строки практически неограниченна. Разновидности бегущих строк показаны на рисунке 6.

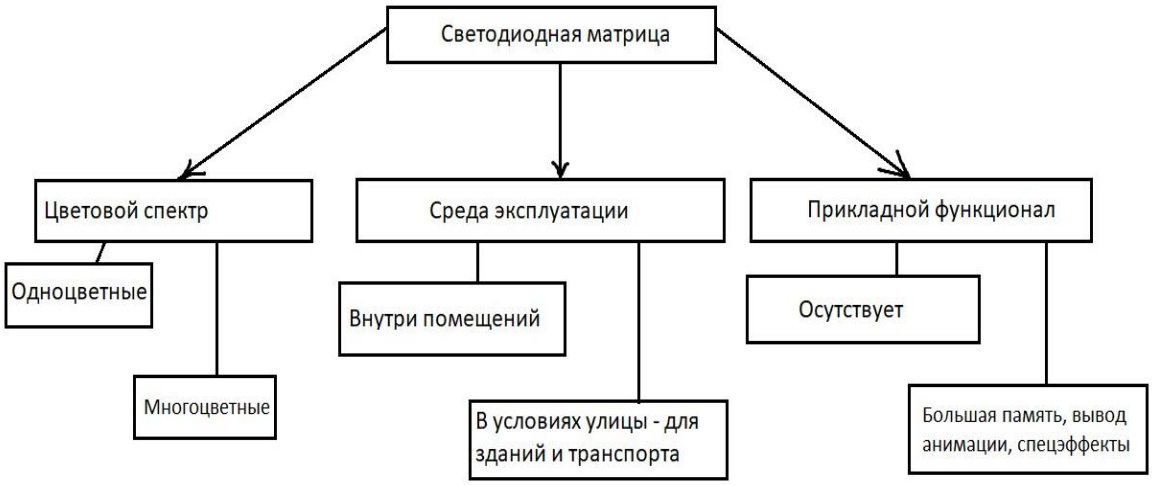

Рисунок 6 – Разновидности вариаций светодиодных матриц

Исходя из рисунка, можно понять, что вариаций бегущих строк существует огромное множество, поэтому большинство строк и изготавливается сугубо под заказ.

1.3 Принцип работы многофункциональной светодиодной матрицы

Принцип работы бегущей строки базируется на периодическом включении/выключении в определенном порядке комбинаций светодиодных ламп, размещенных на большом табло. В результате получается динамическая картинка – наблюдателю кажется, что текст или изображение движется.

Светодиодный экран представляет собой совокупность диодов, подключенных к контроллеру – специальному устройству, которое управляет работой табло, согласно заданной программы включая одни диоды и выключая другие.

Принцип работы светодиодного экрана тот же, что используется в современных телевизорах и единственное отличие заключается в отсутствии приемных устройств и прочих блоков, отвечающих за расшифровку телесигнала. Их роль выполняет один блок, в памяти которого хранится вся необходимая информация, благодаря которой на светодиодные экраны выводится изображение.

Шаг пикселя – расстояние между соседними светодиодами, чем оно меньше, чем более сложные изображения может воссоздавать светодиодный экран. От яркости зависит четкость получаемой картинки. Этот показатель особенно важен при установке табло на открытом месте, где на экран попадают прямые солнечные лучи. В таких обстоятельствах воссоздать четкую картинку можно только с помощью светодиодных экранов очень высокой яркости.

Принцип действия светодиодной бегущей строки заключается во взаимодействии, как с аппаратными, так и с программными средствами.

Основными элементами совокупности работы светодиодной матрицы являются:

– Микроконтроллер для хранения режимов и подключения основного алгоритма работы

– Блок питания для работы всех комплектующих системы от датчиков до светодиодного дисплея

– Светодиодный дисплей для отображения заданной информации

1.4 Анализ требований к устройству

Конфигурируя светодиодную матрицу, необходимо осознавать, в каких условиях она будет эксплуатироваться, а также какой набор функций нужен вам сегодня и может понадобиться завтра.

Важно обращать внимание в том числе на:

– Необходимые габариты табло исходя из места установки,

– Особенности места установки - от этого зависит тип светодиодов,

– Расстояние, с которого пользователи будут считывать информацию с табло – от этого зависит высота букв,

– Вертикальное и горизонтальное разрешение - от него зависит объем информации.

Корпус должен быть выполнен по стандарту защиты IP67 / IPX4 и выше, типоразмер корпуса не более 400x100x2000 мм, питание системы осуществляется непрерывно, с входным напряжением 220 Вольт для нужд размещения на торговых филиалах заказчика.

Рабочие температуры устройства должны составлять от -40 до +45 градусов Цельсия при влажности воздуха не более 90%, чтобы выдерживать температурные условия климатической зоны, в которой будет эксплуатироваться устройство.

Шаг светодиода должен составлять не менее 1,5см, для того чтобы массив светодиодов был виден людям на расстоянии не менее 20М и был различим текст с табло.

Светодиодной матрицей принято называть технологическое приспособление способное выводить изображение, работая со всеми типами графической информации, или с большинством из них. В эту работу входят такие действия как ввод, хранение, обработка, передача и вывод этой информации.

Для нормальной работы устройств и запуска графических файлов корректно, были придуманы расширения файлов.

Расширения файлов это - подпись файла, в которой содержится информация о файле, и указан тип данных к которому он принадлежит.

В таблице 1 можно видеть несколько наиболее популярных и часто используемых расширений файлов [2].

Таблица 1 - Примеры расширений

Тип информации	Расширение
Изображение	png, jpeg, ico, bpm, gif
Видео	wmv, mp4, mpeg
Аудио	mp3
Текст	txt

Передавать информацию и управлять состоянием строки можно при помощи прямого физического подключения, но также если устройство оснащено модулем беспроводной связи – можно сопрягаться не подключаясь вручную.

Беспроводное соединение – это сопряжение двух и более точек связи, позволяющая объединить потоки данных и совершать обмен между источниками информации без использования физических соединений, то есть при помощи радиоволн.

«Wi-Fi – технология беспроводной локальной сети с устройствами на основе стандартов IEEE 802.11. Логотип Wi-Fi является торговой маркой Wi-Fi Alliance. Под аббревиатурой Wi-Fi (от английского словосочетания Wireless Fidelity, которое можно дословно перевести как «беспроводная точность») в настоящее время развивается целое семейство стандартов передачи цифровых потоков данных по радиоканалам. Основными диапазонами Wi-Fi считаются 2.4 ГГц (2412 МГц-2472 МГц) и 5 ГГц (5160-5825 МГц). Сигнал Wi-Fi может передаваться на километры даже при низкой мощности передачи, но для приема Wi-Fi-сигнала с обычного Wi-Fi-маршрутизатора на далеком расстоянии нужна антенна с высоким коэффициентом усиления (например параболическая антенна или Wi-Fi-пушка).

Стандарт IEEE 802.11n был утверждён 11 сентября 2009 года. Его применение позволяет повысить скорость передачи данных практически вчетверо по сравнению с устройствами стандартов 802.11g (максимальная

скорость которых равна 54 Мбит/с), при условии использования в режиме 802.11n с другими устройствами 802.11n. Теоретически 802.11n способен обеспечить скорость передачи данных до 600 Мбит/с. С 2011 по 2013 разрабатывался стандарт IEEE 802.11ac, стандарт принят в январе 2014 года. Скорость передачи данных при использовании 802.11ac может достигать нескольких Гбит/с. Большинство ведущих производителей оборудования уже анонсировали устройства, поддерживающие данный стандарт».[9].

1.5 Анализ существующих решений

Учитывая тот факт, что размещение бегущих строк требует особенных габаритов, зависящих от места установки можно понять, что серийное производство с определенным типоразмером наладить невозможно. Также и условия эксплуатации совершенно разные для организаций-заказчиков, много различных функциональных вариаций и их использование зависит от заказчика, поэтому, серийно выпускать укомплектованные наборы для сборок тоже не имеют смысла.

В локальной близости есть лишь пара мастерских, находящихся в городе Пермь, они занимаются изготовлением вывесок на заказ, не имеют маркировок и сертификации товара, что законодательно позволяет им изготавливать свою продукцию лишь на заказ.

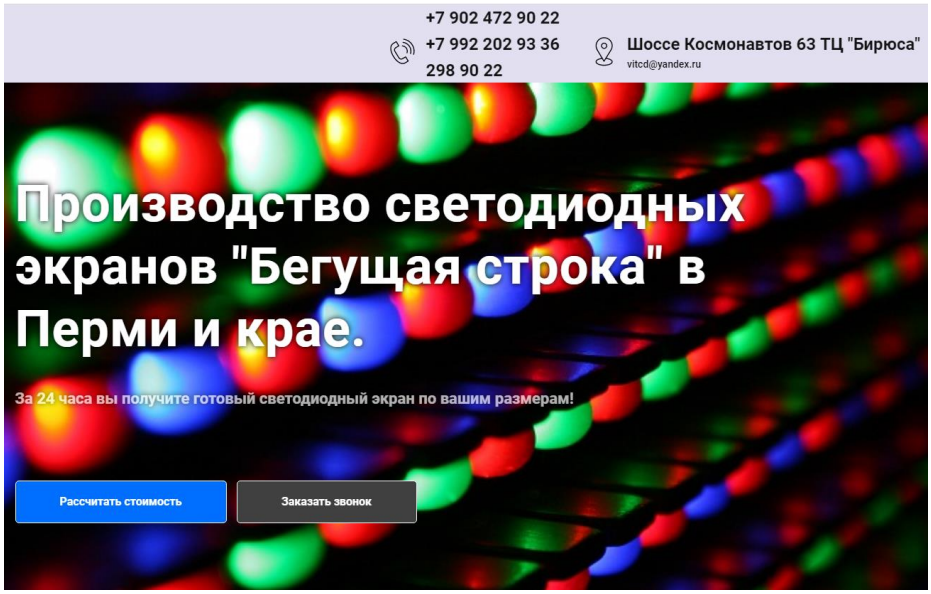

Рисунок 7 – Web-сайт STROKA59

STROKA59 – Самое большое предприятие в Пермском крае по изготовлению бегущих строк, его деятельность концентрируется лишь на бегущих строках

Самый большой производитель световой уличной рекламы в Пермском крае, его деятельность не ограничивается одними лишь светодиодными строками.

Цены на продукцию данных фирм достигают до 400% себестоимости бегущей строки, изготовление строки, подобной создаваемой для ВКР приблизительно обойдется в сумму от 19000 до 42000 рублей.

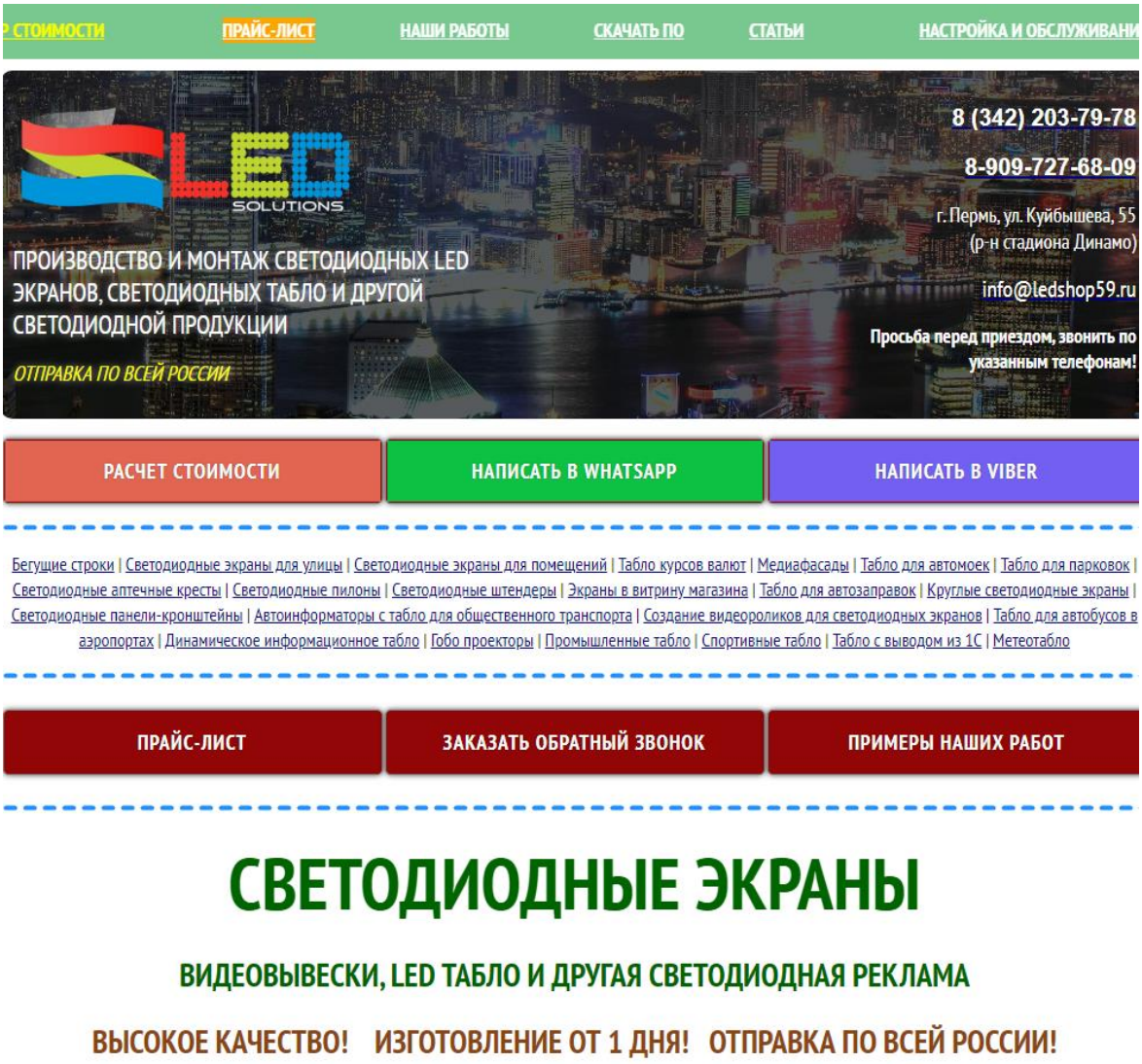

Рисунок 8 – Web-сайт Ledshop59

И действительно – на рынке РФ нет ни одного серийного производителя бегущих строк, всё выпускается небольшими мастерскими и собирается кустарно, под заказ и индивидуальные нужды.

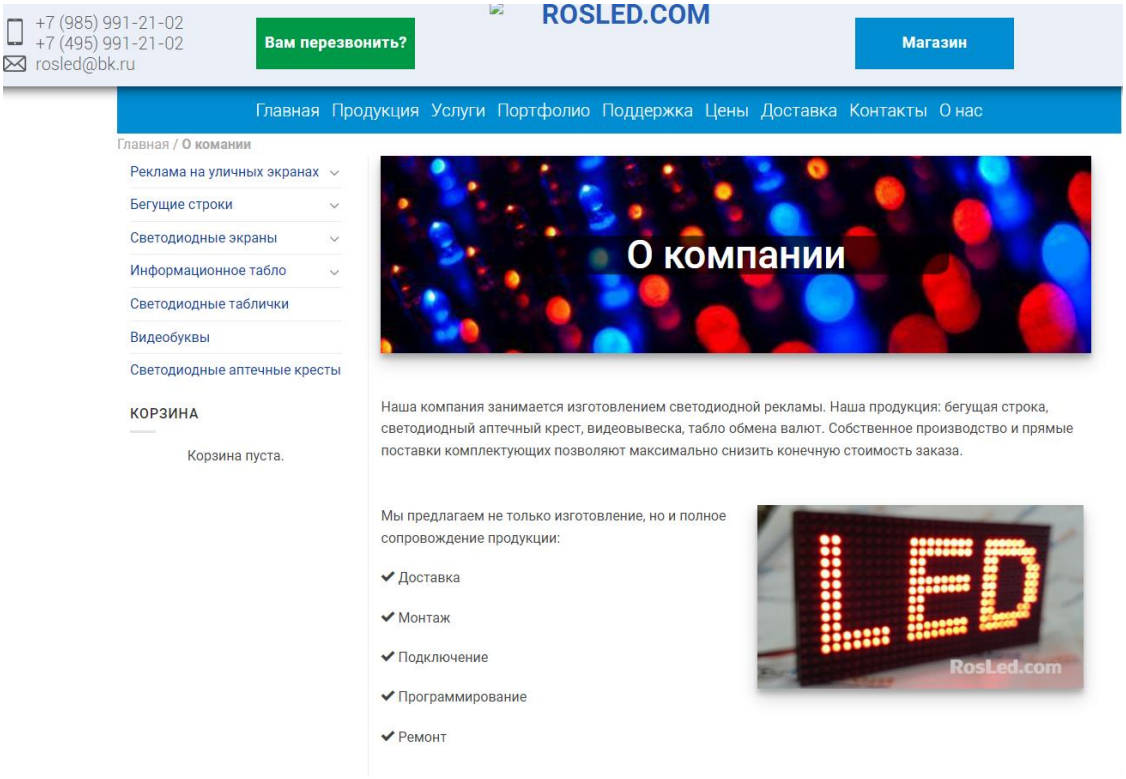

Рисунок 9 – Web-сайт Rosled

Всего лишь одна компания в РФ имеет свое конкретное и законное производство, сопровождающееся правильной маркировкой, согласно которой товар можно считать качественным по законам РФ. Тем не менее, производство не типизировано и остаётся кустарным. [12]

Именно поэтому бегущие строки часто собираются различными энтузиастами, например Alex Gyver, известный блогер в сфере изготовления самодельных устройств на базе микроконтроллеров, также собирал бегущие строки и различные конструкции с адресными светодиодными массивами. Обычно его устройства очень простые, недорогие в изготовлении и собраны во многом из доступных и подручных деталей.

1.6 Выводы по разделу

В результате исследовательской части выпускной квалификационной работы был изучен уличный маркетинг, рассмотрены основные вариации рекламы на улицах, также отдельно была рассмотрена бегущая строка и все её разновидности, области применения, места установок. Был изучен принцип работы бегущей строки, состав и совокупность её компонентов, проведены аналогии работы с другими видами дисплеев, проанализированы требования к проектируемому устройству, так как в зависимости от вида бегущей строки и сферы её использования требования очень сильно разнятся, из-за места, цели, климата, времени в котором проектируемое устройство будет эксплуатироваться заказчиком. Также был рассмотрен и рынок бегущих строк, где их можно купить, изучена медиана цен, в результате чего было выявлено, что из-за множества вариаций бегущих строк в большинстве случаев не выйдет приобрести устройство с необходимым набором функций, были рассмотрены компании, производящие бегущие строки на заказ под цели заказчика, а также и изучены способы инженеров-энтузиастов, которые собирают бегущие строки своими руками при незамысловатой конструкции.

