Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Архитектура вычислительных систем

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

«Программирование на платформе Arduino»

БГУИР КП 1-40 04 01 006 ПЗ

Студент гр. 753502

Владымцев В.Д.

Руководитель

Ст. преподаватель кафедры информатики

В.В. Шиманский

Минск 2019

Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Факультет компьютерных систем и сетей

Кафедра информатики

УТВЕРЖДАЮ
Заведующий кафедрой
Информатики
_____Волорова Н. А.
«___» __ _____ 2019 г.

ЗАДАНИЕ
по курсовому проекту

Группа 753502

Студенту *Владымцеву Вадиму Денисовичу*

1.Тема проекта: Программирование на платформе Arduino
2.Сроки сдачи студентом законченного проекта: 21.12.2019 г.
3.Исходные данные к проекту: Для написания курсового проекта была выбрана платформа Arduino, программное обеспечение Arduino IDE с Prismatik в модификации psieg, язык программирования C.
4.Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов):
Введение
Раздел 1. Анализ предметной области
Раздел 2. Описание работы контроллеров
Раздел 3. Сравнение с существующими аналогами
Раздел 4. Анализ работы программы устройства
Заключение. Список использованных источников. Приложение
5.Перечень графического материала (с указанием обязательных чертежей и графиков):
_____
6.Консультанты по проекту: Шиманский В. В.
7.Дата выдачи задания: 20.09.2019 г
8.Календарный график работы над проектом на весь период проектирования (с указанием сроков выполнения и трудоемкости отдельных этапов):

№ п/п	Наименование этапов курсового проекта	Срок выполнения этапов проекта	Примечание
1.	1-я опроцентовка (введение, раздел 1)	5.10.2019	30%
2.	2-я опроцентовка (раздел 2, раздел 3)	28.10.2019	65%
3.	3-я опроцентовка (демонстрация, заключение)	12.11.2019	100%
4.	Защита курсового проекта	21.12.2019	Согласно графику

Руководитель _____ (Шиманский В. В.)
Задание принял к исполнению 15.09.2019 _____ (_____)

Введение

Philips Lighting, мировой лидер в области освещения, представила результаты исследования, согласно которым люди во всём мире не уделяют должного внимания заботе о здоровье глаз. Это происходит на фоне рекордного уровня близорукости — Всемирная организация здравоохранения прогнозирует, что к 2050 году у каждого второго человека будет миопия, а из-за компьютеров и гаджетов нагрузка на зрение только увеличится. Несмотря на то, что в повседневной жизни люди постоянно полагаются на зрение, массовое увлечение фитнесом и похудением вытеснило заботу о здоровье глаз на второй план.

Ученые Philips Lighting решили выяснить, как качественное LED освещение может обеспечить глазам больший комфорт, и провели исследование с более чем 8 000 участниками из 11 стран: Китая, Чехии, Франции, Германии, Индонезии, Польши, Испании, Швеции, Таиланда, Турции и США. Результаты показали, что в среднем человек более шести часов в день находится перед экранами телевизора или компьютера и часто сталкивается с напряжением глаз, при этом только 42% опрошенных используют щадящее освещение. Аналогичным образом, только треть (32%) людей во всем мире, покупая осветительные приборы, учитывает, насколько свет комфортен для их зрения. Исследование также продемонстрировало, что люди не отдают заботе о глазах приоритет в вопросах здоровья и общего самочувствия. Сейчас большинство во всем мире (68%) считает вес и физическую форму (57%) основными индикаторами здоровья, при этом только треть (34%) оценивает зрение как показатель общего самочувствия. Кроме того, лишь половина опрошенных указали, что забота о зрении является для них одним трех приоритетных направлений в области личного ухода, а 57% не посещают офтальмолога на регулярной основе. К тому же, только два человека из пяти (40%) выбирают освещение, обеспечивающее здоровье глаз.

Так же учёные доказали, что подсветка позади экрана снимает напряжение с глаз при работе с монитором. Холодный белый свет не всегда может быть правильным выходом в этой ситуации, так как слишком тёмные оттенки различных цветов на экране могут сильно контрастировать с подсветкой. Тем самым создавая тёмные пятна перед зрением человека и усугубляя ситуацию со зрением. Достаточно примитивным решением этой проблемы является использование настольной лампы, которую следует направлять на стену за монитором компьютера.

Существует более креативное решение данной проблемы, которое, кроме того, позволяет визуально расширить отображаемый контент на мониторе. Впервые данную технологию запатентовала компания Philips для своих телевизоров в 2002 году. Она работала по вертикальным краям экрана и, соответственно, подсвечивала стену различными цветами за телевизором в соответствии картинке на экране. В 2007 году кампания объявила о полной

фоновой подсветке и выпустила соответствующий патент. Далее данная технология приобрела название Ambilight.

Рисунок 1. Телевизор Philips 46PFL9705H с включенной 3-сторонней функцией «Ambilight».

Технология Ambilight развивалась, эволюционировала и насчитывает 3 поколения.

- Обычная подсветка лампой.
- Ambilight 2 — двухканальная подсветка. Привязка подсветки идет к преобладающему цвету.
- Ambilight Surround — трехсторонняя подсветка.
- Ambilight Full Surround. В этом поколении экран окружили освещением со всех сторон. Теперь процессор анализирует четыре зоны на экране в соответствии с количеством подсвечивающих ламп. Также добавили экран-панель к корпусу телевизора. Этот экран делает освещение еще мягче.
- Ambilight Spectra — это технология, которая создает как бы объемное изображение за счет использования светодиодов. Эта модификация технологии за счет использования современных микропроцессоров отслеживает экранные цвета и действия и формирует вокруг телевизора красивую подсветку, которая расширяет картинку на экране.

В 2019 году есть два типа фоновой адаптивной подсветки экранов телевизоров или мониторов: Edge LED и Direct LED. Обе подсветки имеют свои плюсы и минусы, однако в широком кругу не используются, так как компания Philips не продаёт права на использование этих технологий.

Темой данного курсового проекта является разработка аналога вышеописанной системы динамической подсветки монитора компьютера на платформе Arduino Nano и Arduino UNO.

1. Анализ предметной области

1.1 Краткие теоретические сведения

Arduino - аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода-вывода и среда разработки на языке Processing/Wiring. Arduino может использоваться как для создания автономных интерактивных объектов, так и подключаться к программному обеспечению, выполняемому на компьютере (например, Adobe Flash, Processing, Max, Pure Data,SuperCollider). Рассылаемые в настоящее время версии могут быть заказаны уже распаянными. Информация об устройстве платы находится в открытом доступе и может быть использована теми, кто предпочитает собирать платы самостоятельно. Микроконтроллеры ATmega328 дёшевы и стоят около 10$.

Проект Arduino был удостоен почётного упоминания при вручении призов Prix Ars Electronica 2006 в категории Digital Communities.

Интегрированная среда разработки Arduino - это кроссплатформенное приложение на Java, включающее в себя редактор кода, компилятор и модуль передачи прошивки в плату.

