



Опыт сообщества в полной мере

Светодиодные проекты на Arduino

Получите максимум от вашего Arduino, чтобы разрабатывать увлекательные и креативные светодиодные проекты

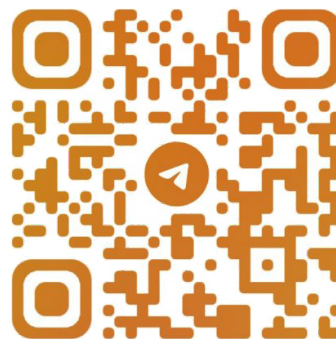
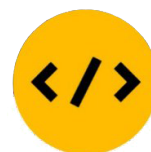
Самарт Шах Утсав Шах

Светодиодные проекты на Arduino

Получите максимум от вашего Arduino, чтобы
разрабатывать увлекательные
и креативные светодиодные проекты

Самарт Шах

Утсав Шах



@CODELIBRARY_IT

Об авторах

Самарт Шах - инженер-программист по профессии и создатель. Он руководит производственной деятельностью в Pune Makers и возглавляет клуб Infosys Robotics Club. Он любит создавать творческие / инновационные прототипы с использованием новейшего оборудования / датчиков (Raspberry Pi, Arduino, Kinect, Leap Motion и многие другие) и программного обеспечения. Он выступал с докладами на различных национальных и международных конференциях. Он написал книгу о Raspberry Pi под названием Learning Raspberry Pi, Packt Publishing. В течение дня он работает над различными методами визуализации данных и фреймворками пользовательского интерфейса. По ночам он ведет блог, читает, пишет и многое другое. Вы можете узнать больше о нем на <http://samarthshah.com>.

Утсав Шах - инженер по КИП, который любит работать как над новейшим оборудованием, так и над программными технологиями. Он был представлен на ведущем веб-сайте Индии <http://yourstory.in> and Ahmedabad Mirror (Times Group) за его исследовательскую работу по «Преобразованию языка жестов в речь» с использованием контроллера Leap Motioncontroller. Помимо своей постоянной работы в Infosys Limited, он руководит деятельностью Infosys Robotics Club. В свободное время он любит читать книги и работать над передовыми технологиями.

Оглавление

Предисловие	v
Глава 1: Начало работы с Arduino и светодиодами	1
Платы Arduino	1
Разные платы Arduino	2
Как выбрать плату Arduino для вашего проекта	3
Arduino UNO	5
Arduino IDE	7
Установка Arduino IDE	7
В Windows	7
Понимание Arduino IDE	8
Перед тем как начать	9
Блок питания	9
Проверка соединения	10
"Hello World"	11
Использование последовательной связи	13
Последовательная запись	13
Последовательное чтение	14
Мир светодиодов	16
Резюме	17
Глава 2: Проект 1 - Светодиодный ночник	19
Введение в макетную плату	19
Состав макетной платы	20
Использование макетной платы	20
Управление несколькими светодиодами	21
Простой контроллер светофора	21

LED затухание	23
Широтно-импульсная модуляция (ШИМ)	23
Использование ШИМ на Arduino	24
Создание лампы настроения	25
Использование светодиода RGB	26
Почему светодиоды RGB меняют цвет?	26
Создание лампы настроения	27
Разработка светодиодного ночника	31
Знакомство с кнопкой	31
Лампа Pixar	32
Резюме	35
Глава 3: Проект 2 - Подсветка телевизора с ДУ	37
Знакомство с ИК-светодиодами	37
Что такое ИК-светодиод?	38
Применение ИК-светодиодов / ИК-связи	38
ИК-датчики	38
Рабочий механизм	38
Программирование базового ИК-датчика	39
Как получать данные с пульта от телевизора	41
Светодиодные ленты	45
Управление светодиодной лентой с помощью Arduino	46
Резюме	55
Глава 4: Проект 3 - Светодиодный куб	57
Начало работы с пайкой	57
Что вам понадобится	57
Советы по безопасности	58
Изготовление светодиодного куба	59
Необходимые компоненты	59
Принцип, лежащий в основе дизайна	60
Изготовление	61
Ошибки, которые следует избегать	64
Крепление к доске	65
Программирование светодиодного куба 4 * 4 * 4	68
Резюме	71
Глава 5: Звуковая визуализация и светодиодная елка	73
Введение в звуковую визуализацию	74
Как визуализировать звук	74
Что такое БПФ (быстрое преобразование Фурье)	75