2 КОНСТРУКТОРСИЙ РАЗДЕЛ

2.1 Проектирование структуры разрабатываемого устройства

Для начала разработки устройства необходимо выделить в системе структурные блоки, их выполняемые функции и совокупность взаимосвязей меж ними. После этого будет готова структурная схема устройства.

Для проектируемого устройства можно выделить следующие блоки: блок светодиодных матриц, управления, блок питания, а также блок беспроводного соединения.

Блок питания включает в себя элементную базу обычного компьютерного блока питания с активным охлаждением и подключённым к нему разъёму питания для сети 220В.

В блок управления входит банк памяти, кнопка и NodeMCU. Интерфейс реализуется на отдельном устройстве с операционной системой Android, Windows или Linux. Данный блок позволяет взаимодействовать с режимами устройства и задавать желаемый текст и эффекты. NodeMCU отвечает за управление основными функциями системы, а также она предназначена для сбора и обработки информации.

В блок светодиодных матриц входит массив светодиодов, задачей матрицы является вывод заданной пользователем информации.

Отдельным блоком можно выделить блок беспроводного соединения. Данный блок имеет в своём составе модуль.

Взаимосвязь между блоками может быть, как однонаправленной, так и двунаправленной. Разница между этими понятиями заключается в направлении передачи сигнала или данных. В случае однонаправленного сигнала данные всегда передаются от одного блока к другому, и поддержка обратного ответа не предусмотрена, в случае двунаправленной связи, все блоки, соединённые этой связью, могут быть как приёмниками, так и отправителями.

Связи с блоком питания установлены со всеми остальными блоками, поскольку каждому элементу необходимо питание.

На рисунке 10 изображена полученная структурная схема разрабатываемого устройства.

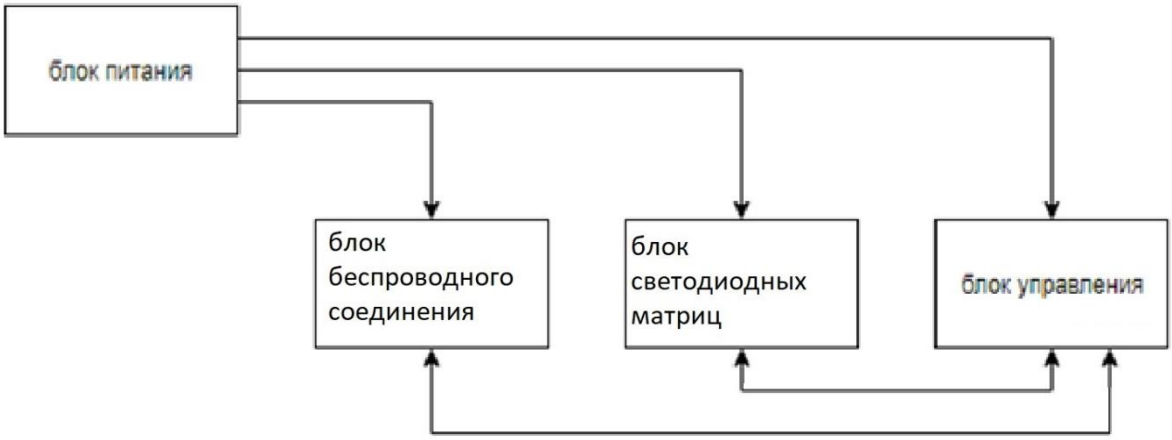

Рисунок 10 – Структурная схема устройства

Принципиальная схема разрабатывалась в специализированной САПР, из нескольких популярных САПР на рынке (EasyEDA, DipTrace и Kicad) была выбрана именно EasyEDA, так как к САПР выдвигаются следующие требования: возможность разработки печатной платы по принципиальной схеме, автотрассировка, возможность добавления элементов пользователем, отображение печатной платы в формате 3D и возможность экспорта полученных схем в формат png.

EasyEDA – САПР для разработки принципиальной схемы, с возможностью перевода схемы в режим разработки печатной платы. Имеется возможность автотрассировки дорожек платы, готовый вариант можно посмотреть в 3D варианте плате. Имеется online версия программы, для которой требуется лишь авторизация, без необходимости скачивать приложение.

DipTrace – программный продукт позволяющий выполнять разработку принципиальных схем и печатных плат. Предоставляет возможность просмотра разработанной печатной платы в формате 3D модели, с

установленными элементами. При работе имеется возможность использовать автотрассировщик.

Kicad – это САПР для разработки как принципиальной схемы, так и печатной платы различных устройств. В программе имеется возможность работы с различными существующими элементами схемы, а также добавления своих. САПР не поддерживает возможности автотрассировки, а также не может выводить полученную печатную плату в 3D форме.

Сравнение представленных программ рассматриваем в таблице 2.

Таблица 2 – Сравнение программ для разработки принципиальной схемы

	EasyEDA	DipTrace	Kicad
Возможность добавления пользователями элементов схемы	+	+	+
Разработка печатной платы	+	+	+
Отображение 3D модели разработанной печатной платы	+	+	-
Возможность экспорта принципиальной схемы и схемы печатной платы в png	+	+	+
Возможность автотрасировки	+	+	-

Проанализировав таблицу 3, можно сделать вывод что наиболее подходящим вариантом для разработки принципиальной схемы и печатной платы является САПР EasyEDA, поскольку Kicad не удовлетворяет требованиям возможности автотрассировки и отображения печатной платы в 3D форме, а DipTrace не имеет возможности работы в online режиме, это потребует времени на установку.

После выбора программы для разработки принципиальной схемы, начинается разработка принципиальной схемы, на которой обозначаются все используемые комплектующих, их контакты и обозначения, также на принципиальной схеме выстраивается связь между контактами элементов схемы.

На рисунке 11 изображена разработанная принципиальная схема устройства со всеми подключенными системами.

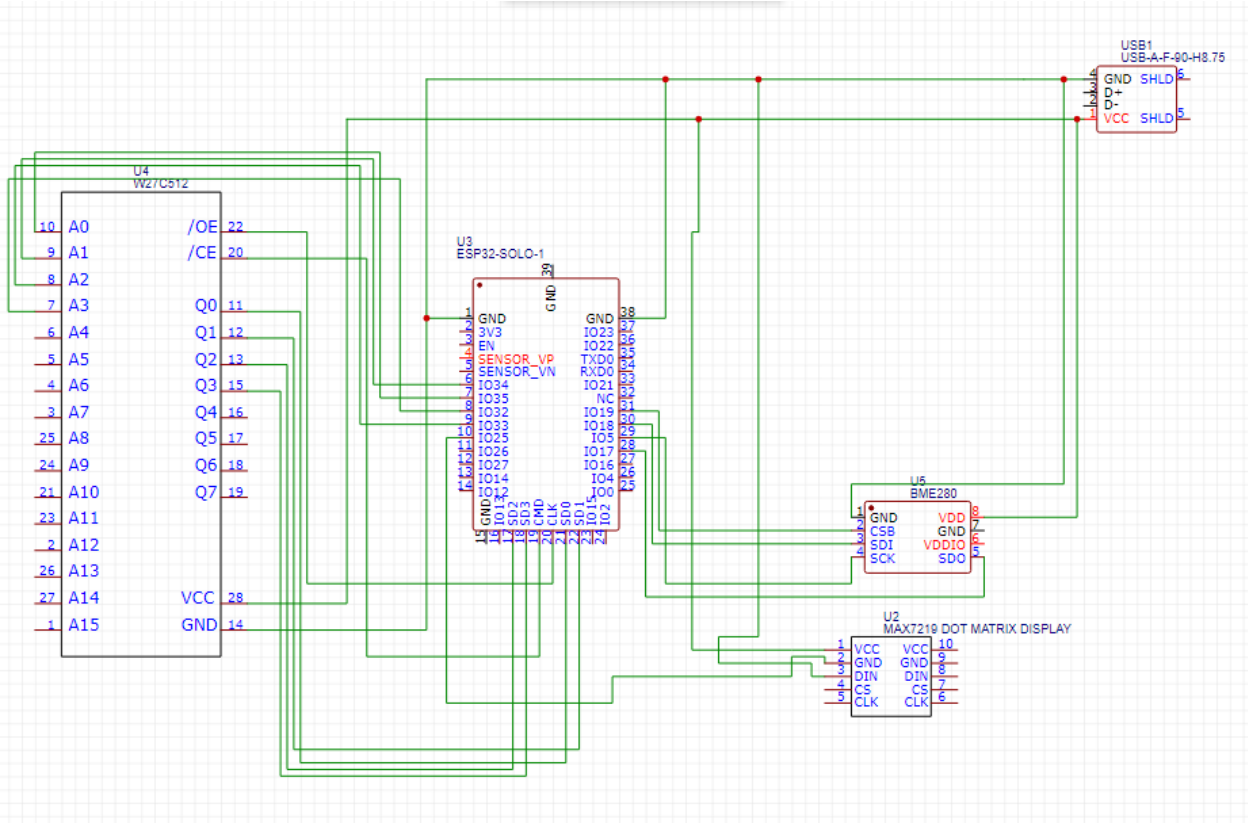

Рисунок 11 – Принципиальная схема устройства

Также, при помощи принципиальной схемы была спроектирована печатная плата устройства, изображенная на рисунке 12

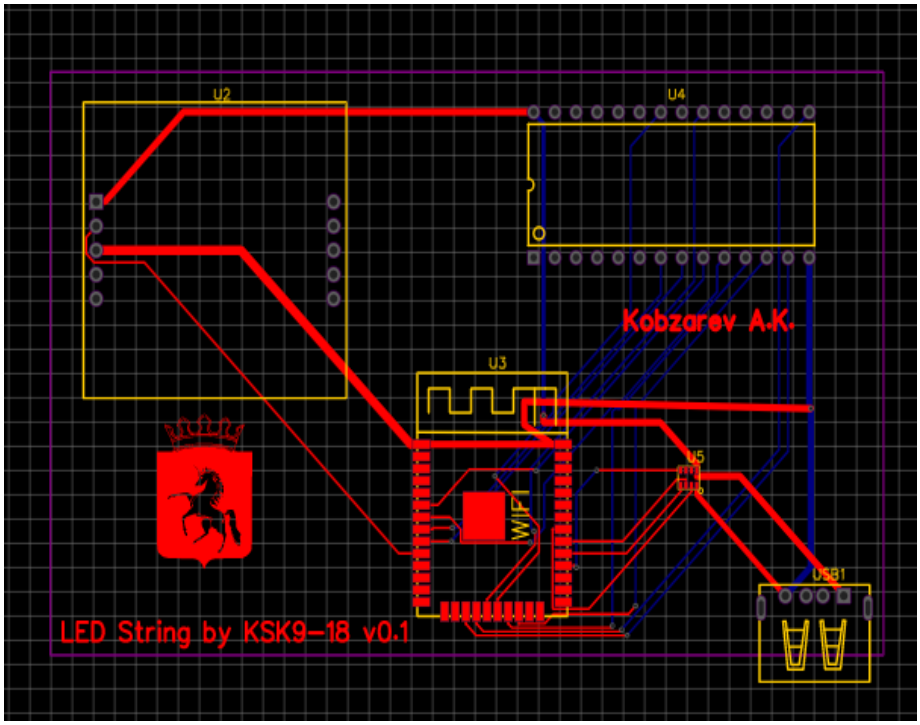

Рисунок 12 – Печатная плата устройства

Основным источником питания системы является блок питания постоянного тока, который имеет номинальное напряжение ~5 вольт и силу тока равную ~12 ампер. Источник питания подключается к портам питания с помощью прямого соединения (пайки).

К системе беспроводной передачи данных относится Wi-Fi модуль ESP-12 для управления и контроля бегущей строки, который обозначает как U4. После получения сигнала от пользователя Wi-Fi модуль передает информацию микроконтроллеру для изменения состояния светодиодной матрицы.

К системе светодиодных матриц относится адресная светодиодная лента, которой обозначается как U2. Её функция довольно проста, и сводится к тому, чтобы менять состояние при подаче управляющего сигнала в соответствии с заданной информацией.

Для упрощения понимания внешнего физического строения устройства была разработана эскизная схема, изображенная на рисунке 13.

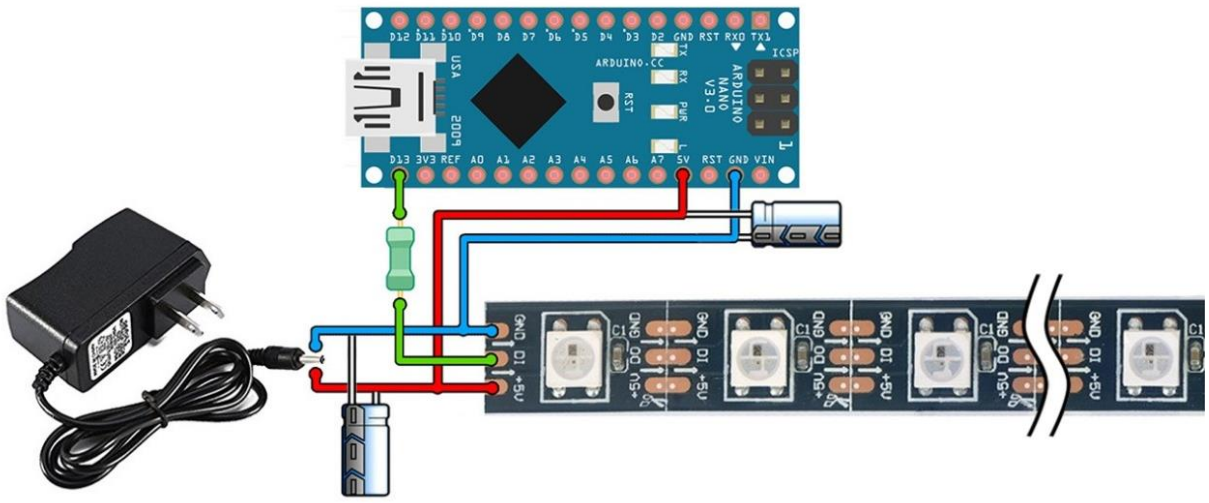

Рисунок 13 – Эскизная схема устройства

Данная схема демонстрирует простоту конструкции бегущей строки, а также массивность размеров её компонентов при масштабировании.

2.2 Выбор комплектующих для разрабатываемого устройства

К сравнению представлены микроконтроллеры NodeMCU, ATmega2560, ATmega32u4, ATmega168P, STM32. Для сравнения были выбраны эти микроконтроллеры по причине того, что они удовлетворяют требованиям по минимальному количеству линий ввода-вывода и интерфейсу UART. Далее данные микроконтроллеры будут сравниваться по своей производительности и цене. Сравнение перечисленных микроконтроллеров представлено в таблице 2.

Таблица 2 – Сравнение микроконтроллеров

Название	NodeMCU	Atmega2560	Atmega32u4	ATmega168	STM32
Цена, руб.	183	740	310	250	412
Тактовая частота, МГц	160	16	16	20	96
Флеш-память, Кб	128	256	32	16	64
Линии ввода-вывода	20	54	20	20	32
UART	1	4	1	1	1

После выполнения сравнения параметров микроконтроллеров был выбран микроконтроллер NodeMCU, поскольку его стоимость гораздо ниже аналогов, и он обладает хорошим быстродействием. У данного микроконтроллера подходящее количество цифровых и аналоговых выходов и идеально подходит для устройства по критерию цена/производительность.

- Плата NODEMCU V3;
- Конденсаторы 4,7В 1000мкФ и 4,7В 470мкФ;
- Блок питания 5В 10А;
- Адресная светодиодная лента 60 светодиодов/метр 5В 12М;
- Резистор 220Ом ;
- Корпус уровня защиты IPX4/IP67
- Рассеиватель света с пропуском светового потока не менее 30% [14].

2.3 Разработка программы и устройства

Рассмотрев различные варианты, было принято решение взять задумки и сильные стороны проектов VVIP68, Alex Gyver, ArduinoPlus. Они являются профессионалами в сфере проектирования и сборке устройств на основе Arduino, в своих блогах они подробно описывают весь процесс[10].

Для разработки управляющей программы необходимо выбрать среду разработки программного кода для микроконтроллера. Для этого будет выполнен выбор среди сред: Micro chip, MPLAB, WinAVR.

К рассматриваемым средам разработки выдвигаются данные требования: встроенный компилятор для C++, встроенный программатор, возможность отладки, возможность симулирования работы микроконтроллера, работа на Windows 10 и способ распространения.

Arduino IDE – среда разработки от компании Arduino, имеет статус свободно распространяемого ПО для энтузиастов, имеет удобный инструментарий поиска, использования и установки библиотек. Среда разработки позволяет писать код на языке программирования C++ или на Assembler. Имеется встроенный симулятор работы микроконтроллера, который может показать, что записано в регистры и память микроконтроллера, с учётом разных моделей.

MPLAB – среда разработки являющаяся набором программ для работы с кодом микроконтроллера. MPLAB позволяет писать и отлаживать код на C++ или Assembler, для этого имеется встроенный компилятор кода.

WinAVR – среда разработки для написания кода микроконтроллеров различных семейств. В данном программном продукте имеется возможность написания и отладки кода, без возможности отображения состояния регистров. Интерфейс программы наиболее прост в освоении и работе.

Сравнение рассмотренных сред разработки показано в таблице 3.

Таблица 3 – Сравнение функций сред разработки

	ArduinoIDE	MPLAB	WinAVR
Встроенный компилятор для C++	+	+	+
Возможность отладки	+	+	+
Возможность симуляции	+	+	-
Свободно распространяемое ПО	+	+	+
Встроенный программатор	+	+	+
Поддержка работы на Windows 10	+	+	-

Выполнив анализ таблицы 3, была выбрана среда разработки Arduino IDE, по причине того, что она наиболее подходит под требуемые задачи, а также легко взаимодействует, ищет и устанавливает библиотеки. MPLAB является более серьёзным и сложным в освоении инструментом. WinAVR не имеет симуляции, а также имеет проблемы при работе на Windows 10.