Теоретические сведения о платформе Arduino

Плата Arduino состоит из микроконтроллера Atmel AVR (ATmega328P и ATmega168 в новых версиях и ATmega8 в старых), а также элементов обвязки для программирования и интеграции с другими схемами. На многих платах присутствует линейный стабилизатор напряжения +5В или +3,3В. Тактирование осуществляется на частоте 16 или 8 МГц кварцевым резонатором (в некоторых версиях керамическим резонатором). В микроконтроллер предварительно прошивается загрузчик BootLoader, поэтому внешний программатор не нужен.

На концептуальном уровне все платы программируются через RS-232 (последовательное соединение), но реализация этого способа отличается от версии к версии. Плата Serial Arduino содержит простую инвертирующую схему для конвертирования уровней сигналов RS-232 в уровни ТТЛ, и наоборот. Текущие рассылаемые платы, например, Diecimila, программируются через USB, что осуществляется благодаря микросхеме конвертера USB-to-Serial FTDI FT232R. В версии платформы Arduino Uno в качестве конвертера используется микроконтроллер Atmega8 в SMD-корпусе. Данное решение позволяет программировать конвертер так, чтобы платформа сразу определялась как мышь, джойстик или иное устройство по усмотрению разработчика со всеми необходимыми дополнительными сигналами управления. В некоторых вариантах, таких как Arduino Mini или неофициальной Boarduino, для программирования требуется подключение отдельной платы USB-to-Serial или кабеля.

Рисунок 2. Arduino Nano V.3.

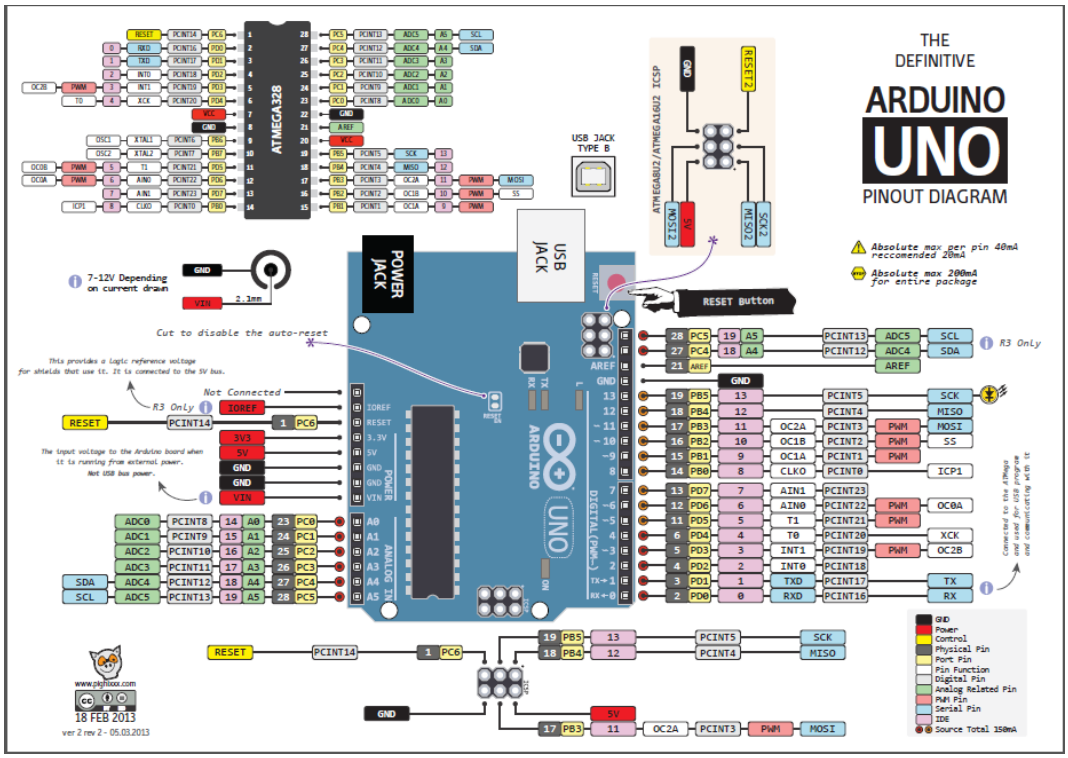

Рисунок 3. Arduino UNO.

Платы Arduino позволяют использовать большую часть I/O выводов микроконтроллера во внешних схемах. Например, в плате Diecimila доступно 14 цифровых входов/выходов, 6 из которых могут выдавать ШИМ сигнал, и 6 аналоговых входов. Эти сигналы доступны на плате через контактные площадки или штыревые разъемы. Также доступны несколько видов внешних

плат расширения, называемых «англ. shields» (дословно: «щиты»), которые присоединяются к плате Arduino через штыревые разъёмы.

Что можно подключить к Arduino

К пинам микроконтроллера можно подключать огромное количество разнообразных устройств и датчиков. Ардуино умеет считывать значения датчиков, обрабатывать их и управлять механизмами в соответствии с установленной прошивкой. Например: можно подключить датчик света и реле. Когда освещение в помещении становится ниже заданного уровня ардуино открывает реле. Это самый простой пример использования. Ниже не полный перечень устройств и датчиков, которые можно подключить:

Переферийные устройства:

- Кнопки, переключатели, сенсорные панели
- Светодиоды
- Динамики и микрофоны
- Коллекторные, безколлекторные и шаговые электродвигатели
- Сервоприводы
- ЖК и LCD дисплеи.
- Устройства считывающие радиометки RFID и NFC
- Ультразвуковые и лазерные датчики расстояния
- Модули Ethernet, WiFi и Bluetooth
- Кардридеры SD
- Модули GSM для совершения звонков и приема/отправки SMS
- GPS для получения точных координат местоположения

Датчики:

- Освещенности
- Магнитного поля
- Температуры
- Влажности воздуха и почвы
- Уровня шума
- Вибрации
- Огня и дыма
- Электронные компасы
- Гироскопы
- Акселерометры

Рисунок 4. Два микроконтроллера Arduino подключенных совместно.

Программное обеспечение

Интегрированная среда разработки Arduino - это кроссплатформенное приложение на Java, включающее в себя редактор кода, компилятор и модуль передачи прошивки в плату.

Среда разработки основана на языке программирования Processing и спроектирована для программирования новичками, не знакомыми близко с разработкой программного обеспечения. Язык программирования аналогичен используемому в проекте Wiring. Строго говоря, это C++, дополненный некоторыми библиотеками. Программы обрабатываются с помощью препроцессора, а затем компилируется с помощью AVR-GCC.