Звуковая визуализация с использованием Arduino	76
Изготовление светодиодной новогодней елки	85
Резюме	90
Глава 6: Постоянство зрения	91
<hr/>	
Создайте свое собственное постоянство видения	91
Программирование светодиодной матрицы	93
Разные типы двигателей	94
Двигатели постоянного тока	95
Серводвигатели	95
Шаговые двигатели	95
Различные применения двигателей	96
Управление двигателем DC с помощью Arduino	96
Синхронизация светодиодной матрицы с двигателем	100
Воплотите свои усилия в жизнь	103
Вращение руками	103
Использование двух разных Arduinos или внешних двигателей	104
Используйте существующие реальные устройства	104
Резюме	105
Глава 7. Устранение неполадок и дополнительные ресурсы	107
<hr/>	
Устранение неисправностей	107
Не могу загрузить программу	107
Тусклый светодиод	109
Ресурсы - опытные пользователи	109
Проекты	109
Twitter Mood Light	109
Дверной замок с секретным детектором	110
Светодиодная велосипедная куртка	110
Кофейник с поддержкой Twitter	110
Полезные ресурсы	111
Hackaday	111
Блог Arduino	111
Журнал Make	111
Bildr	111
Учебные пособия	112
Tronixstuff	112
Adafruit	112
Все о схемах	112
Хакерские пространства	112
Форум Arduino	113
Резюме	113

Предисловие

Arduino - это платформа для создания прототипов с открытым исходным кодом, основанная на простом в использовании аппаратном и программном обеспечении. Arduino использовался в тысячах различных проектов и приложений широким кругом программистов и художников, и их вклад позволил получить невероятное количество информации, которая может быть большим подспорьем и для новичков и экспертов.

Эта книга станет вашим спутником, чтобы раскрыть в вас творческие способности. По мере прохождения книги вы узнаете, как разрабатывать различные проекты с помощью Arduino.

О чем эта книга

Глава 1, «**Начало работы с Arduino и светодиодами**», знакомит вас с различными платами Arduino, за которыми следуют инструкции по установке для Arduino IDE. Вы напишете программу «Hello World» с помощью Arduino IDE и узнаете о последовательной связи.

Глава 2, «**Проект 1 - Светодиодный ночник**», представляет вам некоторые интересные вещи по управлению светодиодами и покажет, как управлять различными светодиодами с художественным подходом.

Глава 3, «**Проект 2 - Подсветка телевизора с дистанционным управлением**», обучает основам работы с ИК-светодиодами и основам ИК-связи. Узнав о программировании ИК-датчика, вы будете использовать ИК-датчик для управления подсветкой телевизора с помощью пульта дистанционного управления.

Глава 4, «**Проект 3 - Светодиодный куб**», подробно знакомит вас с пайкой. Мы также создадим светодиодный куб $4 * 4 * 4$ с помощью платы Arduino UNO.

В главе 5 «**Визуализация звука и светодиодная рождественская елка**» показано, как визуализировать звук с помощью Arduino, а затем мы разработаем светодиодную рождественскую елку.

Глава 6 «**Постоянство зрения**» ознакомит вас с визуализацией информации в пространстве с помощью светодиодов. Вы создадите жезл ,используя светодиодную матрицу и двигатель.

Глава 7, «**Устранение неполадок и дополнительные ресурсы**», начинается с общих методов устранения неполадок. Во второй и последней части главы обсуждаются ресурсы, которые могут быть полезны, если вы хотите выполнять сложные задачи с помощью *Arduino*.

Что вам понадобится для этой книги

Все, что вам нужно, - это Arduino Uno и энтузиазм для работы над интересными проектами.

Для кого эта книга

Любой, у кого есть базовые знания компьютера, должен получить от этой книги максимум пользы. Хотя знакомство с базовой электроникой было бы полезно, но это не обязательно.