После выбора среды разработки был выбран язык программирования Arduino Wiring, который имеет схожих синтаксис с C++, имея лишь дополнительные функции, позволяющие сделать написание кода наиболее удобным, а также отсутствие лишних команд и повсеместное упрощение.

Для написания программного кода необходимо разработать блок-схему алгоритма работы устройства. На рисунке 14 можно видеть блок-схему алгоритма работу программы для микроконтроллера.

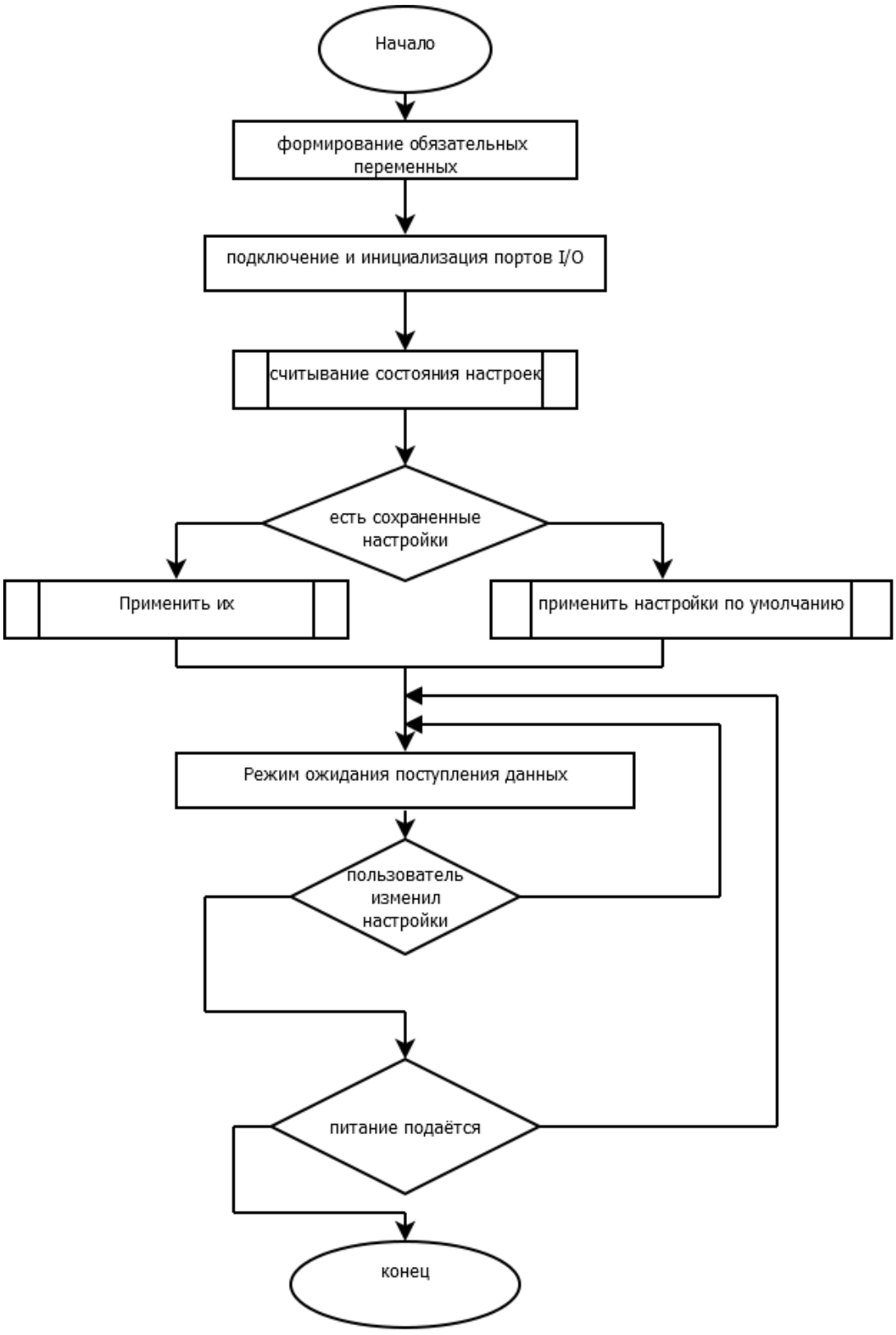

Рисунок 14 – Блок–схема алгоритма работы устройства

У Alex Gyver было заимствовано приложение для смартфонов на базе Android, у VVIP68 была заимствованная часть скетчей кода, содержащая в

себе игры, эффекты, а также сопряжение по Wi-Fi, у ArduinoPlus был заимствован набор универсальных шрифтов для вывода на дисплей матрицы.

Для физической сборки системы необходимо выполнить следующие действия:

Сборка матрицы:

– Лудим необходимые для соединения контакты на плате блоке питания и матрице;

– Начинаем замеры расстояний гибких соединений проводниками ;

– Осуществляем пайку контактов ленты параллельным методом построения светодиодных матриц;

– Выводим провода блока питания вдоль контактных площадок лент матрицы;

– Паяем провода питания к матрице по направлению, указанному на адресных лентах;

– Оснащаем линии данных и питания необходимыми конденсаторами и резисторами;

– Изолируем все места соединения термопрокладками или силиконовыми жидкими изолирующими средствами;

Сборка корпуса:

– Изготавливаем заготовки для стенок корпуса из полимеров;

– Обрабатываем и скрепляем элементы задней крышки вместе;

– Фиксируем заднюю часть крышки вместе с проводкой питания;

– .Размещаем микроконтроллер и светодиодную матрицу внутри задней крышки корпуса

– Соединяем вместе элементы передней крышки корпуса;

– Вырезаем оргстекло и частично прозрачный материал для рассеивания света

– Собираем переднюю крышку;

– Устанавливаем соединительные элементы двух частей корпуса на трвёрдые соединения или магниты;

– Проверяем проводку на наличие изломов;

– Соединяем части корпуса в единое целое;

Подключение к компьютеру:

– Припаиваем все элементы к плате с помощью кабелей и проводов;

– Подключаем плату к компьютеру и загружаем прошивку для тестирования работоспособности ленты (управляющая программа целиком представлена в приложении А);

– Проверяем на работоспособность.

– Загружаем прошивку рабочей версии системы [Прошивка представлена в приложении А];

– В программе ADRUINO выбираем плату NODEMCU v3 и порт, на котором она находится;

– Проверяем на работоспособность.

Отладка:

– Поставить DEBUG 1, загружаем прошивку;

– Открыть монитор порта;

– Менять режимы, цвета, эффекты, осуществлять переподключение к системе. Смотреть на значения, должны меняться;

– Если значения меняются правильно, поставить DEBUG 0, и загружаем прошивку.

Достоинства сборки:

Водонепроницаемость;

– Устойчивость от перепадов напряжения;

– Беспроводное сопряжение;

– Малые затраты расходников;

– Цена.

– Долговечность

– Компактность внутреннего устройства

– Охват людских взглядов с дальнего расстояния

Недостатки сборки:

– Проблематичность ремонта отдельных светодиодов на табло

– Плохой охват взглядов в яркие, светлые дни.

– Величина и сложность пайки.

– Необходимость источника питания с высокой силой тока из-за использования низкого напряжения всеми компонентами [3].

2.4 Выводы по разделу

В результате выполнения конструкторского раздела была спроектирована конструкция всего устройства, благодаря основным требованиям были составлены структурная, эскизная, принципиальная схемы, путём сравнения была подобрана необходимая элементная база устройства.

Также была спроектирована печатная плата с распределением и направлением соединений проводников и электронных элементов на ней.

Была составлена блок-схема принципа работы программного кода, в результате чего были выбраны среда программирования и написан программный код на упрощённом языке C++ – Wiring.

В результате работы были изучены сильные и слабы стороны создаваемого устройства

3 ОРГАНИЗАЦИОННО–ЭКОНОМИЧЕСКАЯ ЧАСТЬ

3.1 Цель и основные результаты внедрения разработки

Целью внедрения бегущей строки является привлечение покупательной способности торговой точки заказчика путем обращения взоров людей, проходящих мимо на информационное светодиодное табло и их информирования о различных преимуществах покупки продукции в данном торговом пункте заказчика.

Внедрение АСУ будет проходить на основе технического задания и договора, составленного между исполнителем и заказчиком.

Ожидаемая экономическая эффективность достигается за счёт разницы цен между проектируемым устройством, устройствами, производимыми на заказ, а также устройствами продаваемыми в рознице (по данным Яндекс.Маркета)[21]. Учитывая стоимость данного производимого устройства в 14000 и среднее арифметическое цен рынка, равное 28400, можно сделать вывод, что экономическая выгода от отказа в покупке или заказе многофункциональной бегущей строки составит 14400 рублей

3.2 Исходные данные для расчетов

В хозяйственно-операционные нужды были включены параметры, представленные в таблице

Таблица 4 – Состав затрат на хозяйственно-операционные нужды

№	Наименование	Цена за единицу, руб.	Количество, шт.	Всего, руб.
	Картридж	170	1	170
	Бумага	0,2	200	40
	Итого			210

Состав затрат на хозяйственно-операционные нужды не был обширным, так как требовалось только распечатать документацию

Таблица 5 – Исходные данные для расчета затрат на разработку и внедрение ИТ

Наименование показателя	Условное обозначение	Единица измерения	Значение показателя
Оплата труда подрядного программиста	$О_п$	руб.	2000
Комплектующие	$Ц_к$	руб.	6514
Монтаж	$Ц_м$	руб.	1000
Доставка	$Ц_д$	руб.	276
хозяйственно-операционные нужды	$Ц_х$	руб.	210
Итого	$З_{рпр}$		10000

Расчёты на разработку указанные в таблице 5 говорят о действительной небольшой стоимости светодиодных строк, о их доступности.

3.3. Расчет затрат на создание АСУ

Цена программного продукта, который разработан одной организаций по заказу другой и не предназначен для тиражирования, определяется по формуле:

$$Ц_{ПП} = З_{РПР} + П_п + НДС \ , \quad (1)$$

Формула 1

где $Ц_{ПП}$ – цена программного продукта, руб.;

$З_{РПР}$ – затраты на разработку проектного решения, руб.;

$П_п$ – планируемая прибыль, руб.;

$НДС$ – налог на добавленную стоимость, руб.

$З_{РПР} = 10000$ руб.

Планируемая прибыль рассчитывается по формуле 2.

$$П_п = З_{РПР} \times R_{НПП} \ , \quad (2)$$

Формула 2

где $R_{НПП}$ – нормативная рентабельность ПП, определяемая организацией.

$П_п = 10000 \times 0{,}4 = 4000$ руб.

НДС, начисленный на ПП, определяется следующим образом, указанным в формуле 3.

$$НДС = (З_{РПР} + П_{п}) \times k_{НДС}, \quad (3)$$

Формула 3

где $k_{НДС}$ – ставка налога на добавленную стоимость.

$НДС = (10000 + 4000) \times 0{,}2 = 2800$ руб.

$Ц_{ПП} = 10000 + 4000 + 2800 = 16800$ руб.

Расчет налога на прибыль:

$НП = П_{п} \times k_{НП},$

где $НП$ – величина налога на прибыль, руб.;

$k_{НП}$ – ставка налога на прибыль.

$НП = 4000 \times 0{,}24 = 960$ руб.

Прибыль, оставшаяся в организации после уплаты налогов, определяется по формуле:

$$П_{ч} = П_{п} - НП. \quad (4)$$

Формула 4

$П_{ч} = 4000 - 960 = 3040$ руб.

Поступления в госбюджет (*БП*) от реализации проекта составят:

$БП = НП + НДС$.

$БП = 960 + 2800 = 3760$ руб.

3.4. Выводы по разделу

Был осуществлён подсчёт основных экономических параметров внедрения устройства, была посчитана себестоимость устройства.

Также были произведены вычисления величины налоговых отчислений, подробно разобраны расходы на внедрение проектируемого устройства

Также были установлены цель и основные результаты внедрения разработки, из которых ясно, что результат, ожидаемый от устройства, – это повышение прибыли с торговых пунктов заказчика путем привлечения дополнительных клиентов.

Учитывая экономию из-за отказа от покупки готовой аналогичной строки в местных компаниях, экономическая эффективность составит в среднем 14400 рублей.

4 ОХРАНА ТРУДА И ПРОМЫШЛЕННАЯ ЭКОЛОГИЯ

4.1 Анализ вредных и опасных факторов на рабочем месте инженера-электроника

Опасные и вредоносные производственные моменты по природе происхождения различают следующие группы: [19]

– физические;

– химические;

– психофизиологические;

– биологические.

В помещении на инженера-программиста могут отрицательно влиять следующие физиологические факторы:

– высокая и низкая температура воздуха;

– излишняя запыленность и загрязненность воздуха;

– увеличенная и пониженная влажность воздуха;

– недостающая освещенность трудового места;

– превышающий дозволенные нормы шум;

– высокий уровень ионизирующего излучения;

– высокий уровень электромагнитных полей;

– поднятый уровень статического электричества;

– опасность поражения электрическим током;

– тусклость экрана дисплея.

К химически небезопасным факторам, безостановочно функционирующим на инженера-программиста относятся функциональные частицы, появляющиеся в результате ионизации воздуха при работе компьютера. К психологически вредоносным факторам, воздействующим на человека в процессе его рабочего дня относятся следующие:

–эмоциональные перегрузки;

–умственная усталость;

–перенапряжение глаз.

Опасные и вредные производственные факторы по природе возникновения делятся на следующие группы:

- физические;
- химические;
- психофизиологические;
- биологические.

В помещении на программиста могут негативно действовать следующие физические факторы:

- повышенная и пониженная температура воздуха;
- чрезмерная запыленность и загазованность воздуха;
- повышенная и пониженная влажность воздуха;
- недостаточная освещенность рабочего места;
- превышающий допустимые нормы шум;
- повышенный уровень ионизирующего излучения;
- повышенный уровень электромагнитных полей;
- повышенный уровень статического электричества;
- опасность поражения электрическим током;
- блеклость экрана дисплея.

К химически опасным факторам, постоянно действующим на программиста относятся активные частицы, возникающие в результате ионизации воздуха при работе компьютера.

К психологически вредным факторам, воздействующим на программиста в течении его рабочей смены можно отнести следующие:

- нервно-эмоциональные перегрузки;
- умственное напряжение;
- перенапряжение зрительного анализатора.

4.2 Мероприятия по снижению воздействия выявленных вредных и опасных производственных факторов

Санитарно-гигиенические процедуры ориентированы на создание безопасных условий труда. К ним относятся:

– Гигиеническая стандартизация;

– Контролирование состояния воздушной среды;

– Соблюдение гигиенических условий среди повышенной угрозы воздействия ядов (аварийные ситуации, ремонтные работы);

– Использование средств защиты и вентиляция.

Лечебно-профилактические мероприятия сориентированы на предостережение зарождения производственных отравлений и заболеваний. К ним относятся:

– Предварительный при поступлении на работу и последующие периодические мед осмотры;

– Организация специального питания;

– Ультрафиолетовое кварцевание рабочего места;

– Дыхательная гимнастика.

Список средств и мероприятий, способных снизить факторы риска на предприятии следующий:

– Средства нормализации воздушной среды производственных помещений и рабочих мест: устройства для поддержания нормируемой величины барометрического давления; вентиляции и очистки воздуха; кондиционирования воздуха; локализации вредных факторов; отопления; автоматического контроля и сигнализации; дезодорации воздуха.

– Средства нормализации освещения производственных помещений и рабочих мест: источники света; осветительные приборы; световые проемы; светозащитные устройства; светофильтры.

– Средства защиты от повышенного уровня ионизирующих излучений: оградительные устройства; предупредительные устройства;

герметизирующие устройства; защитные покрытия; устройства улавливания и очистки воздуха и жидкостей; средства дезактивации; устройства автоматического контроля; устройства дистанционного управления; средства защиты при транспортировании и временном хранении радиоактивных веществ; знаки безопасности; емкости для радиоактивных отходов.

– Средства защиты от повышенного уровня инфракрасных излучений: оградительные устройства; герметизирующие устройства; теплоизолирующие устройства; вентиляционные устройства; устройства автоматического контроля и сигнализации; устройства дистанционного управления; знаки безопасности.

– Средства защиты от повышенного или пониженного уровня ультрафиолетовых излучений: оградительные устройства; устройства для вентиляции воздуха; устройства автоматического контроля и сигнализации; устройства дистанционного управления; знаки безопасности.

– Средства защиты от повышенного уровня электромагнитных излучений: оградительные устройства; защитные покрытия; герметизирующие устройства; устройства автоматического контроля и сигнализации; устройства дистанционного управления; знаки безопасности.

– Средства защиты от повышенной напряженности магнитных и электрических полей: оградительные устройства; устройства защитного заземления; изолирующие устройства и покрытия; знаки безопасности.

– Средства защиты от повышенного уровня лазерного излучения: оградительные устройства; предохранительные устройства; устройства автоматического контроля и сигнализации; устройства дистанционного управления; знаки безопасности.

– Средства защиты от повышенного уровня шума: оградительные устройства; звукоизолирующие, звукопоглощающие устройства; глушители шума; устройства автоматического контроля и сигнализации; устройства дистанционного управления.

– Средства защиты от повышенного уровня вибрации: оградительные устройства; виброизолирующие, виброгасящие и вибропоглощающие устройства; устройства автоматического контроля и сигнализации; устройства дистанционного управления.

– Средства защиты от повышенного уровня ультразвука: оградительные устройства; звукоизолирующие, звукопоглощающие устройства; устройства автоматического контроля и сигнализации; устройства дистанционного управления.

– Средства защиты от повышенного уровня инфразвуковых колебаний: оградительные устройства; знаки безопасности.

– Средства защиты от поражения электрическим током: оградительные устройства; устройства автоматического контроля и сигнализации; изолирующие устройства и покрытия; устройства защитного заземления и зануления; устройства автоматического отключения; устройства выравнивания потенциалов и понижения напряжения; устройства дистанционного управления; предохранительные устройства; молниеотводы и разрядники; знаки безопасности.

– Средства защиты от повышенного уровня статического электричества: заземляющие устройства; нейтрализаторы; увлажняющие устройства; антиэлектростатические вещества; экранизирующие устройства.

– Средства защиты от пониженных или повышенных температур поверхностей оборудования, материалов и заготовок: оградительные устройства; устройства автоматического контроля и сигнализации; термо изолирующие устройства; устройства дистанционного управления.