***Операторы**:*

- setup()
- loop()

Управляющие операторы:

- if
- if…else
- for
- switch case
- while
- do… while
- break
- continue
- return
- goto

Синтаксис:

- ; (точка с запятой)
- {} (фигурные скобки)
- // (однострокoвый коментарий)
- /* */ (многострокoвый коментарий)
- #define
- #include

Арифметические операторы:

- = (оператор присваивания)
- + (сложение)
- — (вычитание)
- (умножение)
- / (деление)
- % (остаток от деления)

Операторы сравнения:

- == (равно)
- != (не равно)
- < (меньше чем)
- > (больше чем)
- <= (меньше или равно)
- >= (больше или равно)

Логические операторы:

- && (И)
- || (ИЛИ)
- ! (НЕ)

Указатели доступа:

- указатель
- & ссылка

Битовые операторы:

- & (побитовое И)
- | (побитовое ИЛИ)
- ^ (побитовое XOR или исключающее ИЛИ)
- ~ (побитовое НЕ)
- << (побитовый сдвиг влево)
- >> (побитовый сдвиг вправо)

Составные операторы:

- ++ (инкремент)
- — (декремент)
- += (составное сложение)
- -= (составное вычитание)
- *= (составное умножение)

- /= (составное деление)
- &= (составное побитовое И)
- |= (составное побитовое ИЛИ)

Данные:

Константы:

- HIGH | LOW
- INPUT | OUTPUT| INPUT_PULLUP
- true | false
- целочисленные константы
- константы с плавающей точкой

Типы данных:

- void
- boolean
- char
- unsigned char
- byte
- int
- unsigned int
- word
- long
- unsigned long
- short
- float
- double
- string — массив символов
- String — объект
- массивы

Преобразование типов:

- char()
- byte()
- int()
- word()
- long()
- float()

Область видимости переменной и спецификаторы:

- Область видимости переменной
- static
- volatile
- const

Вспомогательная функция:

- sizeof()

Функции:

Цифровой ввод/вывод:

- pinMode()
- digitalWrite()
- digitalRead()

Аналоговый ввод/вывод:

- analogReference()
- analogRead()
- analogWrite() — PWM

Расширенный ввод/вывод:

- tone()
- noTone()
- shiftOut()
- shiftIn()
- pulseIn()

Время:

- millis()
- micros()
- delay()
- delayMicroseconds()

Математические вычисления:

- min()
- max()
- abs()
- constrain()
- map()
- pow()
- sqrt()
- sq()

Тригонометрия:

- sin()
- cos()
- tan()

Случайные числа:

- randomSeed()
- random()

Биты и байты:

- lowByte()
- highByte()
- bitRead()
- bitWrite()

- bitSet()
- bitClear()
- bit()

Внешние прерывания:

- attachInterrupt()
- detachInterrupt()

Прерывания:

- interrupts()
- noInterrupts()

Среда разработки

Среда разработки Arduino состоит из встроенного текстового редактора программного кода, области сообщений, окна вывода текста(консоли), панели инструментов с кнопками часто используемых команд и нескольких меню. Для загрузки программ и связи среда разработки подключается к аппаратной части Arduino.

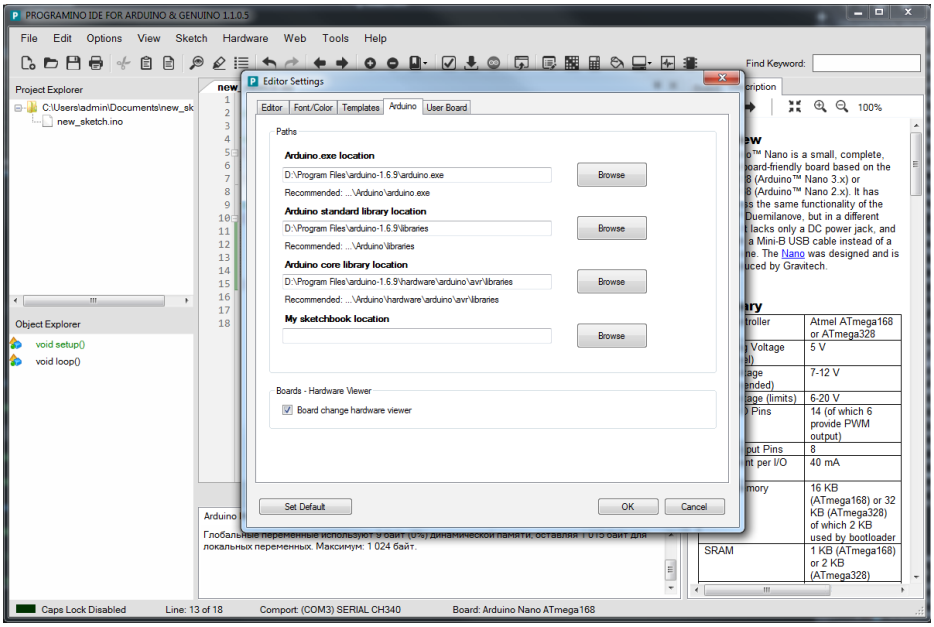

Рис 3. Среда разработки PROGRAMINO для Arduino и GENUINO

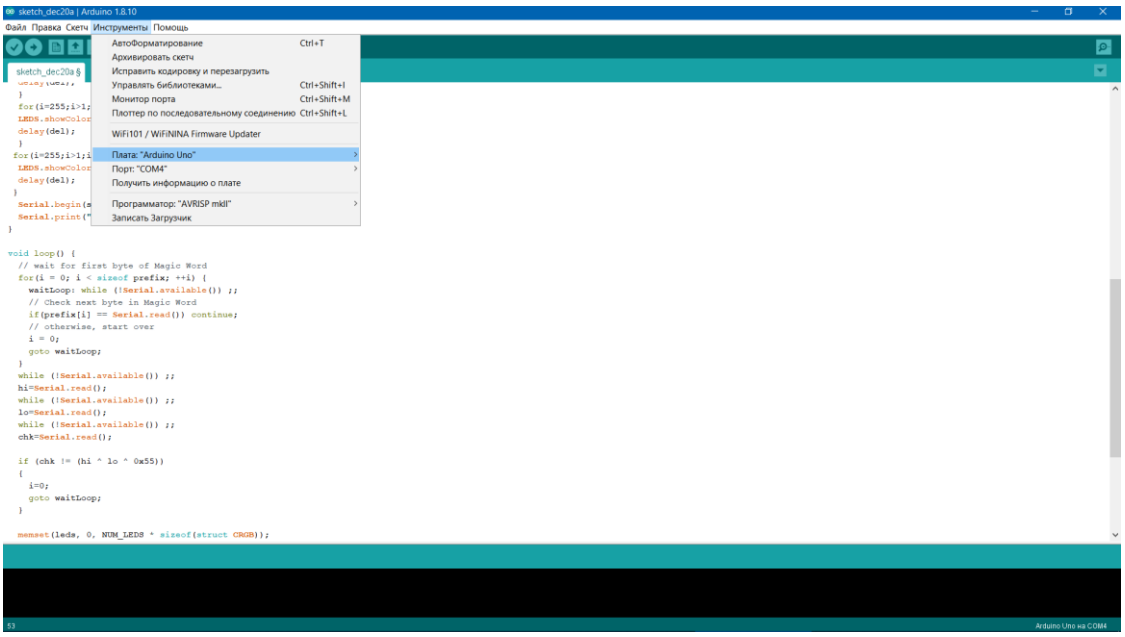

Рис 4. Среда разработки Arduino

Программа, написанная в среде Arduino, называется скетч. Скетч пишется в текстовом редакторе, имеющем инструменты вырезки/вставки, поиска/замены текста. Во время сохранения и экспорта проекта в области сообщений появляются пояснения, также могут отображаться возникшие ошибки. Окно вывода текста(консоль) показывает сообщения Arduino, включающие полные отчеты об ошибках и другую информацию. Кнопки панели инструментов позволяют проверить и записать программу, создать, открыть и сохранить скетч, открыть мониторинг последовательной шины.