1

Начало работы с Arduino и светодиодами

Добро пожаловать в захватывающий мир физических вычислений! Сегодня любители и эксперты во всем мире используют Arduino для создания интерактивных объектов и создания крутых прототипов. В этой главе вы познакомитесь с различными платами Arduino, а затем получите инструкции по установке Arduino IDE. Вы напишете программу HelloWorld, используя Arduino IDE, и узнаете о последовательной связи. К концу этой главы вы получите базовые знания об Arduino и ее IDE, которые пригодятся в оставшихся главах книги. В этой главе мы рассмотрим следующее:

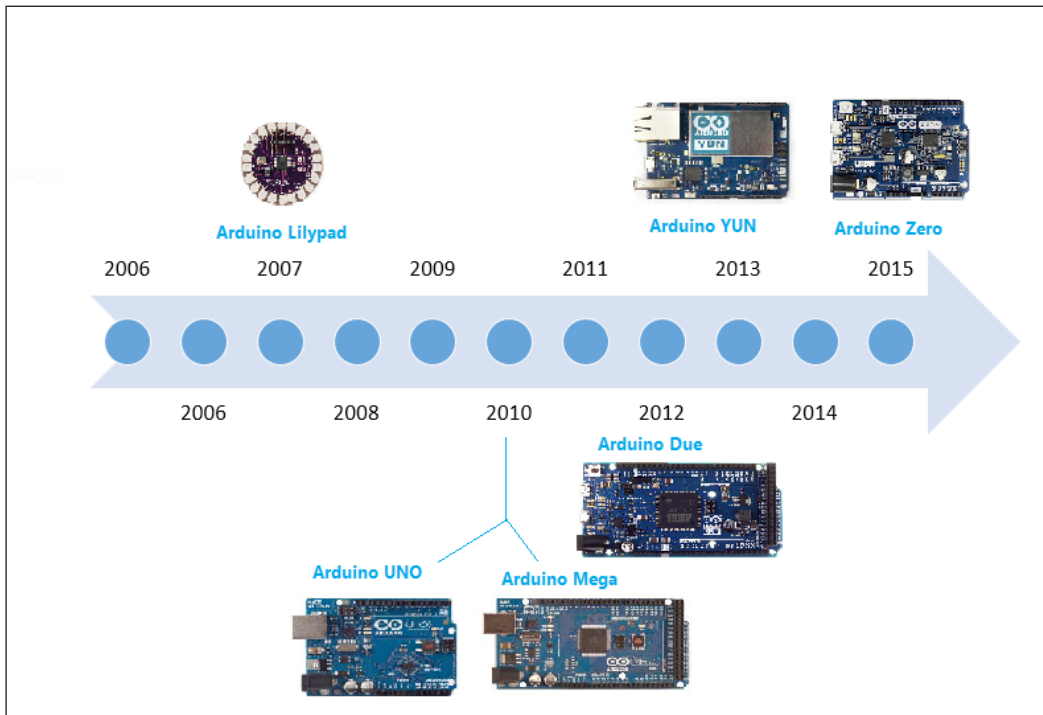
- Платы Arduino
- Arduino IDE
- Платы Arduino
- "Hello World"
- Использование последовательной связи
- Мир светодиодов

Платы Arduino

Arduino изначально создавался для художников и дизайнеров как простой и быстрый инструмент для создания прототипов. Дизайнеры могли создавать сложные устройства и произведения искусства, даже не обладая знаниями в области электроники и программирования. Итак, понятно, что первые несколько шагов изучения Arduino очень просты. В этом разделе вы познакомитесь с различными платами Arduino и узнаете, как выбрать плату Arduino для своего проекта, а также получите некоторую информацию о плате Arduino UNO, которую вы будете использовать в этой книге.