– Средства защиты от повышенных или пониженных температур воздуха, температурных перепадов: оградительные устройства; устройства автоматического контроля и сигнализации; термо изолирующие устройства; устройства дистанционного управления; устройства для обогрева и охлаждения.

– Средства защиты от воздействия механических факторов: оградительные устройства; устройства автоматического контроля и сигнализации; предохранительные устройства; устройства дистанционного управления; тормозные устройства; знаки безопасности.

– Средства защиты от воздействия химических факторов: оградительные устройства; устройства автоматического контроля и сигнализации; герметизирующие устройства; устройства для вентиляции и очистки воздуха; устройства для удаления токсичных веществ; устройства дистанционного управления; знаки безопасности.

– Средства защиты от воздействия биологических факторов: оборудование и препараты для дезинфекции, дезинсекции, стерилизации, дератизации; оградительные устройства; герметизирующие устройства; устройства для вентиляции и очистки воздуха; знаки безопасности.

4.3 Экологические требования к утилизации вычислительной и оргтехники

Инженерно-технические мероприятия, нацеленные на смену устаревших и внедрение новейших технологических процессов и оборудования, содействующих исключению негативных для человека факторов. Многообещающими направлениями тут являются: автоматизация, электро-механизация и дистанционное регулирование производственных процессов, текущих в неблагоприятных для организма условиях труда, сопровождающихся выделением вредоносных веществ.

Инженерно-технические мероприятия, направленные на замену старых и внедрение новых технологических процессов и оборудования, способствующих оздоровлению неблагоприятных условий труда. Необходимыми направлениями здесь являются:

– Автоматизация;

– Механизация;

– Дистанционное управление производственных процессов

Список техники, подлежащей обязательной утилизации ввиду потенциальной опасности, содержится в приказе Министерства природных ресурсов, а также в Постановлении Правительства РФ № 818. В него входят:

- Компьютеры;
- Мониторы;
- Люминесцентные лампы;
- Медицинская техника;
- Бытовые приборы;
- Аккумуляторы;
- Сканеры;
- Ксероксы;
- Принтеры и картриджи для них и др;

Почему неиспользуемая оргтехника и вычислительная техника – это не просто груда ненужного металлолома

Все, что невозможно продать или вторично использовать, являет собой бесполезные вещи, другими словами - мусор. Скопление подобных предметов недопустимо на производстве или предприятии, где абсолютно все существует с целью извлечения прибыли. Если ЭВМ больше неэффективна или окончательно испорчена, работать на ней нельзя, то это означает, что необходимо будет провести замену на его место.

Очевидно, что от бесполезных вещей необходимо эффективно избавляться. Поодобные оргтехнике активы списываются и утилизируются следуя специальным процедурам, которые предусмотренны законами Российской Федерации. Причины регулирования утилизации офисной и вычислительной техники:

Профилактика вреда окружающей среде. Отработанная офисная техника относится к опасным отходам. При производстве компьютеров и других агрегатов применяются вещества, опасные для жизнедеятельности,

например, свинец, мышьяк и др. Обычное выбрасывание техники, особенно регулярное, может нанести непоправимый вред экологии и здоровью.

Сбережение того, что можно сохранить. В состав оргтехники входят детали, содержащие цветной металл, а также определенное количество драгметаллов: золота, серебра, платины. Таким образом, Налоговый кодекс РФ считает даже абсолютно непригодную технику не лишенной определенной ценности. Металл можно извлечь и использовать повторно, кроме того, ценные составляющие необходимо правильно провести по бухгалтерии как часть активов.

Алгоритм для представителей фирм-владельцев старой оргтехники:

– Выбрать компанию для утилизации техники, сертифицированную в Пробирной палате.

– Составить предварительный список техники, подлежащей утилизации.

– Согласовать с фирмой стоимость ее услуг в зависимости от количества единиц техники и предоставляемого сервиса.

– Заключить договор на утилизацию списанных основных технических средств.

– Демонтаж поименованной техники.

– Вывоз демонтированной аппаратуры.

– Подписание акта о выполнении работ и окончательный расчет.

Необходимо проверить правильность оформления и полноту пакета документов об утилизации: об услуге их оформления можно договориться с утилизирующей фирмой дополнительно.

– Утилизация и переработка средств вычислительной техники необходима по трем причинам:

– Вычислительная техника изготавливается с применением металлов, в том числе драгоценных;

– Некоторые компоненты благополучно перерабатываются и применяются в изготовлении повторно. А это – экономия ресурсов;

– Защита окружающей среды.

К средствам вычислительной техники относятся:

– Персональные компьютеры;

– Ноутбуки.

Совместно с ними требуется утилизировать периферийные устройства: сканеры, принтеры, клавиатуры, мыши, агрегаты кормления и т.д. Утилизация компьютеров и оргтехники включает последующие этапы:

– Расчет цены драгметаллов;

– Формирование акта промышленной экспертизы;

– Утилизация;

– Предоставление акта об утилизации для бухгалтерского учета и снятия с баланса.

4.4 Выводы по разделу

Был проведен анализ вредных и опасных факторов на рабочем месте инженера-электроника, было выявлено, что даже на таком, на первый взгляд, безопасном рабочем месте есть столько факторов риска, способных пагубно влиять на состояние здоровья рабочих.

Также были изучены и мероприятия по снижению влияния вредных и опасных факторов на рабочем месте, для нейтрализации или снижения пагубных воздействий на здоровье людей.

В качестве заключения сделаны выводы, что утилизация - необходимая процедура для производства, для предотвращения использования устаревших технологий, изношенности технологического оборудования и заботы об окружающей среде,

ЗАКЛЮЧЕНИЕ

На основании результатов проведенного исследования можно сделать вывод, что цель выпускной квалификационной работы достигнута. Был создан макет, по средствам имеющихся компонентов и материалов, который способен корректно выполнять поставленные задачи. Конечно, интерфейс можно модернизировать и улучшать в зависимости от поставленных задач разработчиком или заказчиком, а именно, менять компоненты, так как используемые в данном исследовании, были больше нацелены на демонстрацию, нежели на какую-либо другую задачу. Была исследована и сконструирована основная платформа, потенциал которой ограничен только фантазиями и возможностями инженеров проектировщиков.

Был проведён анализ предметной области и изучены требования проектируемого устройства, а именно – были получены навыки в понимании процесса сборки, правильном подключении элементов друг к другу, пониманию назначения и возможностей программируемой платформы NodeMCU (назначение контактов, начальный уровень программирования, перекодировка информации, передача ее через Com-порт).

Также были выбраны: элементная база для использования в данном проекте, комплектация принципиальной схемы и печатной платы, спроектированы соединения контактов проводниками на печатной плате, был написан программный код в среде ArduinoIDE на языке Wiring.

Был осуществлён расчет экономических параметров проектируемого устройства, доказана его экономическая эффективность в рамках рынка, подсчитана себестоимость производства проектируемого устройства.

Также были подвергнуты рассмотрению основы охраны труда и охраны экологии, изучены факторы риска, пагубно влияющие на рабочих, рассмотрены методы правильной утилизации офисной техники в связи с

законодательством РФ.

Разработанная система является основой, способной принимать множество трансформаций и модернизаций, различными путями, чтобы удовлетворить современным нуждам заказчика, при этом не обязательно использовать компоненты и программы, представленные в данной работе, так как существует множество вариаций и каждый может выбрать то, что необходимо конкретного для его задачи, при этом основная идея сохраняется.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Разработка электроники. О микроконтроллерах на пальцах [электронный ресурс]. URL: https://habr.com/ru/post/445936/ – (дата обращения 18.02.2022г)

2. Микроконтроллеры. Устройство и особенности. [электронный ресурс]. URL: https://electrosam.ru/glavnaja/slabotochnye-seti/oborudovanie/mikrokontrollery/ – (дата обращения 18.02.2022г)

3. Бегущая строка своими руками [электронный ресурс]. URL: https://alexgyver.ru/gyvermatrixbt/ – (дата обращения 18.02.2022г)

4. [электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Наружная_реклама – (дата обращения 18.02.2022г)

5. [электронный ресурс]. URL: https://jtspcb.com – (дата обращения 18.02.2022г)

6. [электронный ресурс]. URL: https://promdevelop.com/business/outdoor-advertising/ – (дата обращения 18.02.2022г)

7. [электронный ресурс]. URL: https://texterra.ru/blog/partizanskiy-marketing-125-primerov-dlya-vdokhnoveniya.html – (дата обращения 18.02.2022г)

8. [электронный ресурс]. URL: https://diego74.ru/articles/svetodiodnye-reklamnye-begushie-stroki-vidy-oblast-primeneniya-ekspluataciya – (дата обращения 18.02.2022г)

9. [электронный ресурс]. URL: https://alexgyver.ru/gyvermatrixbt/ – (дата обращения 18.02.2022г)

10. Bluetooth матрица на адресных светодиодах [электронный ресурс]. URL: https://alexgyver.ru/matrix_guide/ – (дата обращения 18.02.2022г)

11. Заработок на изготовлении бегущих строк [электронный ресурс]. URL: https://abcbiznes.ru/biznes-idei/634-kak-zarabotat-na-svetodiodnyh-beguschih-strokah.html – (дата обращения 18.02.2022г)

12. Изготовление светодиодных лент на заказ [электронный ресурс]. URL: rosled.com – (дата обращения 18.02.2022г)

13. Ядин, Даниэль – Маркетинговые коммуникации: современная креативная реклама [физический ресурс].

14. Микушин, А. – Занимательно о микроконтроллерах [физический ресурс].

15. Моргун, Олег – Визуальная культура наружной рекламы [физический ресурс].

16. Курушин, В. Д. – Графический дизайн и реклама [физический ресурс].

17. Щепетков. Н. Наружная реклама – Световая реклама в городе [физический ресурс].

18. Колоса. С. Наружная реклама – Технологии наружной рекламы [физический ресурс].

19. Основные порядки БЖД [электронный ресурс]. URL: https://studopedia.ru/9_200601_teoreticheskie-osnovi-i-prakticheskie-funktsii-bzhd.html – (дата обращения 18.02.2022г)

20. Наружная реклама [электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Наружная_реклама – (дата обращения 18.02.2022г)

21. Электронный рынок товаров и услуг от Яндекс, Режим доступа: URL: https://market.yandex.ru

22. Улицкий Г.А. Бегущая строка в светодиодной матрице / Улицкий Г.А. – Вестник коломенского института московского политехнического университета. серия: естественные и технические науки 2016 – 117 с.

23. Сазиков А.В. Цифровая газета (бегущая строка) / Сазиков А.В. – Компьютер и визуальная культура дизайна. Цифровая революция-2017. – 142 с.

24. Волков В. А. Светодиодные автомобильные бегущие строки / Волков В. А. – Полупроводниковая светотехника. 2020. – 46 с.

25. Андреева Л.П., Андреев Г.Я. Информационное табло «бегущая строка» на светодиодных матрицах / Андреева Л.П., Андреев Г.Я – Научные исследования и инновации. 2012. – 103 с.

26. Кретов С.В. Разработка цифрового электронного табло» / Кретов С.В., Инноватика – 2007. – 37 с.