1.2 Анализ существующих готовых решений

1.2.1 Коммерческие решения

Среди *коммерческих* разработок существуют решения от компании Philips. Старый патент размещён на сайте Philips и приставка Lightpack для любого устройства. Сначала рассмотрим решение компании *Philips* подробнее:

Процесс обработки сигнала происходит через драйвер монитора. Он обрабатывает сигнал поступающий с видеокарты и отправляет пакеты данных по 24 байта (яркость, красный, зелёный, синий) на встроенную светодиодную ленту.

Перейдём к решению *Lightpack*:

Lightpack представляет из себя коробку с микропроцессором внутри, который анализурет поступаемый сигнал с HDMI порта.

- Сигнал со всех устройств подается на 5in-1out HDMI свитч, который поддерживает HDCP.

- Выходной сигнал подается на 1in-2out HDMI splitter, который мало того, что поддерживает HDCP, так еще отключайте его на выходе.
- Один из выходных сигналов идет на телевизор.
- Другой выходной сигнал идет на HDMI to AV конвертер.
- S-Video сигнал идет на коробочку захвата.
- Коробочка захвата подключается к вечно работающему HTCP по USB, который подключен к микрокомпьютеру.

Сейчас технология Lightpack обновилась и обзавелась поддерживает 4K мониторов. Различия в спецификациях коммерческих решений можно посмотреть в таблице ниже.

Возможности	Lightpack HD	Lightpack UHD	Philips Ambilight
Работает на ТВ	Да	Да	Нет
Работает на любом ПК	Да	Да	Да
Работает на любом HDMI порте	Да	Да	Да
4 HDMI выхода для всех видов девайсов	Да	Да	Нет
Поддержка разрешения 4K	Нет	Да	Да
LED лента для любых размерво мониторов	Да	Да	Нет
Простая установка	Да	Да	Да
Поддержка 30 градусной изогнутости экрана	Да	Да	Да
Управление с смартфона	Да	Да	Нет
Управление с ПК	Да	Да	Да
Работа в Flash режиме	Да	Да	Да
Удалённый контроль	Да	Да	Нет

Таблица 1. Сравнение коммерческих технологий.

Исходя из полученных результатов мы видим, что технология Lightpack UHD лучше своих конкурентов. Следовательно, в курсовом проекте будет делаться упор на повторение результата технологии Lightpack UHD.

1.2.2 Некоммерческие решения

Рассмотрим бесплатные решения с открытыми исходным кодом.

Первое решение подразумевает использование HDMI разветвителя и последующий программный анализ HDMI сигнала на мощном одноплатном компьютере. Решение похоже на коммерческий продукт Lightpack. В качестве одноплатного компьютера можно использовать Raspberry Pi. В данном способе реализации используется готовый компьютер, что не подходит теме курсовой проекта. Рассмотрим положительные стороны проекта:

- Качественный бесперебойный сигнал
- Использование дешёвых одноплатных компьютеров

Минусы:

- Используется готовый компьютер
- Увеличенное энергопотребление

Второе решение подразумевает разработку аппаратного устройства, которое будет анализировать HDMI сигнал. Для упрощения тестирования реализации можно воспользоваться написанием VHDL программы для ПЛИС с HDMI входом. Рассмотрим плюсы реализации данного проекта:

- Использование интегральной схемы
- Дешевизна проекта
- Собственное гибкое ПО

Минусы:

- Медленная обработка сигнала
- Задержка подачи сигнала
- Сложная модификация алгоритмов интегральной платы

Третий способ подразумевает использование микроконтроллера Arduino и ПК. Программное обеспечение на ПК анализирует изображение на экране и формирует специальные команды для микроконтроллера через USB (версии 2.0 и старше), который, в свою очередь, управляет световой лентой. В данной реализации возможно применить световую ленту не только в качестве динамической подсветки монитора. При изменении ПО на компьютере световую ленту также можно использовать в качестве устройства визуализации аудиосигнала. Рассмотрим плюсы реализации проекта:

- Использование дешёвого микроконтроллера Arduino
- Собственное ПО
- Визуализация аудиосигнала по заданным цветам
- Независимо программируемый цвет ленты
- Возможность работы на любом ПК
- Возможность работы на любой операционной системе
- Гибкое ПО

Минусы:

- Скорость работы
- Излишнее тепловыделение

- Черезмерное энергопотребление
- Задержка сигнала не менее 30 миллисекунд
- Отдельный блок питания

1.3 Постановка задачи

Требуется разработать адаптивную подсветку монитора на платформе Arduino для установки позади экрана.

Конечный продукт должен обладать следующими качествами:

- Подключение через USB
- Потоковая передача на всю цепь с задержкой не более 50 миллисекунд
- С открытым исходным кодом
- Поддержка любого форм-фактора экрана
- Поддержка любого разрешения экрана
- Минимальный размер микроконтроллера
- Возможность расширения цепи светодиодов
- Работа на платформе Arduino UNO и NANO
- Усиленное питание цепи

2. Описание работы подсветки Adalight

2.1 Начало работы

В предыдущей главе мы определили требования которым должен отвечать конечный продукт. Этот пункт будет посвящен собственно его проектированию и разработке.

Первое, что должно быть сделано - проектирование системы в целом и подбор необходимых комплектующих. После – программные модули для работы Arduino и взаимодействия с API.

2.2 Необходимые комплектующие для Lightpack Arduino

Для создания Adalight Arduino понадобятся следующие компоненты:

- Индивидуально адресуемая светодиодная лента RGB
- Arduino UNO или NANO
- Макетная плата небольшого размера
- Несколько кабелей
- 12V DC адаптер питания
- Двусторонний скотч
- 4-5 Скрепки
- Затяжки (стяжки) пластиковые для проводов

2.3 Программные модули Adalight Arduino

Для создания Adalight Arduino необходими следующие прогарммные модули:

- IDE Arduino с поддеркой Old Bootloader
- Свободное ПО Prismatik которое позволяет захватывать изображение с минимальными задержками через средства ОС.
- Библиотека FastLED для работы с светодиодными лентами
- Модуль Lightpack (Модифицированный)

2.4 Схема подключения

Arduino Nano – плата, которая работает на чипе ATmega328P и имеет минимальные размеры, которые лучше всего подходят для создания компактных устройств.