Различные платы Arduino

Новички часто путаются, когда открывают для себя проекты Arduino. Когда они ищут Arduino, они слышат и читают такие термины, как Uno, Zero и Lilypad. Дело в том, что нет такого понятия, как «Ардуино». В 2006 году команда Arduino спроектировала и разработала плату микроконтроллера и выпустила ее по лицензии с открытым исходным кодом. Эти версии в основном имели итальянские названия. Вот количество плат, которые команда разработала за последние 10 лет:



Команда Arduino не просто улучшила дизайн, они изобрели новые конструкции для конкретных случаев использования. Например, они разработали Arduino Lilypad для встраивания платы в текстиль. Из нее можно создавать интерактивные футболки. В следующей таблице показаны возможности различных плат Arduino. Платы Arduino могут отличаться по внешнему виду, но у них много общего. Вы можете использовать те же инструменты и библиотеки для программирования:

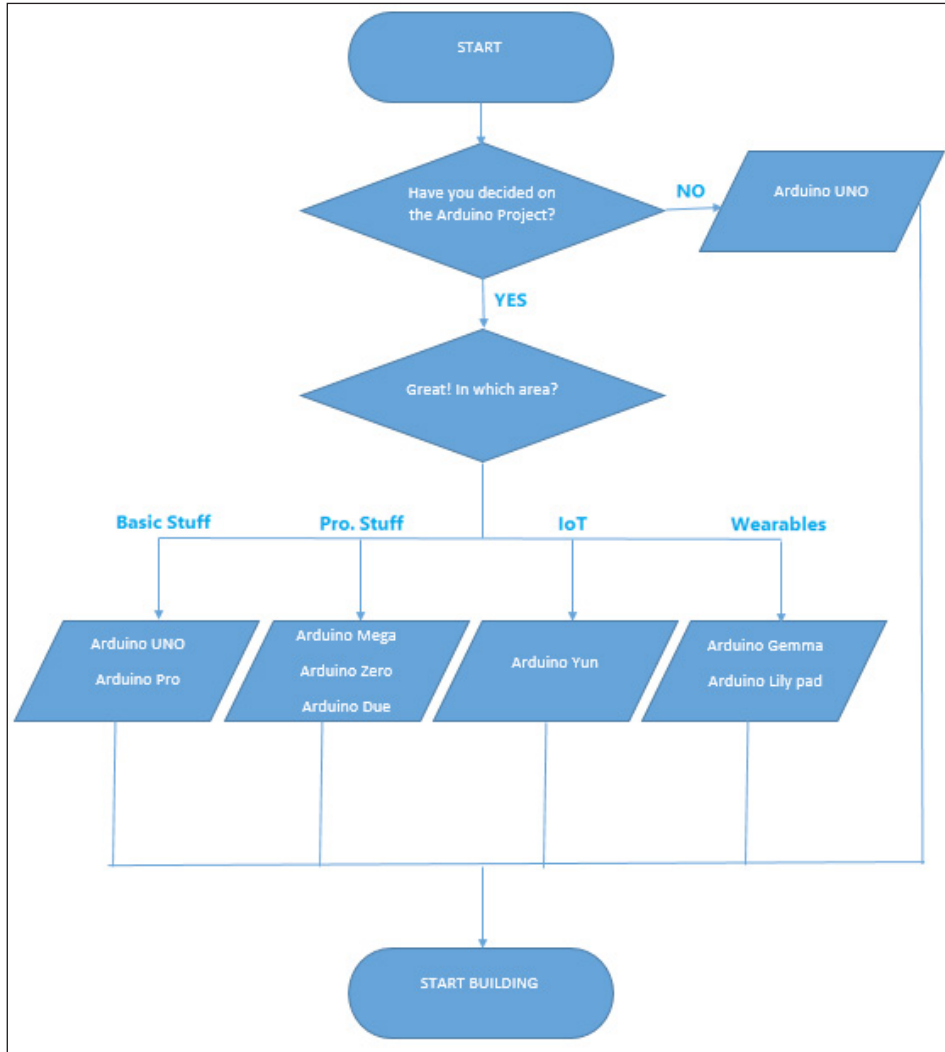
Name	Processor	Dimension	Voltage	Flash(kB)	Digital I/O(PWM) pins	Analog input pins
Arduino Lilypad	ATmega168V	51 mm outer diameter	2.7-5.5 V	16	14(6)	6
Arduino YUN	Atmega32U4	68.6 mm × 53.3 mm	5 V	32	14(6)	12
Arduino Mega	ATmega2560	101.6 mm × 53.3 mm	5 V	256	54(15)	16
Arduino Due	ATSAM3X8E	101.6 mm × 53.3 mm	3.3 V	512	54(12)	12
Arduino Zero	ATSAMD21G18A	68.6 mm × 53.3 mm	3.3 V	256	14(12)	6
Arduino UNO	ATmega328P	68.6 mm × 53.3 mm	5 V	32	14(6)	6