ПРИЛОЖЕНИЕ А – ПРОГРАММНЫЙ КОД

```
// ********************** МАТРИЦА **********************
// если прошивка не лезет в Arduino NANO - отключай режимы! Строка 60 и ниже

#define BRIGHTNESS 150          // стандартная маскимальная яркость (0-255)
#define CURRENT_LIMIT 2000      // лимит по току в миллиамперах, автоматически управляет яркостью (пожалей свой блок питания!) 0 - выключить лимит

#define WIDTH 16                // ширина матрицы
#define HEIGHT 16               // высота матрицы
#define SEGMENTS 1              // диодов в одном "пикселе" (для создания матрицы из кусков ленты)

#define COLOR_ORDER GRB         // порядок цветов на ленте. Если цвет отображается некорректно - меняйте. Начать можно с RGB

#define MATRIX_TYPE 0           // тип матрицы: 0 - зигзаг, 1 - параллельная
#define CONNECTION_ANGLE 0      // угол подключения: 0 - левый нижний, 1 - левый верхний, 2 - правый верхний, 3 - правый нижний
#define STRIP_DIRECTION 0       // направление ленты из угла: 0 - вправо, 1 - вверх, 2 - влево, 3 - вниз
// при неправильной настрйоке матрицы вы получите предупреждение "Wrong matrix parameters! Set to default"

#define MCU_TYPE 0              // микроконтроллер:
//                              0 - AVR (Arduino NANO/MEGA/UNO)
```

```
//                  1 - ESP8266 (NodeMCU, Wemos D1)
//                  2 - STM32 (Blue Pill)

// ******************** ЭФФЕКТЫ И РЕЖИМЫ ********************
#define D_TEXT_SPEED 100          // скорость бегущего текста по умолчанию (мс)
#define D_EFFECT_SPEED 80         // скорость эффектов по умолчанию (мс)
#define D_GAME_SPEED 250          // скорость игр по умолчанию (мс)
#define D_GIF_SPEED 80            // скорость гифок (мс)
#define DEMO_GAME_SPEED 60        // скорость игр в демо режиме (мс)

boolean AUTOPLAY = 1;             // 0 выкл / 1 вкл автоматическую смену режимов (откл. можно со смартфона)
int AUTOPLAY_PERIOD = 10;         // время между авто сменой режимов (секунды)
#define IDLE_TIME 10              // время бездействия кнопок или Bluetooth (в секундах) после которого запускается автосмена режимов и демо в играх

// о поддерживаемых цветах читай тут https://alexgyver.ru/gyvermatrixos-guide/
#define GLOBAL_COLOR_1 CRGB::Green      // основной цвет №1 для игр
#define GLOBAL_COLOR_2 CRGB::Orange     // основной цвет №2 для игр

#define SCORE_SIZE 0              // размер символов счёта в игре. 0 - маленький для 8x8 (шрифт 3x5), 1 - большой (шрифт 5x7)
```

```
#define FONT_TYPE 1              // (0 / 1) два вида маленького шрифта в выводе игрового счёта

#define USE_BUTTONS 1            // использовать физические кнопки управления играми (0 нет, 1 да)
#define BT_MODE 1            // использовать блютус (0 нет, 1 да)
#define USE_NOISE_EFFECTS 1    // крутые полноэкранные эффекты (0 нет, 1 да) СИЛЬНО ЖРУТ ПАМЯТЬ!!!11
#define USE_FONTS 1            // использовать буквы (бегущая строка) (0 нет, 1 да)
#define USE_CLOCK 0          // использовать часы (0 нет, 1 да)

// игры
#define USE_SNAKE 0          // игра змейка (0 нет, 1 да)
#define USE_TETRIS 0         // игра тетрис (0 нет, 1 да)
#define USE_MAZE 0           // игра лабиринт (0 нет, 1 да)
#define USE_RUNNER 0         // игра бегалка-прыгалка (0 нет, 1 да)
#define USE_FLAPPY 0         // игра flappy bird
#define USE_ARKAN 0          // игра арканоид

//          ******************          ПИНЫ          ПОДКЛЮЧЕНИЯ ********************
// Arduino (Nano, Mega)
#if (MCU_TYPE == 0)
#define LED_PIN 6          // пин ленты
#define BUTT_UP 3          // кнопка вверх
#define BUTT_DOWN 5        // кнопка вниз
#define BUTT_LEFT 2        // кнопка влево
#define BUTT_RIGHT 4       // кнопка вправо
#define BUTT_SET 7         // кнопка выбор/игра
```

```
// esp8266
#elif (MCU_TYPE == 1)
#define LED_PIN 2          // пин ленты
#define BUTT_UP 14         // кнопка вверх
#define BUTT_DOWN 13       // кнопка вниз
#define BUTT_LEFT 0        // кнопка влево
#define BUTT_RIGHT 12      // кнопка вправо
#define BUTT_SET 15        // кнопка выбор/игра

// STM32 (BluePill)
#elif (MCU_TYPE == 2)
#define LED_PIN PB12       // пин ленты
#define BUTT_UP PA1        // кнопка вверх
#define BUTT_DOWN PA3      // кнопка вниз
#define BUTT_LEFT PA0      // кнопка влево
#define BUTT_RIGHT PA2     // кнопка вправо
#define BUTT_SET PA4       // кнопка выбор/игра
#endif

//          ********************************          FOR          ENTH
**********************************
#define DEBUG 0
#define NUM_LEDS WIDTH * HEIGHT * SEGMENTS

#define RUNNING_STRING 0
#define CLOCK_MODE 1
#define GAME_MODE 2
#define MADNESS_NOISE 3
#define CLOUD_NOISE 4
#define LAVA_NOISE 5
```

```
#define PLASMA_NOISE 6
#define RAINBOW_NOISE 7
#define RAINBOWSTRIPE_NOISE 8
#define ZEBRA_NOISE 9
#define FOREST_NOISE 10
#define OCEAN_NOISE 11
#define SNOW_ROUTINE 12
#define SPARKLES_ROUTINE 13
#define MATRIX_ROUTINE 14
#define STARFALL_ROUTINE 15
#define BALL_ROUTINE 16
#define BALLS_ROUTINE 17
#define RAINBOW_ROUTINE 18
#define RAINBOWDIAGONAL_ROUTINE 19
#define FIRE_ROUTINE 20
#define IMAGE_MODE 21

#if (MCU_TYPE == 1)
#define FASTLED_INTERRUPT_RETRY_COUNT 0
#define FASTLED_ALLOW_INTERRUPTS 0
#include <ESP8266WiFi.h>
#endif

#include "FastLED.h"
CRGB leds[NUM_LEDS];
String runningText = "";

static const byte maxDim = max(WIDTH, HEIGHT);
byte buttons = 4;   // 0 - верх, 1 - право, 2 - низ, 3 - лево, 4 - не нажата
int globalBrightness = BRIGHTNESS;
```

```
byte globalSpeed = 200;
uint32_t globalColor = 0x00ff00;   // цвет при запуске зелёный
byte breathBrightness;
boolean loadingFlag = true;
byte frameNum;
int gameSpeed = DEMO_GAME_SPEED;
boolean gameDemo = true;
boolean idleState = true;  // флаг холостого режима работы
boolean BTcontrol = false;  // флаг контроля с блютус. Если false - управление с кнопок
int8_t thisMode = 0;
boolean controlFlag = false;
boolean gamemodeFlag = false;
boolean mazeMode = false;
int effects_speed = D_EFFECT_SPEED;
int8_t hrs = 10, mins = 25, secs;
boolean dotFlag;
byte modeCode;     // 0 бегущая, 1 часы, 2 игры, 3 нойс маднесс и далее, 21 гифка или картинка,
boolean fullTextFlag = false;
boolean clockSet = false;

#if (USE_FONTS == 1)
#include "fonts.h"
#endif

uint32_t autoplayTime = ((long)AUTOPLAY_PERIOD * 1000);
uint32_t autoplayTimer;

#include "timerMinim.h"
```

```
timerMinim effectTimer(D_EFFECT_SPEED);
timerMinim gameTimer(DEMO_GAME_SPEED);
timerMinim scrollTimer(D_TEXT_SPEED);
timerMinim idleTimer((long)IDLE_TIME * 1000);
timerMinim changeTimer(70);
timerMinim halfsecTimer(500);

#if (USE_CLOCK == 1 && (MCU_TYPE == 0 || MCU_TYPE == 1))
#include <Wire.h>
#include "RTClib.h"

RTC_DS3231 rtc;
// RTC_DS1307 rtc;
#endif

void setup() {
#if (BT_MODE == 1)
  Serial.begin(9600);
#endif

#if (MCU_TYPE == 1)
  WiFi.setSleepMode(WIFI_NONE_SLEEP);
#endif

#if (USE_CLOCK == 1 && (MCU_TYPE == 0 || MCU_TYPE == 1))
  rtc.begin();
  if (rtc.lostPower()) {
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
  }
  DateTime now = rtc.now();
```

```
  secs = now.second();
  mins = now.minute();
  hrs = now.hour();
#endif

  // настройки ленты
  FastLED.addLeds<WS2812, LED_PIN, COLOR_ORDER>(leds, NUM_LEDS).setCorrection( TypicalLEDStrip );
  FastLED.setBrightness(BRIGHTNESS);
  if (CURRENT_LIMIT > 0) FastLED.setMaxPowerInVoltsAndMilliamps(5, CURRENT_LIMIT);
  FastLED.clear();
  FastLED.show();
  randomSeed(analogRead(0) + analogRead(1));  // пинаем генератор случайных чисел
}

void loop() {
  customRoutine();
  bluetoothRoutine();
}
// вкладка работы с bt & wifi

#if (BT_MODE == 1)
#define PARSE_AMOUNT 4  // максимальное количество значений в массиве, который хотим получить
#define header '$'  // стартовый символ
#define divider ' '  // разделительный символ
#define ending ';'  // завершающий символ
```

```
byte prevY = 0;
byte prevX = 0;
boolean runningFlag;
boolean gameFlag;
boolean effectsFlag;
byte game;
byte effect;
byte intData[PARSE_AMOUNT];     // массив численных значений после парсинга
uint32_t prevColor;
boolean recievedFlag;
byte lastMode = 0;
boolean parseStarted;

void bluetoothRoutine() {
  parsing();                    // принимаем данные

  if (!parseStarted && BTcontrol) {             // на время принятия данных матрицу не обновляем!

    if (runningFlag) fillString(runningText, globalColor);   // бегущая строка
    if (gameFlag) games();                      // игры
    if (effectsFlag) effects();                 // эффекты
  }
}

// блок эффектов, работают по общему таймеру
void effects() {
  if (effectTimer.isReady()) {
    switch (effect) {
```

```
case 0: brightnessRoutine();
  break;
case 1: colorsRoutine();
  break;
case 2: snowRoutine();
  break;
case 3: ballRoutine();
  break;
case 4: rainbowRoutine();
  break;
case 5: rainbowColorsRoutine();
  break;
case 6: fireRoutine();
  break;
case 7: matrixRoutine();
  break;
case 8: ballsRoutine();
  break;
case 9: //wavesRoutine();  // убран из этой версии, т.к. хлам
  break;
case 10: starfallRoutine();
  break;
case 11: sparklesRoutine();
  break;
case 12: rainbowDiagonalRoutine();
  break;
case 13: madnessNoise();
  break;
case 14: cloudNoise();
  break;
```

```
      case 15: lavaNoise();
        break;
      case 16: plasmaNoise();
        break;
      case 17: rainbowNoise();
        break;
      case 18: rainbowStripeNoise();
        break;
      case 19: zebraNoise();
        break;
      case 20: forestNoise();
        break;
      case 21: oceanNoise();
        break;
    }
    FastLED.show();
  }
}

// блок игр
void games() {
  switch (game) {
    case 0:
      snakeRoutine();
      break;
    case 1:
      tetrisRoutine();
      break;
    case 2:
      mazeRoutine();
```

```
      break;
    case 3:
      runnerRoutine();
      break;
    case 4:
      flappyRoutine();
      break;
    case 5:
      arkanoidRoutine();
      break;
  }
}

byte parse_index;
String string_convert = "";
enum modes {NORMAL, COLOR, TEXT} parseMode;


// ********************** ПРИНИМАЕМ ДАННЫЕ **********************
void parsing() {
  // ****************** ОБРАБОТКА ******************
  /*
    Протокол связи, посылка начинается с режима. Режимы:
    0 - отправка цвета $0 colorHEX;
    1 - отправка координат точки $1 X Y;
    2 - заливка - $2;
    3 - очистка - $3;
    4 - яркость - $4 value;
    5 - картинка $5 colorHEX X Y;
```

```
  6 - текст $6 some text
  7 - управление текстом: $7 1; пуск, $7 0; стоп
  8 - эффект
    - $8 0 номерЭффекта;
    - $8 1 старт/стоп;
  9 - игра
  10 - кнопка вверх
  11 - кнопка вправо
  12 - кнопка вниз
  13 - кнопка влево
  14 - пауза в игре
  15 - скорость $8 скорость;
*/
if (recievedFlag) {      // если получены данные
  recievedFlag = false;

  if (intData[0] != 16) {
    idleTimer.reset();
    idleState = false;

    if (!BTcontrol) {
      gameSpeed = globalSpeed * 4;
      gameTimer.setInterval(gameSpeed);
      BTcontrol = true;
    }
  }

  switch (intData[0]) {
    case 1:
      drawPixelXY(intData[1], intData[2], gammaCorrection(globalColor));
```

```
      FastLED.show();
      break;
    case 2:
      fillAll(gammaCorrection(globalColor));
      FastLED.show();
      break;
    case 3:
      FastLED.clear();
      FastLED.show();
      break;
    case 4:
      globalBrightness = intData[1];
      breathBrightness = globalBrightness;
      FastLED.setBrightness(globalBrightness);
      FastLED.show();
      break;
    case 5:
      drawPixelXY(intData[2], intData[3], gammaCorrection(globalColor));
      // делаем обновление матрицы каждую строчку, чтобы реже обновляться
      // и не пропускать пакеты данных (потому что отправка на большую матрицу занимает много времени)
      if (prevY != intData[3] || ( (intData[3] == 0) && (intData[2] == WIDTH - 1) ) ) {
        prevY = intData[3];
        FastLED.show();
      }
      break;
    case 6:
      loadingFlag = true;
```

```
      // строка принимается в переменную runningText
      break;
    case 7:
      if (intData[1] == 1) runningFlag = true;
      if (intData[1] == 0) runningFlag = false;
      break;
    case 8:
      if (intData[1] == 0) {
        effect = intData[2];
        gameFlag = false;
        loadingFlag = true;
        breathBrightness = globalBrightness;
        FastLED.setBrightness(globalBrightness);    // возвращаем яркость
        globalSpeed = intData[3];
        gameTimer.setInterval(globalSpeed * 4);
      }
      else if (intData[1] == 1) effectsFlag = !effectsFlag;
      break;
    case 9:
      if (lastMode != 1) loadingFlag = true;      // начать новую игру при переходе со всех режимов кроме рисования
      effectsFlag = false;
      //gameFlag = true;
      game = intData[1];
      globalSpeed = intData[2];
      gameSpeed = globalSpeed * 4;
      gameTimer.setInterval(gameSpeed);
      break;
    case 10:
      buttons = 0;
```

```
            controlFlag = true;
            break;
          case 11:
            buttons = 1;
            controlFlag = true;
            break;
          case 12:
            buttons = 2;
            controlFlag = true;
            break;
          case 13:
            buttons = 3;
            controlFlag = true;
            break;
          case 14:
            gameFlag = !gameFlag;
            break;
          case 15: globalSpeed = intData[1];
            if (gameFlag) {
              gameSpeed = globalSpeed * 4;          // для игр скорость нужно
меньше!
              gameTimer.setInterval(gameSpeed);
            }
            if (effectsFlag) effectTimer.setInterval(globalSpeed);
            if (runningFlag) scrollTimer.setInterval(globalSpeed);
            break;
          case 16:
            if (intData[1] == 0) AUTOPLAY = true;
            else if (intData[1] == 1) AUTOPLAY = false;
            else if (intData[1] == 2) prevMode();
```

```
      else if (intData[1] == 3) nextMode();
      break;
     case 17: autoplayTime = ((long)intData[1] * 1000);
      autoplayTimer = millis();
      break;
    }
    lastMode = intData[0];  // запомнить предыдущий режим
   }

   // ****************** ПАРСИНГ ****************
   if (Serial.available() > 0) {
    char incomingByte;
    if (parseMode == TEXT) {     // если нужно принять строку
     runningText = Serial.readString();  // принимаем всю
     incomingByte = ending;             // сразу завершаем парс
     parseMode = NORMAL;
    } else {
     incomingByte = Serial.read();        // обязательно ЧИТАЕМ входящий символ
    }
    if (parseStarted) {                   // если приняли начальный символ (парсинг разрешён)
     if (incomingByte != divider && incomingByte != ending) {   // если это не пробел И не конец
      string_convert += incomingByte;       // складываем в строку
     } else {                        // если это пробел или ; конец пакета
      if (parse_index == 0) {
       byte thisMode = string_convert.toInt();
       if (thisMode == 0 || thisMode == 5) parseMode = COLOR;       // передача цвета (в отдельную переменную)
```

```
            else if (thisMode == 6) parseMode = TEXT;
            else parseMode = NORMAL;
            //if (thisMode != 7 || thisMode != 0) runningFlag = false;
          }

          if (parse_index == 1) {       // для второго (с нуля) символа в посылке
            if (parseMode == NORMAL) intData[parse_index] = string_convert.toInt();           // преобразуем строку в int и кладём в массив}
            //if (parseMode == COLOR) globalColor = strtol(&string_convert[0], NULL, 16);     // преобразуем строку HEX в цифру
            if (parseMode == COLOR) globalColor = (uint32_t)HEXtoInt(string_convert);     // преобразуем строку HEX в цифру
          } else {
            intData[parse_index] = string_convert.toInt();  // преобразуем строку в int и кладём в массив
          }
          string_convert = "";              // очищаем строку
          parse_index++;                                  // переходим к парсингу следующего элемента массива
        }
      }
      if (incomingByte == header) {           // если это $
        parseStarted = true;                    // поднимаем флаг, что можно парсить
        parse_index = 0;                      // сбрасываем индекс
        string_convert = "";                // очищаем строку
      }
      if (incomingByte == ending) {             // если таки приняли ; - конец парсинга
        parseMode == NORMAL;
```

```
      parseStarted = false;                  // сброс
      recievedFlag = true;                   // флаг на принятие
     }
    }
   }

   // hex string to uint32_t
   uint32_t HEXtoInt(String hexValue) {
    byte tens, ones, number1, number2, number3;
    tens = (hexValue[0] < '9') ? hexValue[0] - '0' : hexValue[0] - '7';
    ones = (hexValue[1] < '9') ? hexValue[1] - '0' : hexValue[1] - '7';
    number1 = (16 * tens) + ones;

    tens = (hexValue[2] < '9') ? hexValue[2] - '0' : hexValue[2] - '7';
    ones = (hexValue[3] < '9') ? hexValue[3] - '0' : hexValue[3] - '7';
    number2 = (16 * tens) + ones;

    tens = (hexValue[4] < '9') ? hexValue[4] - '0' : hexValue[4] - '7';
    ones = (hexValue[5] < '9') ? hexValue[5] - '0' : hexValue[5] - '7';
    number3 = (16 * tens) + ones;

    return ((uint32_t)number1 << 16 | (uint32_t)number2 << 8 | number3 <<
0);
   }

   #elif (BT_MODE == 0)
   void bluetoothRoutine() {
    return;
   }
   #endif
```

```
// эффекты

// *************** НАСТРОЙКИ ЭФФЕКТОВ ***************
// эффект "синусоиды" - ОТКЛЮЧЕН
#define WAVES_AMOUNT 2    // количество синусоид

// эффект "шарики"
#define BALLS_AMOUNT 3    // количество "шариков"
#define CLEAR_PATH 1      // очищать путь
#define BALL_TRACK 1      // (0 / 1) - вкл/выкл следы шариков
#define DRAW_WALLS 0      // режим с рисованием препятствий для шаров (не работает на ESP и STM32)
#define TRACK_STEP 70     // длина хвоста шарика (чем больше цифра, тем хвост короче)

// эффект "квадратик"
#define BALL_SIZE 3       // размер шара
#define RANDOM_COLOR 1    // случайный цвет при отскоке

// эффект "огонь"
#define SPARKLES 1        // вылетающие угольки вкл выкл
#define HUE_ADD 0         // добавка цвета в огонь (от 0 до 230) - меняет весь цвет пламени

// эффект "кометы"
#define TAIL_STEP 100     // длина хвоста кометы
#define SATURATION 150    // насыщенность кометы (от 0 до 255)
#define STAR_DENSE 60     // количество (шанс появления) комет

// эффект "конфетти"
```

```
#define DENSE 3           // плотность конфетти
#define BRIGHT_STEP 70    // шаг уменьшения яркости

// эффект "снег"
#define SNOW_DENSE 10     // плотность снегопада

// --------------------- ДЛЯ РАЗРАБОТЧИКОВ ---------------------

// ********** "дыхание" яркостью **********
boolean brightnessDirection;
void brightnessRoutine() {
  if (brightnessDirection) {
    breathBrightness += 2;
    if (breathBrightness > globalBrightness - 1) {
      brightnessDirection = false;
    }
  } else {
    breathBrightness -= 2;
    if (breathBrightness < 1) {
      brightnessDirection = true;
    }
  }
  FastLED.setBrightness(breathBrightness);
}

// ********** смена цвета активных светодиодов (рисунка) **********
byte hue;
void colorsRoutine() {
  hue += 4;
```

```
  for (int i = 0; i < NUM_LEDS; i++) {
    if (getPixColor(i) > 0) leds[i] = CHSV(hue, 255, 255);
  }
 }

 // ********** снегопад 2.0 **********
 void snowRoutine() {
  modeCode = 12;
  // сдвигаем всё вниз
  for (byte x = 0; x < WIDTH; x++) {
   for (byte y = 0; y < HEIGHT - 1; y++) {
    drawPixelXY(x, y, getPixColorXY(x, y + 1));
   }
  }

  for (byte x = 0; x < WIDTH; x++) {
   // заполняем случайно верхнюю строку
   // а также не даём двум блокам по вертикали вместе быть
   if (getPixColorXY(x, HEIGHT - 2) == 0 && (random(0, SNOW_DENSE) == 0))
    drawPixelXY(x, HEIGHT - 1, 0xE0FFFF - 0x101010 * random(0, 4));
   else
    drawPixelXY(x, HEIGHT - 1, 0x000000);
  }
 }

 // ***************************** БЛУДНЫЙ КУБИК *****************************
 int coordB[2];
 int8_t vectorB[2];
```

```
CRGB ballColor;

void ballRoutine() {
  if (loadingFlag) {
    for (byte i = 0; i < 2; i++) {
      coordB[i] = WIDTH / 2 * 10;
      vectorB[i] = random(8, 20);
      ballColor = CHSV(random(0, 9) * 28, 255, 255);
    }
    modeCode = 16;
    loadingFlag = false;
  }
  for (byte i = 0; i < 2; i++) {
    coordB[i] += vectorB[i];
    if (coordB[i] < 0) {
      coordB[i] = 0;
      vectorB[i] = -vectorB[i];
      if (RANDOM_COLOR) ballColor = CHSV(random(0, 9) * 28, 255, 255);
      //vectorB[i] += random(0, 6) - 3;
    }
  }
  if (coordB[0] > (WIDTH - BALL_SIZE) * 10) {
    coordB[0] = (WIDTH - BALL_SIZE) * 10;
    vectorB[0] = -vectorB[0];
    if (RANDOM_COLOR) ballColor = CHSV(random(0, 9) * 28, 255, 255);
    //vectorB[0] += random(0, 6) - 3;
  }
  if (coordB[1] > (HEIGHT - BALL_SIZE) * 10) {
    coordB[1] = (HEIGHT - BALL_SIZE) * 10;
```

```
    vectorB[1] = -vectorB[1];
    if (RANDOM_COLOR) ballColor = CHSV(random(0, 9) * 28, 255, 255);
    //vectorB[1] += random(0, 6) - 3;
  }
  FastLED.clear();
  for (byte i = 0; i < BALL_SIZE; i++)
    for (byte j = 0; j < BALL_SIZE; j++)
      leds[getPixelNumber(coordB[0] / 10 + i, coordB[1] / 10 + j)] = ballColor;
}

// ********** радуга заливка **********
void rainbowRoutine() {
  modeCode = 18;
  hue += 3;
  for (byte i = 0; i < WIDTH; i++) {
    CHSV thisColor = CHSV((byte)(hue + i * float(255 / WIDTH)), 255, 255);
    for (byte j = 0; j < HEIGHT; j++)
      drawPixelXY(i, j, thisColor);   //leds[getPixelNumber(i, j)] = thisColor;
  }
}

// ********** радуга дигональная **********
void rainbowDiagonalRoutine() {
  modeCode = 19;
  hue += 3;
  for (byte x = 0; x < WIDTH; x++) {
    for (byte y = 0; y < HEIGHT; y++) {
```

```
            CHSV thisColor = CHSV((byte)(hue + (float)(WIDTH / HEIGHT * x +
y) * (float)(255 / maxDim)), 255, 255);
            drawPixelXY(x, y, thisColor); //leds[getPixelNumber(i, j)] = thisColor;
          }
        }
      }


      // ********** радуга активных светодиодов (рисунка) **********
      void rainbowColorsRoutine() {
        hue++;
        for (byte i = 0; i < WIDTH; i++) {
          CHSV thisColor = CHSV((byte)(hue + i * float(255 / WIDTH)), 255,
255);
          for (byte j = 0; j < HEIGHT; j++)
            if (getPixColor(getPixelNumber(i, j)) > 0) drawPixelXY(i, j, thisColor);
        }
      }


      // ********************** огонь **********************
      unsigned char matrixValue[8][16];
      unsigned char line[WIDTH];
      int pcnt = 0;

      //these values are substracetd from the generated values to give a shape to
the animation
      const unsigned char valueMask[8][16] PROGMEM = {
        {32 , 0  , 0  , 0  , 0  , 0  , 0  , 32 , 32 , 0  , 0  , 0  , 0  , 0  , 0  , 32 },
        {64 , 0  , 0  , 0  , 0  , 0  , 0  , 64 , 64 , 0  , 0  , 0  , 0  , 0  , 0  , 64 },
```

```
  {96 , 32 , 0  , 0  , 0  , 0  , 32 , 96 , 96 , 32 , 0  , 0  , 0  , 0  , 32 , 96 },
  {128, 64 , 32 , 0  , 0  , 32 , 64 , 128, 128, 64 , 32 , 0  , 0  , 32 , 64 , 128},
  {160, 96 , 64 , 32 , 32 , 64 , 96 , 160, 160, 96 , 64 , 32 , 32 , 64 , 96 , 160},
  {192, 128, 96 , 64 , 64 , 96 , 128, 192, 192, 128, 96 , 64 , 64 , 96 , 128, 192},
  {255, 160, 128, 96 , 96 , 128, 160, 255, 255, 160, 128, 96 , 96 , 128, 160, 255},
  {255, 192, 160, 128, 128, 160, 192, 255, 255, 192, 160, 128, 128, 160, 192, 255}
};

//these are the hues for the fire,
//should be between 0 (red) to about 25 (yellow)
const unsigned char hueMask[8][16] PROGMEM = {
  {1 , 11, 19, 25, 25, 22, 11, 1 , 1 , 11, 19, 25, 25, 22, 11, 1 },
  {1 , 8 , 13, 19, 25, 19, 8 , 1 , 1 , 8 , 13, 19, 25, 19, 8 , 1 },
  {1 , 8 , 13, 16, 19, 16, 8 , 1 , 1 , 8 , 13, 16, 19, 16, 8 , 1 },
  {1 , 5 , 11, 13, 13, 13, 5 , 1 , 1 , 5 , 11, 13, 13, 13, 5 , 1 },
  {1 , 5 , 11, 11, 11, 11, 5 , 1 , 1 , 5 , 11, 11, 11, 11, 5 , 1 },
  {0 , 1 , 5 , 8 , 8 , 5 , 1 , 0 , 0 , 1 , 5 , 8 , 8 , 5 , 1 , 0 },
  {0 , 0 , 1 , 5 , 5 , 1 , 0 , 0 , 0 , 0 , 1 , 5 , 5 , 1 , 0 , 0 },
  {0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 }
};

void fireRoutine() {
  if (loadingFlag) {
    modeCode = 20;
    loadingFlag = false;
    FastLED.clear();
    generateLine();
```

```
    memset(matrixValue, 0, sizeof(matrixValue));
  }
  if (pcnt >= 100) {
    shiftUp();
    generateLine();
    pcnt = 0;
  }
  drawFrame(pcnt);
  pcnt += 30;
}


// Randomly generate the next line (matrix row)

void generateLine() {
  for (uint8_t x = 0; x < WIDTH; x++) {
    line[x] = random(64, 255);
  }
}

//shift all values in the matrix up one row

void shiftUp() {
  for (uint8_t y = HEIGHT - 1; y > 0; y--) {
    for (uint8_t x = 0; x < WIDTH; x++) {
      uint8_t newX = x;
      if (x > 15) newX = x - 15;
      if (y > 7) continue;
      matrixValue[y][newX] = matrixValue[y - 1][newX];
    }
```

```
  }

  for (uint8_t x = 0; x < WIDTH; x++) {
    uint8_t newX = x;
    if (x > 15) newX = x - 15;
    matrixValue[0][newX] = line[newX];
  }
}

// draw a frame, interpolating between 2 "key frames"
// @param pcnt percentage of interpolation

void drawFrame(int pcnt) {
  int nextv;

  //each row interpolates with the one before it
  for (unsigned char y = HEIGHT - 1; y > 0; y--) {
    for (unsigned char x = 0; x < WIDTH; x++) {
      uint8_t newX = x;
      if (x > 15) newX = x - 15;
      if (y < 8) {
        nextv =
          (((100.0 - pcnt) * matrixValue[y][newX]
            + pcnt * matrixValue[y - 1][newX]) / 100.0)
          - pgm_read_byte(&(valueMask[y][newX]));

        CRGB color = CHSV(
                       HUE_ADD + pgm_read_byte(&(hueMask[y][newX])), // H
                       255, // S
                       (uint8_t)max(0, nextv) // V
```

```
                    );

                leds[getPixelNumber(x, y)] = color;
              } else if (y == 8 && SPARKLES) {
                if (random(0, 20) == 0 && getPixColorXY(x, y - 1) != 0) drawPixelXY(x, y, getPixColorXY(x, y - 1));
                else drawPixelXY(x, y, 0);
              } else if (SPARKLES) {

                // старая версия для яркости
                if (getPixColorXY(x, y - 1) > 0)
                  drawPixelXY(x, y, getPixColorXY(x, y - 1));
                else drawPixelXY(x, y, 0);

              }
            }
          }

          //first row interpolates with the "next" line
          for (unsigned char x = 0; x < WIDTH; x++) {
            uint8_t newX = x;
            if (x > 15) newX = x - 15;
            CRGB color = CHSV(
                      HUE_ADD + pgm_read_byte(&(hueMask[0][newX])), // H
                      255,          // S
                      (uint8_t)(((100.0 - pcnt) * matrixValue[0][newX] + pcnt * line[newX]) / 100.0) // V
                    );
            leds[getPixelNumber(newX, 0)] = color;
          }
```

```
}

// *************** МАТРИЦА *****************
void matrixRoutine() {
  if (loadingFlag) {
    loadingFlag = false;
    modeCode = 14;
    FastLED.clear();
  }
  for (byte x = 0; x < WIDTH; x++) {
    // заполняем случайно верхнюю строку
    uint32_t thisColor = getPixColorXY(x, HEIGHT - 1);
    if (thisColor == 0)
      drawPixelXY(x, HEIGHT - 1, 0x00FF00 * (random(0, 10) == 0));
    else if (thisColor < 0x002000)
      drawPixelXY(x, HEIGHT - 1, 0);
    else
      drawPixelXY(x, HEIGHT - 1, thisColor - 0x002000);
  }

  // сдвигаем всё вниз
  for (byte x = 0; x < WIDTH; x++) {
    for (byte y = 0; y < HEIGHT - 1; y++) {
      drawPixelXY(x, y, getPixColorXY(x, y + 1));
    }
  }
}
```

```
//                ********************************                ШАРИКИ
**********************************
        int coord[BALLS_AMOUNT][2];
        int8_t vector[BALLS_AMOUNT][2];
        CRGB ballColors[BALLS_AMOUNT];

        void ballsRoutine() {
          if (loadingFlag) {
            modeCode = 17;
            loadingFlag = false;
            for (byte j = 0; j < BALLS_AMOUNT; j++) {
              int sign;

              // забиваем случайными данными
              coord[j][0] = WIDTH / 2 * 10;
              random(0, 2) ? sign = 1 : sign = -1;
              vector[j][0] = random(4, 15) * sign;
              coord[j][1] = HEIGHT / 2 * 10;
              random(0, 2) ? sign = 1 : sign = -1;
              vector[j][1] = random(4, 15) * sign;
              ballColors[j] = CHSV(random(0, 9) * 28, 255, 255);
            }
          }


          if (!BALL_TRACK)    // если режим БЕЗ следов шариков
            FastLED.clear();  // очистить
          else {              // режим со следами
            fader(TRACK_STEP);
          }
```

```
// движение шариков
for (byte j = 0; j < BALLS_AMOUNT; j++) {

 // отскок от нарисованных препятствий
 if (DRAW_WALLS) {
  uint32_t thisColor = getPixColorXY(coord[j][0] / 10 + 1, coord[j][1] /
10);
  if (thisColor == globalColor/* && vector[j][0] > 0*/) {
   vector[j][0] = -vector[j][0];
  }
  thisColor = getPixColorXY(coord[j][0] / 10 - 1, coord[j][1] / 10);
  if (thisColor == globalColor/* && vector[j][0] < 0*/) {
   vector[j][0] = -vector[j][0];
  }
  thisColor = getPixColorXY(coord[j][0] / 10, coord[j][1] / 10 + 1);
  if (thisColor == globalColor/* && vector[j][1] > 0*/) {
   vector[j][1] = -vector[j][1];
  }
  thisColor = getPixColorXY(coord[j][0] / 10, coord[j][1] / 10 - 1);
  if (thisColor == globalColor/* && vector[j][1] < 0*/) {
   vector[j][1] = -vector[j][1];
  }
 }

 // движение шариков
 for (byte i = 0; i < 2; i++) {
  coord[j][i] += vector[j][i];
  if (coord[j][i] < 0) {
   coord[j][i] = 0;
```

```
      vector[j][i] = -vector[j][i];
     }
    }

    if (coord[j][0] > (WIDTH - 1) * 10) {
     coord[j][0] = (WIDTH - 1) * 10;
     vector[j][0] = -vector[j][0];
    }
    if (coord[j][1] > (HEIGHT - 1) * 10) {
     coord[j][1] = (HEIGHT - 1) * 10;
     vector[j][1] = -vector[j][1];
    }
    leds[getPixelNumber(coord[j][0] / 10, coord[j][1] / 10)] =  ballColors[j];
   }
  }


  /*
   //          ********************************          СИНУСОИДЫ
**********************************
   #define DEG_TO_RAD 0.017453
   int t;
   byte w[WAVES_AMOUNT];
   byte phi[WAVES_AMOUNT];
   byte A[WAVES_AMOUNT];
   CRGB waveColors[WAVES_AMOUNT];

   void wavesRoutine() {
   if (loadingFlag) {
    loadingFlag = false;
```

```
    for (byte j = 0; j < WAVES_AMOUNT; j++) {
      // забиваем случайными данными
      w[j] = random(17, 25);
      phi[j] = random(0, 360);
      A[j] = HEIGHT / 2 * random(4, 11) / 10;
      waveColors[j] = CHSV(random(0, 9) * 28, 255, 255);
    }
  }
  if (effectTimer.isReady()) {

    // сдвигаем все пиксели вправо
    for (int i = WIDTH - 1; i > 0; i--)
      for (int j = 0; j < HEIGHT; j++)
        drawPixelXY(i, j, getPixColorXY(i - 1, j));

    // увеличиваем "угол"
    t++;
    if (t > 360) t = 0;

    // заливаем чёрным левую линию
    for (byte i = 0; i < HEIGHT; i++) {
      drawPixelXY(0, i, 0x000000);
    }

    // генерируем позицию точки через синус
    for (byte j = 0; j < WAVES_AMOUNT; j++) {
      float value = HEIGHT / 2 + (float)A[j] * sin((float)w[j] * t *
DEG_TO_RAD + (float)phi[j] * DEG_TO_RAD);
      leds[getPixelNumber(0, (byte)value)] = waveColors[j];
    }
```

```
  }
  }
*/

// функция плавного угасания цвета для всех пикселей
void fader(byte step) {
  for (byte i = 0; i < WIDTH; i++) {
    for (byte j = 0; j < HEIGHT; j++) {
      fadePixel(i, j, step);
    }
  }
}
void fadePixel(byte i, byte j, byte step) {     // новый фейдер
  int pixelNum = getPixelNumber(i, j);
  if (getPixColor(pixelNum) == 0) return;

  if (leds[pixelNum].r >= 30 ||
      leds[pixelNum].g >= 30 ||
      leds[pixelNum].b >= 30) {
    leds[pixelNum].fadeToBlackBy(step);
  } else {
    leds[pixelNum] = 0;
  }
}

/*
  void fadePixel(byte i, byte j, byte step) {     // старый фейдер
  // измеряяем цвет текущего пикселя
  uint32_t thisColor = getPixColorXY(i, j);
```

```
  // если 0, то пропускаем действия и переходим к следующему
  if (thisColor == 0) return;

  // разбиваем цвет на составляющие RGB
  byte rgb[3];
  rgb[0] = (thisColor >> 16) & 0xff;
  rgb[1] = (thisColor >> 8) & 0xff;
  rgb[2] = thisColor & 0xff;

  // ищем максимум
  byte maximum = max(max(rgb[0], rgb[1]), rgb[2]);
  float coef = 0;

  // если есть возможность уменьшить
  if (maximum >= step)
    // уменьшаем и находим коэффициент уменьшения
    coef = (float)(maximum - step) / maximum;

  // далее все цвета умножаем на этот коэффициент
  for (byte i = 0; i < 3; i++) {
    if (rgb[i] > 0) rgb[i] = (float)rgb[i] * coef;
    else rgb[i] = 0;
  }
  leds[getPixelNumber(i, j)] = CRGB(rgb[0], rgb[1], rgb[2]);
  }
*/

// ********************* ЗВЕЗДОПАД ******************
void starfallRoutine() {
  modeCode = 15;
```

```
  // заполняем головами комет левую и верхнюю линию
  for (byte i = HEIGHT / 2; i < HEIGHT; i++) {
   if (getPixColorXY(0, i) == 0
      && (random(0, STAR_DENSE) == 0)
      && getPixColorXY(0, i + 1) == 0
      && getPixColorXY(0, i - 1) == 0)
     leds[getPixelNumber(0, i)] = CHSV(random(0, 200), SATURATION,
255);
  }
  for (byte i = 0; i < WIDTH / 2; i++) {
   if (getPixColorXY(i, HEIGHT - 1) == 0
      && (random(0, STAR_DENSE) == 0)
      && getPixColorXY(i + 1, HEIGHT - 1) == 0
      && getPixColorXY(i - 1, HEIGHT - 1) == 0)
     leds[getPixelNumber(i, HEIGHT - 1)] = CHSV(random(0, 200),
SATURATION, 255);
  }

  // сдвигаем по диагонали
  for (byte y = 0; y < HEIGHT - 1; y++) {
   for (byte x = WIDTH - 1; x > 0; x--) {
    drawPixelXY(x, y, getPixColorXY(x - 1, y + 1));
   }
  }

  // уменьшаем яркость левой и верхней линии, формируем "хвосты"
  for (byte i = HEIGHT / 2; i < HEIGHT; i++) {
   fadePixel(0, i, TAIL_STEP);
  }
  for (byte i = 0; i < WIDTH / 2; i++) {
```

```
    fadePixel(i, HEIGHT - 1, TAIL_STEP);
  }
}

// рандомные гаснущие вспышки
void sparklesRoutine() {
  modeCode = 13;
  for (byte i = 0; i < DENSE; i++) {
    byte x = random(0, WIDTH);
    byte y = random(0, HEIGHT);
    if (getPixColorXY(x, y) == 0)
      leds[getPixelNumber(x, y)] = CHSV(random(0, 255), 255, 255);
  }
  fader(BRIGHT_STEP);
}
// шрифты для вывода текста
const uint8_t fontHEX[][5] PROGMEM = {
  {0x00, 0x00, 0x00, 0x00, 0x00}, //   0x20 32
  {0x00, 0x00, 0x6f, 0x00, 0x00}, // ! 0x21 33
  {0x00, 0x07, 0x00, 0x07, 0x00}, // " 0x22 34
  {0x14, 0x7f, 0x14, 0x7f, 0x14}, // # 0x23 35
  {0x00, 0x07, 0x04, 0x1e, 0x00}, // $ 0x24 36
  {0x23, 0x13, 0x08, 0x64, 0x62}, // % 0x25 37
  {0x36, 0x49, 0x56, 0x20, 0x50}, // & 0x26 38
  {0x00, 0x00, 0x07, 0x00, 0x00}, // ' 0x27 39
  {0x00, 0x1c, 0x22, 0x41, 0x00}, // ( 0x28 40
  {0x00, 0x41, 0x22, 0x1c, 0x00}, // ) 0x29 41
  {0x14, 0x08, 0x3e, 0x08, 0x14}, // * 0x2a 42
  {0x08, 0x08, 0x3e, 0x08, 0x08}, // + 0x2b 43
  {0x00, 0x50, 0x30, 0x00, 0x00}, // , 0x2c 44
```