Arduino Nano - это аналог Arduino Uno, которая также работает на чипе ATmega328P, но отличается формфактором платы, которая в 2-2,5 раза меньше, чем Uno (*53 x 69 мм UNO против 19 x 43 мм NANO*). Размеры подобны маленькому смартфону, и позволяют легко собирать сложные схемы навесным монтажом, но после стадии создания макета идёт сборка

действующих экземпляров, а для этого лучше подходит как раз Nano. Отличие такой миниатюрной платы, заключается в отсутствии вынесенного гнезда для внешнего питания, но вместо него с легкостью можно подключиться напрямую к пинам. В плате используется чип FTDI FT232RL для USB-Serial преобразования и примененяется mini-USB кабель для связи с Arduino вместо стандартного. Связь с различными устройствами обеспечивают UART, I2C и SPI интерфейсы.

2.5 Минимальный набор для создания Adalight Arduino

- Arduino Uno или Nano с процессором ATmega328P
- 4 провода из комплекта Arduino starter kit
- Кабель USB 2.0 – mini USB
- Кабель USB 2.0 – USB Type B
- Блок питания с минимальными характеристиками 5V, 3A (15Вт)
- 3 светодиодные RGB ленты WS2812 или аналогичные
- 3 провода питания
- Стяжки пластиковые

2.6 Описание работы Adalight Arduino

2.6.1 Общее описание

Свободное ПО Prismatik при помощи API операционной системы перехватывает сигнал исходящий из видеокарты и передаёт в усовершенствованный и переработанный модуль Lightpack, который транслирует байты через USB на Arduino. Процессор ATmega328P анализирует следующие данные:

- Номер выбранного монитора
- Данные из драйвера мониитора
  - Размер
  - Разрешение экрана
- Количество светодиодов на ленте
- Расположение светодиодов
- Параметры яркости
- Предлагаемые начальные цвета светодиодов

После этого процессор ATmega328P производит сопоставление данных и потоком через массив передаёт на светодиодную RGB ленту последовательный сигнал.

2.6.2 Светодиодная лента WS2812

Каждый метр ленты состоит из 60 SMD светодиодов типоразмера 5050 WS2812 со встроенным в каждый диод трехканальным *ШИМ-контроллером.* Отличие WS2812 от WS2811, с которыми их иногда путают, заключается собственно в наличии светодиодов (WS2811 — отдельная микросхема контроллера, WS2812 — светодиоды со встроенным контроллером). Описания двух протоколов управления:

На ленте всего три шины: земля, питание и управление. С первыми двумя все очевидно из названия (требуется стабилизированный источник напряжения +5V), а управление работает так:

В течение первых 50 миллисекунд происходит инициализация ленты путем заземления управляющей шины. После этого контроллером отправляется пачка из пакетов по 24 бита (8 бит на каждый цветовой канал), содержащих информацию о яркости, каждый пакет предназначается для одного диода. Данные в пачке ничем не разделяются, т.е. каждый следующий пакет идет непосредственно за предыдущим. Получив всю пачку (длина пачки равна 24 бита X количество диодов в ленте), контроллер первого диода «откусывает» от нее свой пакет, использует информацию по назначению, а остальную пачку ретранслирует вперед. Таким образом, до последнего диода доходит пачка из одного 24-битного пакета.

2.6.3 Описание работы ШИМ-контроллера

Раньше для питания устройств использовали схему с понижающим (или повышающим, или многообмоточным) трансформатором, диодным мостом, фильтром для сглаживания пульсаций. Для стабилизации использовались линейные схемы на параметрических или интегральных стабилизаторах. Главным недостатком был низкий КПД и большой вес и габариты мощных блоков питания.

Во всех современных бытовых электроприборах используются импульсные блоки питания (ИБП, ИИП – одно и то же). В большинстве таких блоков питания в качестве основного управляющего элемента используют ШИМ-контроллер.

ШИМ-контроллер – это устройство, которое содержит в себе ряд схемотехнических решений для управления силовыми ключами. При этом управление происходит на основании информации полученной по цепям обратной связи по току или напряжению – это нужно для стабилизации выходных параметров.

Аббревиатура «ШИМ» расшифровывается, как широтно-импульсная модуляция – это один из методов модуляции сигнала не за счёт величины выходного напряжения, а именно за счёт изменения ширины импульсов. В результате формируется моделируемый сигнал за счёт интегрирования импульсов с помощью С- или LC-цепей, другими словами – за счёт сглаживания.

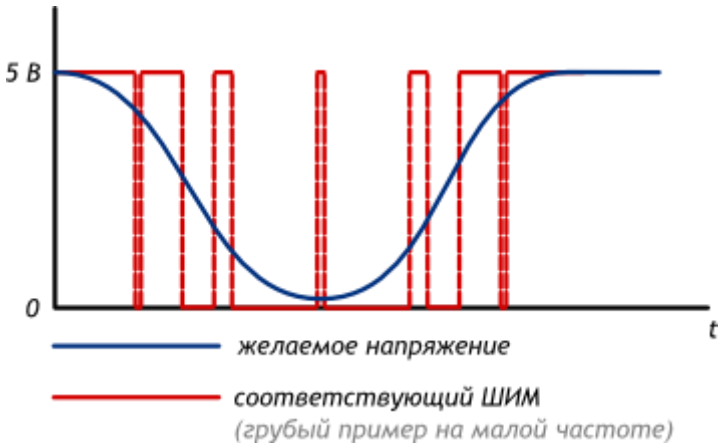

Рис 5. Изменение напряжения при ШИМ-контроллере

Иногда, ШИМ-контроллерами называются генераторы ШИМ-импульсов, но в них нет возможности подключить цепи обратной связи, и они подходят скорее для регуляторов напряжения, чем для обеспечения стабильного питания приборов. Однако в литературе и интернет-порталах часто можно встретить названия типа «ШИМ-контроллер, на NE555» или « на мікроконтроллерах Arduino» - это не совсем верно по вышеуказанным причинам, они могут использоваться только для регулирования выходных параметров, но не для их стабилизации.

2.6.4 Итоговая схема работы

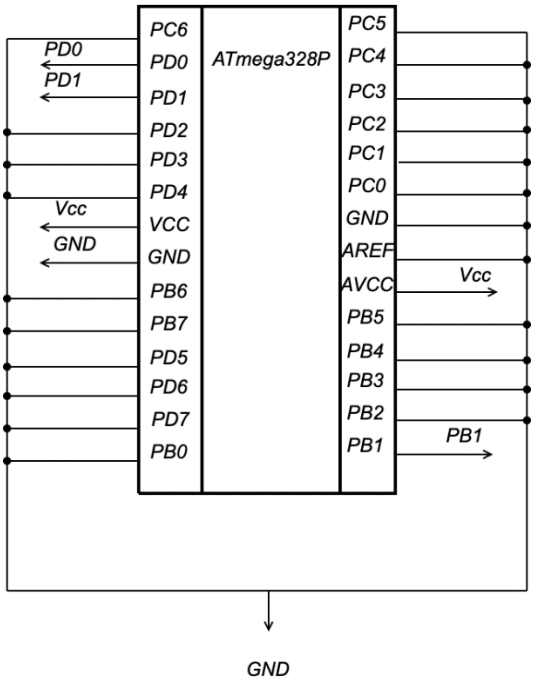

Рис 6. Принципиальная схема микроконтроллера ATmega328P

Общая логика работы такова: инициализируется лента (задается тип контроллера, управляющий pin Arduino (в курсовой работе – это D13),

количество диодов (максимум из-за ограничений Arduino - 255) и их максимальная яркость), задаётся массив из восьмибитных значений яркости каждого цвета для каждого диода и даётся команда на отображение данных, после чего происходит посылка управляющей пачки на ленту, и диоды меняют свой цвет. С цифрового выхода PB1 сигнал подаётся на вход DIN через резистор на 200 Ом. Один из выходов GND соединен с входом GND на светодиодной ленте для корректного управления лентой.