Дизайн и схемы плат AsArduino имеют открытый исходный код, любой может использовать и изменять исходный дизайн платы, а также может создать свою собственную версию платы, совместимой с Arduino. Из-за этого вы можете найти в сети бесчисленное количество клонов Arduino, которые можно запрограммировать с использованием тех же инструментов и библиотек, что и для оригинальных плат Arduino.

Как выбрать плату Arduino для вашего проекта

При наличии множества опций человеку становится сложно решить, какую плату Arduino использовать для проекта. Семейство Arduino огромно, и невозможно прочитать о каждой плате и решить, какую плату использовать для конкретного проекта.

Следующая блок-схема упрощает процесс, предоставляя сотни решений для широко используемых плат *Arduino* и наиболее распространенных вариантов использования / приложений:



Если вы не уверены, что будете собирать и какие аппаратные возможности требуются, начните создавать свой прототип с помощью *Arduino UNO*. *Arduino UNO* имеет лучшую документацию и лучшую поддержку. Это также самая совместимая из всех плат *Arduino*. Большинство существующих библиотек и шилдов совместимы с *Arduino UNO*.

И, наконец, большая часть кода, написанного на более ранних версиях плат *Arduino*, также будет работать с *Arduino UNO*.

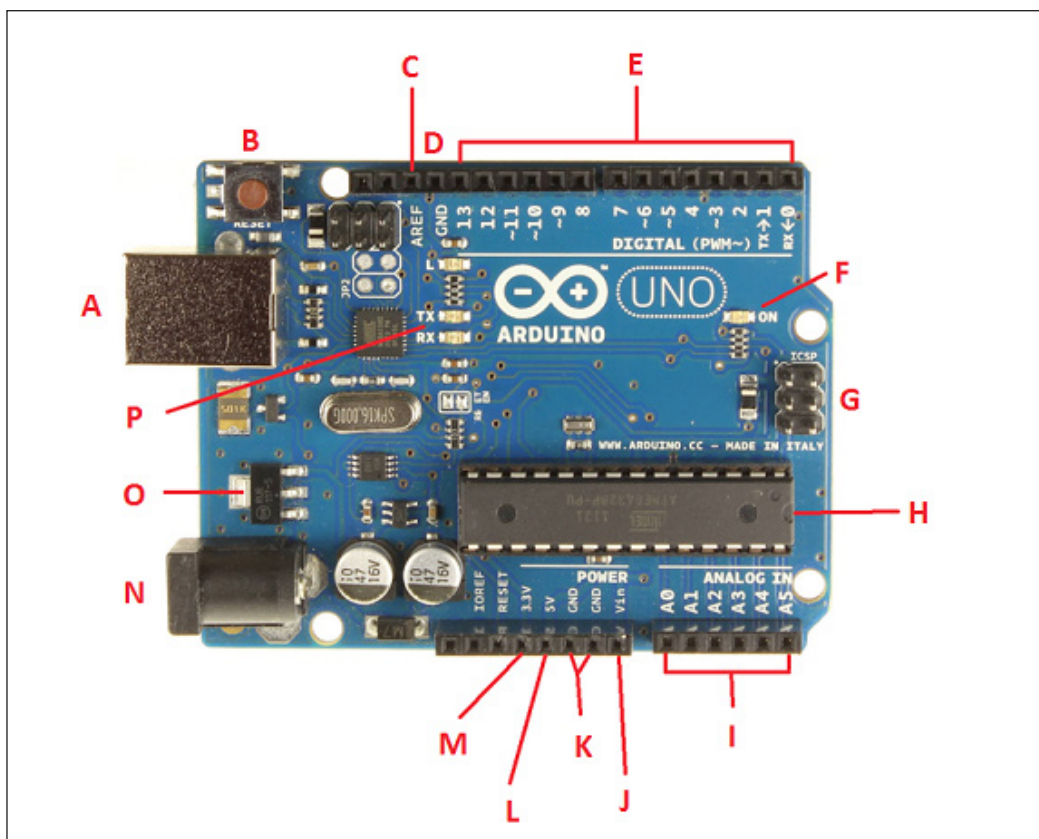


Шилды Arduino - это модульные печатные платы, которые можно установить поверх платы Arduino, что расширяет ее возможности. Хотите подключить Arduino к Интернету? Для этого есть шилд. В сети доступны сотни шилдов, что делает Arduino больше, чем просто платой для разработки.

В этой книге мы будем использовать плату Arduino UNO.

Arduino UNO

«UNO» в переводе с итальянского означает «один», и он был выбран для обозначения выпуска Arduino IDE v1.0. Это первая плата в серии USB-плат Arduino. Как упоминалось ранее, это одна из наиболее широко используемых плат в семействе Arduino:



В этом разделе вы познакомитесь с различными узлами Arduino UNO.

- **A:** USB-штекер. Arduino UNO может питаться от USB-кабеля, идущего от вашего компьютера. USB-соединение также используется для загрузки кода на плату Arduino.
- **B:** кнопка сброса: нажатие на нее временно подключит контакт сброса к земле и перезапустит любой код, загруженный на плату Arduino.
- **C:** **AREF:** Вывод опорного напряжения. В книге не используется.
- **D:** **GND:** цифровая земля.