{0x08, 0x08, 0x08, 0x08, 0x08}, // - 0x2d 45
{0x00, 0x60, 0x60, 0x00, 0x00}, // . 0x2e 46
{0x20, 0x10, 0x08, 0x04, 0x02}, // / 0x2f 47
{0x3e, 0x51, 0x49, 0x45, 0x3e}, // 0 0x30 48
{0x00, 0x42, 0x7f, 0x40, 0x00}, // 1 0x31 49
{0x42, 0x61, 0x51, 0x49, 0x46}, // 2 0x32 50
{0x21, 0x41, 0x45, 0x4b, 0x31}, // 3 0x33 51
{0x18, 0x14, 0x12, 0x7f, 0x10}, // 4 0x34 52
{0x27, 0x45, 0x45, 0x45, 0x39}, // 5 0x35 53
{0x3c, 0x4a, 0x49, 0x49, 0x30}, // 6 0x36 54
{0x01, 0x71, 0x09, 0x05, 0x03}, // 7 0x37 55
{0x36, 0x49, 0x49, 0x49, 0x36}, // 8 0x38 56
{0x06, 0x49, 0x49, 0x29, 0x1e}, // 9 0x39 57
{0x00, 0x36, 0x36, 0x00, 0x00}, // : 0x3a 58
{0x00, 0x56, 0x36, 0x00, 0x00}, // ; 0x3b 59
{0x08, 0x14, 0x22, 0x41, 0x00}, // < 0x3c 60
{0x14, 0x14, 0x14, 0x14, 0x14}, // = 0x3d 61
{0x00, 0x41, 0x22, 0x14, 0x08}, // > 0x3e 62
{0x02, 0x01, 0x51, 0x09, 0x06}, // ? 0x3f 63
{0x3e, 0x41, 0x5d, 0x49, 0x4e}, // @ 0x40 64
{0x7e, 0x09, 0x09, 0x09, 0x7e}, // A 0x41 65
{0x7f, 0x49, 0x49, 0x49, 0x36}, // B 0x42 66
{0x3e, 0x41, 0x41, 0x41, 0x22}, // C 0x43 67
{0x7f, 0x41, 0x41, 0x41, 0x3e}, // D 0x44 68
{0x7f, 0x49, 0x49, 0x49, 0x41}, // E 0x45 69
{0x7f, 0x09, 0x09, 0x09, 0x01}, // F 0x46 70
{0x3e, 0x41, 0x49, 0x49, 0x7a}, // G 0x47 71
{0x7f, 0x08, 0x08, 0x08, 0x7f}, // H 0x48 72
{0x00, 0x41, 0x7f, 0x41, 0x00}, // I 0x49 73
{0x20, 0x40, 0x41, 0x3f, 0x01}, // J 0x4a 74