2.7 Системные требования и компоненты для работы

Для работы на системах Windows

Никакого дополнительного ПО не требуется, всё работает исключительно через WIN API. Однако рекомендуется установить последние обновления безопасности.

Для работы на системах Linux

Уникальность и разнообразие UNIX систем – и беда, и счастье. Поэтому перед корректной работой следует проверить установлены ли следующие компоненты:

- qt5-default
- libqt5serialport5-dev
- build-essential
- libgtk2.0-dev
- libusb-1.0-0-dev
- libnotify-dev
- libudev-dev
- qttools5-dev-tools
- Если вы используете Ubuntu: libappindicator-dev
- Не обязательно, но следует проверить обновление следующего SSL сокета: openssl

Для работы на системах OS X

- Qt SDK (5.0+)
- MacOSX 10.9.sdk
- QtCore.framework
- QtGui.framework
- QtNetwork.framework
- QtOpenGL.framework

3. Сравнение с существующими аналогами

3.1 Коммерческие решения

Возможности	Adalight Arduino	Lightpack UHD	Philips Ambilight
Работает на ТВ	Нет	Да	Нет
Работает на любом ПК	С ограничениями	Да	Да
Работает на любом HDMI порте	Да	Да	Да
4 HDMI выхода для всех видов девайсов	Нет	Да	Нет
Поддержка разрешения 4К	Да	Да	Да
LED лента для любых размерво мониторов	Да	Да	Нет
Простая установка	Да	Да	Да
Поддержка 30 градусной изогнутости экрана	Да	Да	Да
Управление с смартфона	Нет	Да	Нет
Управление с ПК	Да	Да	Да
Работа в Flash режиме	Да	Да	Да
Удалённый контроль	Нет	Да	Нет
Тепловыделение	Среднее	Среднее	Низкое
Высокая скорость работы (не более 50 миллисекунд)	Да	Да	Да
Цена	Приблизительно 110 Бел. рублей	600 Бел. рублей	От 1000 Бел. рублей

Таблица 2. Сравнение курсового проекта с коммерческим решение «Lightpack UHD» и Philips «Ambilight».

Как мы видим, Adalight Arduino уступает коммерческому решению Lightpack UHD, однако она превосходит решение от основателя этой технологии, компании Philips. Так же из таблицы видно, что в ценовом сегменте, Adalight выходит несомненным победителем.

3.2 Некоммерческие решения

Среди некоммерческих решений тяжело провести глубокое сравнение. Однако безусловным плюсом для проекта, была бы замена микроконтроллера Arduino на маломощный микрокомпьютер Raspberry Pi 3 и работу через HDMI сигнал и драйвер монитора. Это позволило сократить энергопотребление и теплоотдачу, увеличить скорость обработки данных до 10 миллисекунд и более качественно сопоставлять цвет светодиодов с изображением на экране. Так же это позволило работать на Любых мониторах, а не только ПК.

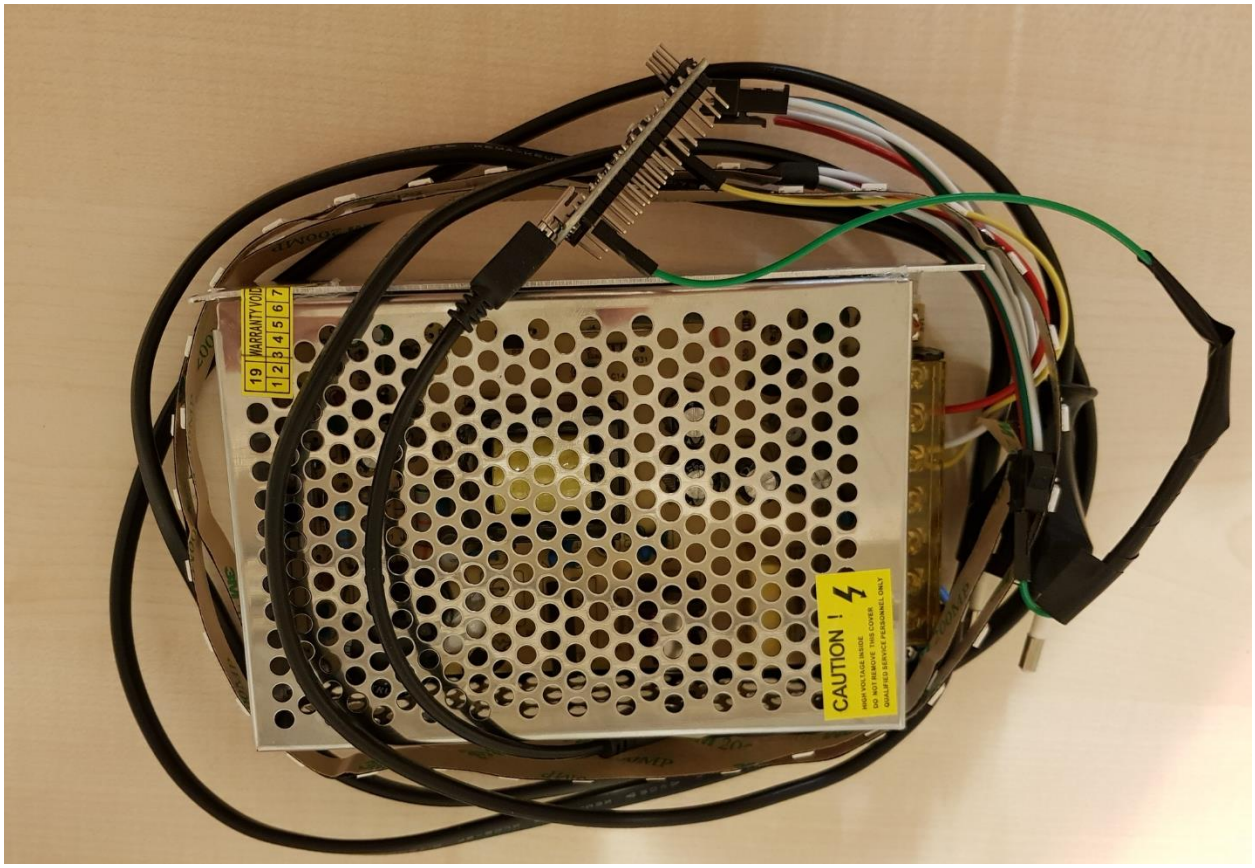

Рисунок 7. «Adalight Arduino» в сложенном виде на базе микроконтроллера Arduino Nano.

Рисунок 8. Итоговый результат работы «Adalight Arduino».