{0x7f, 0x08, 0x14, 0x22, 0x41}, // K 0x4b 75
{0x7f, 0x40, 0x40, 0x40, 0x40}, // L 0x4c 76
{0x7f, 0x02, 0x0c, 0x02, 0x7f}, // M 0x4d 77
{0x7f, 0x04, 0x08, 0x10, 0x7f}, // N 0x4e 78
{0x3e, 0x41, 0x41, 0x41, 0x3e}, // O 0x4f 79
{0x7f, 0x09, 0x09, 0x09, 0x06}, // P 0x50 80
{0x3e, 0x41, 0x51, 0x21, 0x5e}, // Q 0x51 81
{0x7f, 0x09, 0x19, 0x29, 0x46}, // R 0x52 82
{0x46, 0x49, 0x49, 0x49, 0x31}, // S 0x53 83
{0x01, 0x01, 0x7f, 0x01, 0x01}, // T 0x54 84
{0x3f, 0x40, 0x40, 0x40, 0x3f}, // U 0x55 85
{0x0f, 0x30, 0x40, 0x30, 0x0f}, // V 0x56 86
{0x3f, 0x40, 0x30, 0x40, 0x3f}, // W 0x57 87
{0x63, 0x14, 0x08, 0x14, 0x63}, // X 0x58 88
{0x07, 0x08, 0x70, 0x08, 0x07}, // Y 0x59 89
{0x61, 0x51, 0x49, 0x45, 0x43}, // Z 0x5a 90
{0x3c, 0x4a, 0x49, 0x29, 0x1e}, // [0x5b 91
{0x02, 0x04, 0x08, 0x10, 0x20}, // \ 0x5c 92
{0x00, 0x41, 0x7f, 0x00, 0x00}, //] 0x5d 93
{0x04, 0x02, 0x01, 0x02, 0x04}, // ^ 0x5e 94
{0x40, 0x40, 0x40, 0x40, 0x40}, // _ 0x5f 95
{0x00, 0x00, 0x03, 0x04, 0x00}, // ` 0x60 96
{0x20, 0x54, 0x54, 0x54, 0x78}, // a 0x61 97
{0x7f, 0x48, 0x44, 0x44, 0x38}, // b 0x62 98
{0x38, 0x44, 0x44, 0x44, 0x20}, // c 0x63 99
{0x38, 0x44, 0x44, 0x48, 0x7f}, // d 0x64 100
{0x38, 0x54, 0x54, 0x54, 0x18}, // e 0x65 101
{0x08, 0x7e, 0x09, 0x01, 0x02}, // f 0x66 102
{0x0c, 0x52, 0x52, 0x52, 0x3e}, // g 0x67 103
{0x7f, 0x08, 0x04, 0x04, 0x78}, // h 0x68 104

{0x00, 0x44, 0x7d, 0x40, 0x00}, // i 0x69 105
{0x20, 0x40, 0x44, 0x3d, 0x00}, // j 0x6a 106
{0x00, 0x7f, 0x10, 0x28, 0x44}, // k 0x6b 107
{0x00, 0x41, 0x7f, 0x40, 0x00}, // l 0x6c 108
{0x7c, 0x04, 0x18, 0x04, 0x78}, // m 0x6d 109
{0x7c, 0x08, 0x04, 0x04, 0x78}, // n 0x6e 110
{0x38, 0x44, 0x44, 0x44, 0x38}, // o 0x6f 111
{0x7c, 0x14, 0x14, 0x14, 0x08}, // p 0x70 112
{0x08, 0x14, 0x14, 0x18, 0x7c}, // q 0x71 113
{0x7c, 0x08, 0x04, 0x04, 0x08}, // r 0x72 114
{0x48, 0x54, 0x54, 0x54, 0x20}, // s 0x73 115
{0x04, 0x3f, 0x44, 0x40, 0x20}, // t 0x74 116
{0x3c, 0x40, 0x40, 0x20, 0x7c}, // u 0x75 117
{0x1c, 0x20, 0x40, 0x20, 0x1c}, // v 0x76 118
{0x3c, 0x40, 0x30, 0x40, 0x3c}, // w 0x77 119
{0x44, 0x28, 0x10, 0x28, 0x44}, // x 0x78 120
{0x0c, 0x50, 0x50, 0x50, 0x3c}, // y 0x79 121
{0x44, 0x64, 0x54, 0x4c, 0x44}, // z 0x7a 122
{0x00, 0x08, 0x36, 0x41, 0x41}, // { 0x7b 123
{0x00, 0x00, 0x7f, 0x00, 0x00}, // | 0x7c 124
{0x41, 0x41, 0x36, 0x08, 0x00}, // } 0x7d 125
{0x04, 0x02, 0x04, 0x08, 0x04}, // ~ 0x7e 126

{0x7e, 0x09, 0x09, 0x09, 0x7e}, // А 192
{0x7F, 0x49, 0x49, 0x49, 0x71}, // Б
{0x7f, 0x49, 0x49, 0x49, 0x36}, // В
{0x7F, 0x01, 0x01, 0x01, 0x01}, // Г
{0x60, 0x3E, 0x21, 0x3F, 0x60}, // Д
{0x7f, 0x49, 0x49, 0x49, 0x41}, // Е
{0x76, 0x08, 0x7F, 0x08, 0x76}, // Ж

{0x21, 0x41, 0x45, 0x4b, 0x31}, // З
{0x7F, 0x20, 0x10, 0x08, 0x7F}, // И
{0x7E, 0x20, 0x11, 0x08, 0x7E}, // Й
{0x7f, 0x08, 0x14, 0x22, 0x41}, // К
{0x70, 0x0E, 0x01, 0x01, 0x7F}, // Л
{0x7f, 0x02, 0x0c, 0x02, 0x7f}, // М
{0x7f, 0x08, 0x08, 0x08, 0x7f}, // Н
{0x3e, 0x41, 0x41, 0x41, 0x3e}, // О
{0x7F, 0x01, 0x01, 0x01, 0x7F}, // П
{0x7f, 0x09, 0x09, 0x09, 0x06}, // Р
{0x3e, 0x41, 0x41, 0x41, 0x22}, // С
{0x01, 0x01, 0x7f, 0x01, 0x01}, // Т
{0x07, 0x48, 0x48, 0x48, 0x7F}, // У
{0x1C, 0x22, 0x7F, 0x22, 0x1C}, // Ф
{0x63, 0x14, 0x08, 0x14, 0x63}, // Х
{0x7F, 0x40, 0x40, 0x7F, 0xC0}, // Ц
{0x07, 0x08, 0x08, 0x08, 0x7F}, // Ч
{0x7F, 0x40, 0x7F, 0x40, 0x7F}, // Ш
{0x7F, 0x40, 0x7F, 0x40, 0xFF}, // Щ
{0x01, 0x7F, 0x48, 0x48, 0x70}, // Ъ
{0x7F, 0x48, 0x70, 0x00, 0x7F}, // Ы
{0x00, 0x7F, 0x48, 0x48, 0x70}, // Ь
{0x22, 0x41, 0x49, 0x49, 0x3E}, // Э
{0x7F, 0x08, 0x3E, 0x41, 0x3E}, // Ю
{0x46, 0x29, 0x19, 0x09, 0x7F}, // Я 223

{0x20, 0x54, 0x54, 0x54, 0x78}, //а 224
{0x3c, 0x4a, 0x4a, 0x49, 0x31}, //б
{0x7c, 0x54, 0x54, 0x28, 0x00}, //в
{0x7c, 0x04, 0x04, 0x04, 0x0c}, //г

```
  {0xe0, 0x54, 0x4c, 0x44, 0xfc}, //д
  {0x38, 0x54, 0x54, 0x54, 0x18}, //е
  {0x6c, 0x10, 0x7c, 0x10, 0x6c}, //ж
  {0x44, 0x44, 0x54, 0x54, 0x28}, //з
  {0x7c, 0x20, 0x10, 0x08, 0x7c}, //и
  {0x7c, 0x41, 0x22, 0x11, 0x7c}, //й
  {0x7c, 0x10, 0x28, 0x44, 0x00}, //к
  {0x20, 0x44, 0x3c, 0x04, 0x7c}, //л
  {0x7c, 0x08, 0x10, 0x08, 0x7c}, //м
  {0x7c, 0x10, 0x10, 0x10, 0x7c}, //н
  {0x38, 0x44, 0x44, 0x44, 0x38}, //о
  {0x7c, 0x04, 0x04, 0x04, 0x7c}, //п
  {0x7C, 0x14, 0x14, 0x14, 0x08}, //р
  {0x38, 0x44, 0x44, 0x44, 0x20}, //с
  {0x04, 0x04, 0x7c, 0x04, 0x04}, //т
  {0x0C, 0x50, 0x50, 0x50, 0x3C}, //у
  {0x30, 0x48, 0xfc, 0x48, 0x30}, //ф
  {0x44, 0x28, 0x10, 0x28, 0x44}, //х
  {0x7c, 0x40, 0x40, 0x40, 0xfc}, //ц
  {0x0c, 0x10, 0x10, 0x10, 0x7c}, //ч
  {0x7c, 0x40, 0x7c, 0x40, 0x7c}, //ш
  {0x7c, 0x40, 0x7c, 0x40, 0xfc}, //щ
  {0x04, 0x7c, 0x50, 0x50, 0x20}, //ъ
  {0x7c, 0x50, 0x50, 0x20, 0x7c}, //ы
  {0x7c, 0x50, 0x50, 0x20, 0x00}, //ь
  {0x28, 0x44, 0x54, 0x54, 0x38}, //э
  {0x7c, 0x10, 0x38, 0x44, 0x38}, //ю
  {0x08, 0x54, 0x34, 0x14, 0x7c}, //я 255
};
```

```
//игра  тетрис!
#if (USE_TETRIS == 1)
// *************** НАСТРОЙКИ ТЕТРИС ***************
#define FAST_SPEED 20      // скорость падения при удержании "вниз" (меньше - быстрее)
#define STEER_SPEED 40      // скорость перемещения в бок при удержании кнопки (меньше - быстрее) на BT версии не работает!

// ---------------------- ДЛЯ РАЗРАБОТЧИКОВ ----------------------
#define ADD_COLOR 0x010101

int8_t fig = 0, ang = 0, pos = WIDTH / 2, height = HEIGHT - 1;
int8_t prev_ang, prev_pos, prev_height;

uint32_t colors[6] {0x0000EE, 0xEE0000, 0x00EE00, 0x00EEEE, 0xEE00EE, 0xEEEE00};
uint32_t color = 0x000088;
byte color_index;
byte linesToClear;
boolean down_flag = true;
byte lineCleanCounter;

// самая важная часть программы! Координаты пикселей фигур
//  0 - палка
//  1 - кубик
//  2 - Г
//  3 - Г обратная
//  4 - Z
//  5 - Z обратная
//  6 - T
```

```
const int8_t figures[7][12][2] PROGMEM = {
  {
    { -1, 0}, {1, 0}, {2, 0},
    {0, 1}, {0, 2}, {0, 3},
    { -1, 0}, {1, 0}, {2, 0},
    {0, 1}, {0, 2}, {0, 3},
  },
  {
    {0, 1}, {1, 0}, {1, 1},
    {0, 1}, {1, 0}, {1, 1},
    {0, 1}, {1, 0}, {1, 1},
    {0, 1}, {1, 0}, {1, 1},
  },
  {
    { -1, 0}, { -1, 1}, {1, 0},
    {0, 1}, {0, 2}, {1, 2},
    { -2, 1}, { -1, 1}, {0, 1},
    { -1, 0}, {0, 1}, {0, 2},
  },
  {
    { -1, 0}, {1, 0}, {1, 1},
    {0, 1}, {0, 2}, {1, 0},
    {0, 1}, {1, 1}, {2, 1},
    {0, 1}, {0, 2}, { -1, 2},
  },
  {
    { -1, 0}, {0, 1}, {1, 1},
    {0, 1}, { -1, 1}, { -1, 2},
    { -1, 0}, {0, 1}, {1, 1},
```

```
    {0, 1}, { -1, 1}, { -1, 2},
  },
  {
    { -1, 1}, {0, 1}, {1, 0},
    {0, 1}, {1, 1}, {1, 2},
    { -1, 1}, {0, 1}, {1, 0},
    {0, 1}, {1, 1}, {1, 2},
  },
  {
    { -1, 0}, {0, 1}, {1, 0},
    {0, 1}, {0, 2}, {1, 1},
    { -1, 1}, {0, 1}, {1, 1},
    { -1, 1}, {0, 1}, {0, 2},
  }
};

void tetrisRoutine() {
  if (loadingFlag) {
    FastLED.clear();
    loadingFlag = false;
    newGameTetris();
    gamemodeFlag = true;
    modeCode = 2;
  }

  if (checkButtons()) {

    if (buttons == 3) {   // кнопка нажата
      buttons = 4;
      stepLeft();
```

```
  }

  if (buttons == 1) {
    buttons = 4;
    stepRight();
  }

  if (buttons == 0) {
    buttons = 4;
    if (checkArea(3)) {       // проверка возможности поворота
      prev_ang = ang;         // запоминаем старый угол
      ang = ++ang % 4;        // изменяем ang от 0 до 3 (да, хитро)
      redrawFigure(prev_ang, pos, height);    // перерисовать фигуру
    }
  }

  if (buttons == 2) {             // кнопка вниз удерживается
    buttons = 4;
    gameTimer.setInterval(FAST_SPEED);  // увеличить скорость
  }
}

/*
  if (bt_left.isStep()) {    // кнопка нажата и удерживается
  stepLeft();
  }
  if (bt_right.isStep()) {
  stepRight();
  }
*/
```

```
        if (gameTimer.isReady()) {        // главный таймер игры
          prev_height = height;

          if (!checkArea(0)) {                // проверяем столкновение с другими фигурами
            if (height >= HEIGHT - 2) {   // проиграл по высоте
              gameOverTetris();               // игра окончена, очистить всё
              newGameTetris();                // новый раунд
            } else {                  // если не достигли верха
              fixFigure();              // зафиксировать
              checkAndClear();          // проверить ряды и очистить если надо
              newGameTetris();                // новый раунд
            }
          } else if (height == 0) {       // если достигли дна
            fixFigure();                // зафиксировать
            checkAndClear();            // проверить ряды и очистить если надо
            newGameTetris();                // новый раунд
          } else {                    // если путь свободен
            height--;                         // идём вниз
            redrawFigure(ang, pos, prev_height);  // перерисовка
          }
        }
      }

      // поиск и очистка заполненных уровней
      void checkAndClear() {
        linesToClear = 1;                 // счётчик заполненных строк по вертикали. Искусственно принимаем 1 для работы цикла
        boolean full_flag = true;         // флаг заполненности
```

```
while (linesToClear != 0) {       // чисти чисти пока не будет чисто!
 linesToClear = 0;
 byte lineNum = 255;       // высота, с которой начинаются заполненные строки (искусственно увеличена)
 for (byte Y = 0; Y < HEIGHT; Y++) {   // сканируем по высоте
  full_flag = true;                  // поднимаем флаг. Будет сброшен, если найдём чёрный пиксель
  for (byte X = 0; X < WIDTH; X++) {  // проходимся по строкам
   if ((long)getPixColorXY(X, Y) == (long)0x000000) {   // если хоть один пиксель чёрный
    full_flag = false;                          // считаем строку неполной
   }
  }
  if (full_flag) {        // если нашлась заполненная строка
   linesToClear++;       // увеличиваем счётчик заполненных строк
   if (lineNum == 255)   // если это первая найденная строка
    lineNum = Y;              // запоминаем высоту. Значение 255 было просто "заглушкой"
  } else {              // если строка не полная
   if (lineNum != 255)    // если lineNum уже не 255 (значит строки были найдены!!)
    break;             // покинуть цикл
  }
 }
 if (linesToClear > 0) {             // если найденных полных строк больше 1
  lineCleanCounter += linesToClear;        // суммируем количество очищенных линий (игровой "счёт")
```

```
// заполняем весь блок найденных строк белым цветом слева направо
for (byte X = 0; X < WIDTH; X++) {
  for (byte i = 0; i < linesToClear; i++) {
    leds[getPixelNumber(X, lineNum + i)] = CHSV(0, 0, 255);          // закрашиваем его белым
  }
  FastLED.show();
  delay(5);     // задержка между пикселями слева направо
}
delay(10);

// теперь плавно уменьшаем яркость всего белого блока до нуля
for (byte val = 0; val <= 30; val++) {
  for (byte X = 0; X < WIDTH; X++) {
    for (byte i = 0; i < linesToClear; i++) {
      leds[getPixelNumber(X, lineNum + i)] = CHSV(0, 0, 240 - 8 * val); // гасим белый цвет
    }
  }
  FastLED.show();
  delay(5);       // задержка между сменой цвета
}
delay(10);

// и теперь смещаем вниз все пиксели выше уровня с первой найденной строкой
for (byte i = 0; i < linesToClear; i++) {
  for (byte Y = lineNum; Y < HEIGHT - 1; Y++) {
    for (byte X = 0; X < WIDTH; X++) {
```

```
            drawPixelXY(X, Y, getPixColorXY(X, Y + 1));      // сдвигаем вниз
           }
           FastLED.show();
          }
          delay(100);          // задержка между "сдвигами" всех пикселей на один уровень
        }
       }
      }
      gameTimer.reset();
     }


     // функция фиксации фигуры
     void fixFigure() {
      color += ADD_COLOR;                  // чутка перекрасить
      redrawFigure(ang, pos, prev_height);  // перерисовать
     }

     // проигрыш
     void gameOverTetris() {
      FastLED.clear();
      FastLED.show();

      // тут можно вывести счёт lineCleanCounter
      if (!gameDemo) displayScore(lineCleanCounter);
      delay(1000);
      lineCleanCounter = 0;   // сброс счёта
      FastLED.clear();
      FastLED.show();
```

```
  delay(20);
 }

 // новый раунд
 void newGameTetris() {
  Serial.println("lolkek");      // без этого работает некорректно! магия ебаная
  delay(10);
  buttons = 4;
  height = HEIGHT;    // высота = высоте матрицы
  pos = WIDTH / 2;    // фигура появляется в середине
  fig = random(7);    // выбираем слулчайно фигуру
  ang = random(4);    // и угол поворота
  //color = colors[random(6)];      // случайный цвет

  color_index = ++color_index % 6;  // все цвета по очереди
  color = colors[color_index];

  // если включен демо-режим, позицию по горизонтали выбираем случайно
  if (gameDemo) pos = random(1, WIDTH - 1);

  // возвращаем обычную скорость падения
  gameTimer.setInterval(gameSpeed);
  down_flag = false;  // разрешаем ускорять кнопкой "вниз"
  delay(10);
 }

 // управление фигурами вправо и влево
 void stepRight() {
```

```
  if (checkArea(1)) {
    prev_pos = pos;
    if (++pos > WIDTH) pos = WIDTH;
    redrawFigure(ang, prev_pos, height);
  }
}
void stepLeft() {
  if (checkArea(2)) {
    prev_pos = pos;
    if (--pos < 0) pos = 0;
    redrawFigure(ang, prev_pos, height);
  }
}

// проверка на столкновения
boolean checkArea(int8_t check_type) {
  // check type:
  // 0 - проверка лежащих фигур и пола
  // 1 - проверка стенки справа и фигур
  // 2 - проверка стенки слева и фигур
  // 3 - проверка обеих стенок и пола

  boolean flag = true;
  int8_t X, Y;
  boolean offset = 1;
  int8_t this_ang = ang;

  // этот режим для проверки поворота. Поэтому "поворачиваем"
  // фигуру на следующий угол чтобы посмотреть, не столкнётся ли она с чем
```

```
if (check_type == 3) {
  this_ang = ++this_ang % 4;
  offset = 0;   // разрешаем оказаться вплотную к стенке
}

for (byte i = 0; i < 4; i++) {
  // проверяем точки фигуры
  // pos, height - координаты главной точки фигуры в ГЛОБАЛЬНОЙ системе координат
  // X, Y - координаты остальных трёх точек в ГЛОБАЛЬНОЙ системе координат
  if (i == 0) {   // стартовая точка фигуры (начало отсчёта)
    X = pos;
    Y = height;
  } else {        // остальные три точки
    // получаем глобальные координаты точек, прибавив их положение в
    // системе координат главной точки к координатам самой главной
    // точки в глобальной системе координат. Если ты до сюда дочитал,
    // то стукни мне на почту alex@alexgyver.ru . Печенек не дам, но ты молодец!
    X = pos + (int8_t)pgm_read_byte(&figures[fig][this_ang * 3 + i - 1][0]);
    Y = height + (int8_t)pgm_read_byte(&figures[fig][this_ang * 3 + i - 1][1]);

    // дичь дикая! Но на самом деле просто восстанавливаем из прогмема данные
    // и берём нужное число в массиве. Откуда все эти * 3 -1 ... можно додуматься
  }
```

```
// границы поля
if (check_type == 1 || check_type == 3) {
  if (X + 1 > WIDTH - 1) flag = false;    // смотрим следующий справа
  uint32_t getColor;
  if (Y < HEIGHT)
    getColor = getPixColorXY(X + offset, Y);
  if (getColor != color && getColor != 0x000000) {
    flag = false;         // если не СВОЙ цвет и не чёрный
  }
}

if (check_type == 2 || check_type == 3) {
  if (X - 1 < 0) flag = false;    // смотрим следующий слева
  uint32_t getColor;
  if (Y < HEIGHT)
    getColor = getPixColorXY(X - offset, Y);
  if (getColor != color && getColor != 0x000000) {
    flag = false;         // если не СВОЙ цвет и не чёрный
  }
}

if (check_type == 0 || check_type == 3) {
  uint32_t getColor;
  if (Y < HEIGHT) {
    getColor = getPixColorXY(X, Y - 1);
    if (getColor != color && getColor != 0x000000) {
      flag = false;         // если не СВОЙ цвет и не чёрный
    }
  }
```

```
      }
    }
    return flag;     // возвращаем глобальный флаг, который говорит о том, столкнёмся мы с чем то или нет
  }

  // функция, которая удаляет текущую фигуру (красит её чёрным)
  // а затем перерисовывает её в новом положении
  void redrawFigure(int8_t clr_ang, int8_t clr_pos, int8_t clr_height) {
    drawFigure(fig, clr_ang, clr_pos, clr_height, 0x000000);              // стереть фигуру
    drawFigure(fig, ang, pos, height, color);                             // отрисовать
    FastLED.show();
  }

  // функция, отрисовывающая фигуру заданным цветом и под нужным углом
  void drawFigure(byte figure, byte angle, byte x, byte y, uint32_t color) {
    drawPixelXY(x, y, color);                 // рисуем точку начала координат фигуры
    int8_t X, Y;                              // вспомогательные
    for (byte i = 0; i < 3; i++) {    // рисуем 4 точки фигуры
      // что происходит: рисуем фигуру относительно текущей координаты падающей точки
      // просто прибавляем "смещение" из массива координат фигур
      // для этого идём в прогмем (функция pgm_read_byte)
      // обращаемся к массиву по адресу &figures
      // преобразовываем число в int8_t (так как progmem работает только с "unsigned"
```

```
      // angle * 3 + i - обращаемся к координатам согласно текущему углу поворота фигуры

      X = x + (int8_t)pgm_read_byte(&figures[figure][angle * 3 + i][0]);
      Y = y + (int8_t)pgm_read_byte(&figures[figure][angle * 3 + i][1]);
      if (Y > HEIGHT - 1) continue;    // если выходим за пределы поля, пропустить отрисовку
       drawPixelXY(X, Y, color);
     }
    }

    #elif (USE_TETRIS == 0)
    void tetrisRoutine() {
     return;
    }
    #endif
    // крутые полноэкраные эффекты
    // ****************** НАСТРОЙКИ *****************
    // "масштаб" эффектов. Чем меньше, тем крупнее!
    #define MADNESS_SCALE 100
    #define CLOUD_SCALE 30
    #define LAVA_SCALE 50
    #define PLASMA_SCALE 30
    #define RAINBOW_SCALE 30
    #define RAINBOW_S_SCALE 20
    #define ZEBRA_SCALE 30
    #define FOREST_SCALE 120
    #define OCEAN_SCALE 90

    // ****************** ДЛЯ РАЗРАБОТЧИКОВ *****************
```

```
#if (USE_NOISE_EFFECTS == 1)
// The 16 bit version of our coordinates
static uint16_t x;
static uint16_t y;
static uint16_t z;

uint16_t speed = 20; // speed is set dynamically once we've started up
uint16_t scale = 30; // scale is set dynamically once we've started up

// This is the array that we keep our computed noise values in
#define MAX_DIMENSION (max(WIDTH, HEIGHT))
#if (WIDTH > HEIGHT)
uint8_t noise[WIDTH][WIDTH];
#else
uint8_t noise[HEIGHT][HEIGHT];
#endif

CRGBPalette16 currentPalette( PartyColors_p );
uint8_t colorLoop = 1;
uint8_t ihue = 0;

void madnessNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    scale = MADNESS_SCALE;
    modeCode = 3;
  }
  fillnoise8();
  for (int i = 0; i < WIDTH; i++) {
    for (int j = 0; j < HEIGHT; j++) {
```

```
            CRGB thisColor = CHSV(noise[j][i], 255, noise[i][j]);
            drawPixelXY(i, j, thisColor);          //leds[getPixelNumber(i, j)] =
CHSV(noise[j][i], 255, noise[i][j]);

            // You can also explore other ways to constrain the hue used, like below
            // leds[XY(i,j)] = CHSV(ihue + (noise[j][i]>>2),255,noise[i][j]);
          }
        }
        ihue += 1;
      }
      void rainbowNoise() {
        if (loadingFlag) {
          loadingFlag = false;
          currentPalette = RainbowColors_p;
          scale = RAINBOW_SCALE; colorLoop = 1;
          modeCode = 7;
        }
        fillNoiseLED();
      }
      void rainbowStripeNoise() {
        if (loadingFlag) {
          loadingFlag = false;
          currentPalette = RainbowStripeColors_p;
          scale = RAINBOW_S_SCALE; colorLoop = 1;
          modeCode = 8;
        }
        fillNoiseLED();
      }
      void zebraNoise() {
        if (loadingFlag) {
```

```
    loadingFlag = false;
    // 'black out' all 16 palette entries...
    fill_solid( currentPalette, 16, CRGB::Black);
    // and set every fourth one to white.
    currentPalette[0] = CRGB::White;
    currentPalette[4] = CRGB::White;
    currentPalette[8] = CRGB::White;
    currentPalette[12] = CRGB::White;
    scale = ZEBRA_SCALE; colorLoop = 1;
    modeCode = 9;
  }
  fillNoiseLED();
}
void forestNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    currentPalette = ForestColors_p;
    scale = FOREST_SCALE; colorLoop = 0;
    modeCode = 10;
  }
  fillNoiseLED();
}
void oceanNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    currentPalette = OceanColors_p;
    scale = OCEAN_SCALE; colorLoop = 0;
    modeCode = 11;
  }
```

```
  fillNoiseLED();
}
void plasmaNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    currentPalette = PartyColors_p;
    scale = PLASMA_SCALE; colorLoop = 1;
    modeCode = 6;
  }
  fillNoiseLED();
}
void cloudNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    currentPalette = CloudColors_p;
    scale = CLOUD_SCALE; colorLoop = 0;
    modeCode = 4;
  }
  fillNoiseLED();
}
void lavaNoise() {
  if (loadingFlag) {
    loadingFlag = false;
    currentPalette = LavaColors_p;
    scale = LAVA_SCALE; colorLoop = 0;
    modeCode = 5;
  }
  fillNoiseLED();
}
```

```
// ****************** СЛУЖЕБНЫЕ ******************
void fillNoiseLED() {
  uint8_t dataSmoothing = 0;
  if ( speed < 50) {
    dataSmoothing = 200 - (speed * 4);
  }
  for (int i = 0; i < MAX_DIMENSION; i++) {
    int ioffset = scale * i;
    for (int j = 0; j < MAX_DIMENSION; j++) {
      int joffset = scale * j;

      uint8_t data = inoise8(x + ioffset, y + joffset, z);

      data = qsub8(data, 16);
      data = qadd8(data, scale8(data, 39));

      if ( dataSmoothing ) {
        uint8_t olddata = noise[i][j];
        uint8_t newdata = scale8( olddata, dataSmoothing) + scale8( data, 256 - dataSmoothing);
        data = newdata;
      }

      noise[i][j] = data;
    }
  }
  z += speed;

  // apply slow drift to X and Y, just for visual variation.
  x += speed / 8;
```

```
  y -= speed / 16;

  for (int i = 0; i < WIDTH; i++) {
    for (int j = 0; j < HEIGHT; j++) {
      uint8_t index = noise[j][i];
      uint8_t bri =   noise[i][j];
      // if this palette is a 'loop', add a slowly-changing base value
      if ( colorLoop) {
        index += ihue;
      }
      // brighten up, as the color palette itself often contains the
      // light/dark dynamic range desired
      if ( bri > 127 ) {
        bri = 255;
      } else {
        bri = dim8_raw( bri * 2);
      }
      CRGB color = ColorFromPalette( currentPalette, index, bri);
      drawPixelXY(i, j, color);   //leds[getPixelNumber(i, j)] = color;
    }
  }
  ihue += 1;
}

void fillnoise8() {
  for (int i = 0; i < MAX_DIMENSION; i++) {
    int ioffset = scale * i;
    for (int j = 0; j < MAX_DIMENSION; j++) {
      int joffset = scale * j;
      noise[i][j] = inoise8(x + ioffset, y + joffset, z);
```

```
    }
  }
  z += speed;
}

#else
void madnessNoise() {
  return;
}
void cloudNoise() {
  return;
}
void lavaNoise() {
  return;
}
void plasmaNoise() {
  return;
}
void rainbowNoise() {
  return;
}
void rainbowStripeNoise() {
  return;
}
void zebraNoise() {
  return;
}
void forestNoise() {
  return;
}
```

```
void oceanNoise() {
  return;
}
#endif
// работа с бегущим текстом

// *************** НАСТРОЙКИ ***************
#define TEXT_DIRECTION 1  // 1 - по горизонтали, 0 - по вертикали
#define MIRR_V 0          // отразить текст по вертикали (0 / 1)
#define MIRR_H 0          // отразить текст по горизонтали (0 / 1)

#define TEXT_HEIGHT 0      // высота, на которой бежит текст (от низа матрицы)
#define LET_WIDTH 5       // ширина буквы шрифта
#define LET_HEIGHT 8      // высота буквы шрифта
#define SPACE 1           // пробел

// --------------------- ДЛЯ РАЗРАБОТЧИКОВ ----------------------

#if (USE_FONTS == 1)

int offset = WIDTH;

void fillString(String text, uint32_t color) {
  if (loadingFlag) {
    offset = WIDTH;   // перемотка в правый край
    loadingFlag = false;
    modeCode = 0;
    fullTextFlag = false;
  }
```

```
  if (scrollTimer.isReady() || (!BTcontrol && !gamemodeFlag)) {
    FastLED.clear();
    byte i = 0, j = 0;
    while (text[i] != '\0') {
      if ((byte)text[i] > 191) {    // работаем с русскими буквами!
        i++;
      } else {
        drawLetter(j, text[i], offset + j * (LET_WIDTH + SPACE), color);
        i++;
        j++;
      }
    }
    fullTextFlag = false;

    offset--;
    if (offset < -j * (LET_WIDTH + SPACE)) {    // строка убежала
      offset = WIDTH + 3;
      fullTextFlag = true;
    }
    FastLED.show();
  }
}

void drawLetter(uint8_t index, uint8_t letter, int16_t offset, uint32_t color) {
  int8_t start_pos = 0, finish_pos = LET_WIDTH;
  CRGB letterColor;
  if (color == 1) letterColor = CHSV(byte(offset * 10), 255, 255);
  else if (color == 2) letterColor = CHSV(byte(index * 30), 255, 255);
  else letterColor = color;
```

```
  if (offset < -LET_WIDTH || offset > WIDTH) return;
  if (offset < 0) start_pos = -offset;
  if (offset > (WIDTH - LET_WIDTH)) finish_pos = WIDTH - offset;

  for (byte i = start_pos; i < finish_pos; i++) {
    int thisByte;
    if (MIRR_V) thisByte = getFont((byte)letter, LET_WIDTH - 1 - i);
    else thisByte = getFont((byte)letter, i);

    for (byte j = 0; j < LET_HEIGHT; j++) {
      boolean thisBit;

      if (MIRR_H) thisBit = thisByte & (1 << j);
      else thisBit = thisByte & (1 << (LET_HEIGHT - 1 - j));

      // рисуем столбец (i - горизонтальная позиция, j - вертикальная)
      if (TEXT_DIRECTION) {
        if (thisBit) leds[getPixelNumber(offset + i, TEXT_HEIGHT + j)] =
letterColor;
        else drawPixelXY(offset + i, TEXT_HEIGHT + j, 0x000000);
      } else {
        if (thisBit) leds[getPixelNumber(i, offset + TEXT_HEIGHT + j)] =
letterColor;
        else drawPixelXY(i, offset + TEXT_HEIGHT + j, 0x000000);
      }

    }
  }
}
```

```
// ------------- СЛУЖЕБНЫЕ ФУНКЦИИ --------------

// интерпретатор кода символа в массиве fontHEX (для Arduino IDE 1.8.* и выше)
uint8_t getFont(uint8_t font, uint8_t row) {
  font = font - '0' + 16;   // перевод код символа из таблицы ASCII в номер согласно нумерации массива
  if (font <= 90) return pgm_read_byte(&(fontHEX[font][row]));       // для английских букв и символов
  else if (font >= 112 && font <= 159) {    // и пизд*ц ждя русских
   return pgm_read_byte(&(fontHEX[font - 17][row]));
  } else if (font >= 96 && font <= 111) {
   return pgm_read_byte(&(fontHEX[font + 47][row]));
  }
}

#elif (USE_FONTS == 0)
void fillString(String text, uint32_t color) {
  fullTextFlag = false;
  modeCode = 0;
  return;
}
#endif

/*
  // интерпретатор кода символа по ASCII в его номер в массиве fontHEX (для Arduino IDE до 1.6.*)
  uint8_t getFontOld(uint8_t font, uint8_t row) {
```

```
font = font - '0' + 16;   // перевод код символа из таблицы ASCII в номер согласно нумерации массива
if (font < 126) return pgm_read_byte(&(fontHEX[font][row]));     // для английских букв и символов
else return pgm_read_byte(&(fontHEX[font - 65][row]));           // для русских букв и символов (смещение -65 по массиву)
}
*/
```

ПРИЛОЖЕНИЕ Б – СПИСОК ОБОЗНАЧЕНИЙ И СОКРАЩЕНИЙ

IP – Классификация способа защиты, обеспечиваемого оболочкой технического устройства от доступа к опасным частям, попадания внешних твёрдых предметов и воды и проверяемого стандартными методами испытаний

UART – универсальный асинхронный приемопередатчик

КПД – коэффициент полезного действия

САПР – система автоматизированного проектирования

АСУ – автоматизированная система управления

ООН реклама – уличная реклама

IDE – интегрированная среда разработки

Конструкция POS – конструкция точки продаж

РЕФЕРАТ

Кобзарев А.К. Разработка многофункциональной светодиодной строки для ИП Кобзарева О.О., выпускная квалификационная работа: стр.117, рис. 14, табл. 5, формул 4, библ. 26 назв.

Ключевые слова: реклама, бегущая строка, светодиодная матрица, соединение, проектирование, автоматизация, разработка, система, автоматизированное устройство, микроконтроллер, интерфейс, структурная схема, принципиальная схема.

Объект исследования – уличный маркетинг.

Цель работы – разработка многофункциональной светодиодной строки..

В процессе проектирования были получены следующие результаты:

– Создана структурная и принципиальная схема бегущей строки;

– Создан макет, по средствам имеющихся компонентов и материалов, который способен подключаться к компьютеру;

– Изучены программные интерфейсы для написания кода;

– Изучены программы для построения чертежей и принципиальной схемы;

– Получены навыки процесса сборки, правильного подключения.

Основные конструктивные и экономические показатели: низкая стоимость при высоком функционале.

Система может применяться для отображения рекламной информации.