Заключение

В результате курсового проекта было разработано устройство на платформе Arduino представляющее собой полноценную замену коммерческим решениям в области динамической адаптивной подсветки монитора с совместимостью на любом компьютере.

Тестирование устройства продемонстрировало удобство, большой функционал и полезность использования. К минусам продукта можно отнести большое энергопотребление и среднею теплоотдачу. Однако большим плюсом является цена производства этого продукта. Анализируя все плюсы и минусы, данное устройство можно с успехом использовать на персональных компьютерах и мониторах, особенно если учесть весьма полезное научно доказанное влияние на глаза человека.

Список использованной литературы

1. Программирование микроконтроллерных плат Arduino/Freeduino. Улли Соммер. БХВ-Петербург, ISBN 978-5-9775-0727-1, 2012 – 172 с

2. Патент Philips. Display apparatus. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://patents.google.com/patent/EP1379082B1

3. Патент Philips. Ambilight displaying arrangement. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://patents.google.com/patent/US20100091193A1

4. Программа Lightpack. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://github.com/psieg/Lightpack

5. Спецификация ленты WS2812. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

6. Спецификация микроконтроллера ATmega328P. [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.sparkfun.com/datasheets/Components/SMD/ATMega328.pdf

7. ШИМ-контроллер. Спецификация. Устрйоство. [Электронный ресурс]. – Электронные данные. – Режим доступа: http://electrik.info/main/praktika/1441-shim-kontroller-vidy-i-shemy.html

Приложение 1. Текст программы

Код для микроконтроллера Arduino.

```
include <FastLED.h>
#define colorOrder GRB

#define NUM_LEDS 60
#define PIN 13
#define serialRate 115200
uint8_t prefix[] = {'A', 'd', 'a'}, hi, lo, chk, i;
CRGB leds[NUM_LEDS];

void setup()
{
 pinMode(13, OUTPUT);
 digitalWrite(13, LOW);
 FastLED.addLeds<WS2812, PIN, colorOrder>(leds, NUM_LEDS);
 byte del=2;
 // initial RGB flash
 for(i=0;i<255;i++) {
 LEDS.showColor(CRGB(i, 0, 0));
 delay(del);
 }
 for(i=1;i<255;i++) {
 LEDS.showColor(CRGB(255, i, 0));
 delay(del);
 }
 for(i=255;i>1;i--) {
 LEDS.showColor(CRGB(i, 255, 0));
 delay(del);
 }
 for(i=1;i<255;i++) {
 LEDS.showColor(CRGB(0, 255, i));
 delay(del);
 }
 for(i=255;i>1;i--) {
 LEDS.showColor(CRGB(0, i, 255));
 delay(del);
 }
 for(i=0;i<255;i++) {
 LEDS.showColor(CRGB(i, 0, 255));
 delay(del);
 }
```

```
 for(i=255;i>1;i--) {
 LEDS.showColor(CRGB(255, 0, i));
 delay(del);
 }
for(i=255;i>1;i--) {
 LEDS.showColor(CRGB(i, 0, 0));
 delay(del);
}
 Serial.begin(serialRate);
 Serial.print("Ada\n");
}

void loop() {
 // wait for first byte of Magic Word
 for(i = 0; i < sizeof prefix; ++i) {
   waitLoop: while (!Serial.available()) ;;
   // Check next byte in Magic Word
   if(prefix[i] == Serial.read()) continue;
   // otherwise, start over
   i = 0;
   goto waitLoop;
 }
 while (!Serial.available()) ;;
 hi=Serial.read();
 while (!Serial.available()) ;;
 lo=Serial.read();
 while (!Serial.available()) ;;
 chk=Serial.read();

 if (chk != (hi ^ lo ^ 0x55))
 {
   i=0;
   goto waitLoop;
 }

 memset(leds, 0, NUM_LEDS * sizeof(struct CRGB));
 // read the transmission data and set LED values
 for (uint8_t i = 0; i < NUM_LEDS; i++) {
   byte r, g, b;
   while(!Serial.available());
   r = Serial.read();
   while(!Serial.available());
   g = Serial.read();
   while(!Serial.available());
```

```
    b = Serial.read();
    leds[i].r = r;
    leds[i].g = g;
    leds[i].b = b;
   }
 FastLED.show();
}
```

Необходимый FastLED.h и его FastLED.cpp

```
#define FASTLED_INTERNAL
#include "FastLED.h"


#if defined(__SAM3X8E__)
volatile uint32_t fuckit;
#endif

FASTLED_NAMESPACE_BEGIN

void *pSmartMatrix = NULL;

CFastLED FastLED;

CLEDController *CLEDController::m_pHead = NULL;
CLEDController *CLEDController::m_pTail = NULL;
static uint32_t lastshow = 0;

uint32_t _frame_cnt=0;
uint32_t _retry_cnt=0;

// uint32_t CRGB::Squant = ((uint32_t)((__TIME__[4]-'0') * 28))<<16 |
((__TIME__[6]-'0')*50)<<8 | ((__TIME__[7]-'0')*28);

CFastLED::CFastLED() {
        // clear out the array of led controllers
        // m_nControllers = 0;
        m_Scale = 255;
        m_nFPS = 0;
        m_pPowerFunc = NULL;
        m_nPowerData = 0xFFFFFFFF;
}

CLEDController &CFastLED::addLeds(CLEDController *pLed,
```

```
                                                        struct CRGB
*data,
                                                        int
nLedsOrOffset, int nLedsIfOffset) {
        int nOffset = (nLedsIfOffset > 0) ? nLedsOrOffset : 0;
        int nLeds = (nLedsIfOffset > 0) ? nLedsIfOffset : nLedsOrOffset;

        pLed->init();
        pLed->setLeds(data + nOffset, nLeds);
        FastLED.setMaxRefreshRate(pLed->getMaxRefreshRate(),true);
        return *pLed;
}

void CFastLED::show(uint8_t scale) {
        // guard against showing too rapidly
        while(m_nMinMicros && ((micros()-lastshow) < m_nMinMicros));
        lastshow = micros();

        // If we have a function for computing power, use it!
        if(m_pPowerFunc) {
                scale = (*m_pPowerFunc)(scale, m_nPowerData);
        }

        CLEDController *pCur = CLEDController::head();
        while(pCur) {
                uint8_t d = pCur->getDither();
                if(m_nFPS < 100) { pCur->setDither(0); }
                pCur->showLeds(scale);
                pCur->setDither(d);
                pCur = pCur->next();
        }
        countFPS();
}

int CFastLED::count() {
   int x = 0;
        CLEDController *pCur = CLEDController::head();
        while( pCur) {
       x++;
                pCur = pCur->next();
        }
   return x;
}
```

```
CLEDController & CFastLED::operator[](int x) {
	CLEDController *pCur = CLEDController::head();
	while(x-- && pCur) {
		pCur = pCur->next();
	}
	if(pCur == NULL) {
		return *(CLEDController::head());
	} else {
		return *pCur;
	}
}

void CFastLED::showColor(const struct CRGB & color, uint8_t scale) {
	while(m_nMinMicros && ((micros()-lastshow) < m_nMinMicros));
	lastshow = micros();

	// If we have a function for computing power, use it!
	if(m_pPowerFunc) {
		scale = (*m_pPowerFunc)(scale, m_nPowerData);
	}

	CLEDController *pCur = CLEDController::head();
	while(pCur) {
		uint8_t d = pCur->getDither();
		if(m_nFPS < 100) { pCur->setDither(0); }
		pCur->showColor(color, scale);
		pCur->setDither(d);
		pCur = pCur->next();
	}
	countFPS();
}

void CFastLED::clear(bool writeData) {
	if(writeData) {
		showColor(CRGB(0,0,0), 0);
	}
   clearData();
}

void CFastLED::clearData() {
	CLEDController *pCur = CLEDController::head();
	while(pCur) {
		pCur->clearLedData();
		pCur = pCur->next();
```

```
        }
}

void CFastLED::delay(unsigned long ms) {
        unsigned long start = millis();
      do {
#ifndef FASTLED_ACCURATE_CLOCK
                // make sure to allow at least one ms to pass to ensure the clock
moves
                // forward
                ::delay(1);
#endif
                show();
                yield();
        }
        while((millis()-start) < ms);
}

void CFastLED::setTemperature(const struct CRGB & temp) {
        CLEDController *pCur = CLEDController::head();
        while(pCur) {
                pCur->setTemperature(temp);
                pCur = pCur->next();
        }
}

void CFastLED::setCorrection(const struct CRGB & correction) {
        CLEDController *pCur = CLEDController::head();
        while(pCur) {
                pCur->setCorrection(correction);
                pCur = pCur->next();
        }
}

void CFastLED::setDither(uint8_t ditherMode)  {
        CLEDController *pCur = CLEDController::head();
        while(pCur) {
                pCur->setDither(ditherMode);
                pCur = pCur->next();
        }
}

//
```

```
// template<int m, int n> void transpose8(unsigned char A[8], unsigned char
B[8]) {
//      uint32_t x, y, t;
//
//      // Load the array and pack it into x and y.
//      y = *(unsigned int*)(A);
//      x = *(unsigned int*)(A+4);
//
//      // x = (A[0]<<24)   | (A[m]<<16)   | (A[2*m]<<8) | A[3*m];
//      // y = (A[4*m]<<24) | (A[5*m]<<16) | (A[6*m]<<8) | A[7*m];
//
      // // pre-transform x
      // t = (x ^ (x >> 7)) & 0x00AA00AA;  x = x ^ t ^ (t << 7);
      // t = (x ^ (x >>14)) & 0x0000CCCC;  x = x ^ t ^ (t <<14);
                              //
      // // pre-transform y
      // t = (y ^ (y >> 7)) & 0x00AA00AA;  y = y ^ t ^ (t << 7);
      // t = (y ^ (y >>14)) & 0x0000CCCC;  y = y ^ t ^ (t <<14);
                              //
      // // final transform
      // t = (x & 0xF0F0F0F0) | ((y >> 4) & 0x0F0F0F0F);
      // y = ((x << 4) & 0xF0F0F0F0) | (y & 0x0F0F0F0F);
      // x = t;
//
//      B[7*n] = y; y >>= 8;
//      B[6*n] = y; y >>= 8;
//      B[5*n] = y; y >>= 8;
//      B[4*n] = y;
//
//   B[3*n] = x; x >>= 8;
//      B[2*n] = x; x >>= 8;
//      B[n] = x; x >>= 8;
//      B[0] = x;
//      // B[0]=x>>24;    B[n]=x>>16;    B[2*n]=x>>8;  B[3*n]=x>>0;
//      // B[4*n]=y>>24;  B[5*n]=y>>16;  B[6*n]=y>>8;  B[7*n]=y>>0;
// }
//
// void transposeLines(Lines & out, Lines & in) {
//      transpose8<1,2>(in.bytes, out.bytes);
//      transpose8<1,2>(in.bytes + 8, out.bytes + 1);
// }

extern int noise_min;
extern int noise_max;
```

```
void CFastLED::countFPS(int nFrames) {
  static int br = 0;
  static uint32_t lastframe = 0; // millis();

  if(br++ >= nFrames) {
                uint32_t now = millis();
                now -= lastframe;
                m_nFPS = (br * 1000) / now;
    br = 0;
    lastframe = millis();
  }
}

void CFastLED::setMaxRefreshRate(uint16_t refresh, bool constrain) {
  if(constrain) {
    // if we're constraining, the new value of m_nMinMicros _must_ be
higher than previously (because we're only
    // allowed to slow things down if constraining)
    if(refresh > 0) {
      m_nMinMicros = ( (1000000/refresh) >  m_nMinMicros) ?
(1000000/refresh) : m_nMinMicros;
    }
  } else if(refresh > 0) {
    m_nMinMicros = 1000000 / refresh;
  } else {
    m_nMinMicros = 0;
  }
}

extern "C" int atexit(void (* /*func*/ )()) { return 0; }

#ifdef FASTLED_NEEDS_YIELD
extern "C" void yield(void) { }
#endif

#ifdef NEED_CXX_BITS
namespace __cxxabiv1
{
        #if !defined(ESP8266) && !defined(ESP32)
        extern "C" void __cxa_pure_virtual (void) {}
        #endif

        /* guard variables */
```

```
        /* The ABI requires a 64-bit type.  */
        __extension__ typedef int __guard __attribute__((mode(__DI__)));

        extern "C" int __cxa_guard_acquire (__guard *)
__attribute__((weak));
        extern "C" void __cxa_guard_release (__guard *)
__attribute__((weak));
        extern "C" void __cxa_guard_abort (__guard *)
__attribute__((weak));

        extern "C" int __cxa_guard_acquire (__guard *g)
        {
                return !*(char *)(g);
        }

        extern "C" void __cxa_guard_release (__guard *g)
        {
                *(char *)g = 1;
        }

        extern "C" void __cxa_guard_abort (__guard *)
        {

        }
}
#endif

FASTLED_NAMESPACE_END
```

Модифицированный Lightpack config.

```
[General]
MainConfigVersion=4.0
ProfileLast=Lightpack
Language=<System>
DebugLevel=0
IsExpertModeEnabled=true
IsKeepLightsOnAfterExit=false
IsKeepLightsOnAfterLock=false
IsPingDeviceEverySecond=true
IsUpdateFirmwareMessageShown=false
ConnectedDevice=Adalight
SupportedDevices="Lightpack,Adalight,Ardulight,Virtual"
```

```
CheckForUpdates=true
InstallForUpdates=true
IsKeepLightsOnAfterSuspend=false

[API]
IsEnabled=true
ListenOnlyOnLoInterface=false
Port=3636
AuthKey=

[Adalight]
SerialPort=COM4
BaudRate=115200
NumberOfLeds=60
ColorSequence=RGB

[Ardulight]
SerialPort=COM3
BaudRate=115200
NumberOfLeds=60
ColorSequence=GRB

[AlienFx]
NumberOfLeds=1

[Lightpack]
NumberOfLeds=10

[Virtual]
NumberOfLeds=60
```