- **E:** **Пины 0- 13:** ... под меткой DIGITAL являются цифровыми выводами. Используются как для цифрового входа (например, для определения нажатия кнопки), так и для цифрового выхода (например, для включения светодиода). Рядом с некоторыми выводами (3, 5, 6, 9, 10 и 11) стоит знак тильды (~), что означает, что они также могут выдавать широтно-импульсную модуляцию (ШИМ). ШИМ - это метод получения аналогового сигнала цифровыми средствами путем управления длительностью сигнала.
- **F:** **ON:** светодиодный индикатор питания, который загорается при подключении Arduino к источнику питания. Если светодиод не загорается, что-то не так с вашей платой Arduino или с внешним питанием.
- **G:** **In-circuit serial programmer:** Программатор последовательного интерфейса : используется для программирования Arduino с использованием других Arduino или других микроконтроллеров.
- **H:** **Основная микросхема:** это микроконтроллер ATmega 328. Думайте о нем как о мозге вашей платы Arduino.
- **I:** **Пины A0 - A5:** ... являются выводами аналогового входа, которые могут считывать сигнал с аналогового датчика (например, датчика температуры) и преобразовывать его в цифровое значение, которое может быть считано программой Arduino и может использоваться для дальнейшей обработки.
- **J:** **Vin:** Входное напряжение. Arduino может получать питание от разъема постоянного тока (см. **H**), USB-разъема или вывода Vin. Vin может быть до 12В. Встроенная в Arduino микросхема стабилизатора напряжения преобразует Vin в 5В.
- **K:** **GND:** Пин (вывод) земли.

- **L: 5 V:** Вывод питания, который подает 5 вольт.
- **M: 3.3 V:** Вывод питания, подающий 3,3 В.
- **N:** Внешний источник питания.
- **O: Voltage regulator:** Преобразует внешнее питание 7-12В в 5В
- **P: Tx Rx LEDs:** Tx - это передача, RX - прием. В электронике TxRX используется для обозначения выводов, отвечающих за последовательную связь. Эти светодиоды являются индикаторами, когда наш Arduino принимает или передает данные.

Arduino IDE

Поскольку изначально Arduino был разработан для художников и дизайнеров, команда Arduino попыталась разработать программное обеспечение Arduino (IDE) настолько просто, насколько это возможно, без ущерба для мощности инструмента. Прежде чем запускать программу «Hello World», вам необходимо установить Arduino IDE на свой компьютер.

Установка Arduino IDE

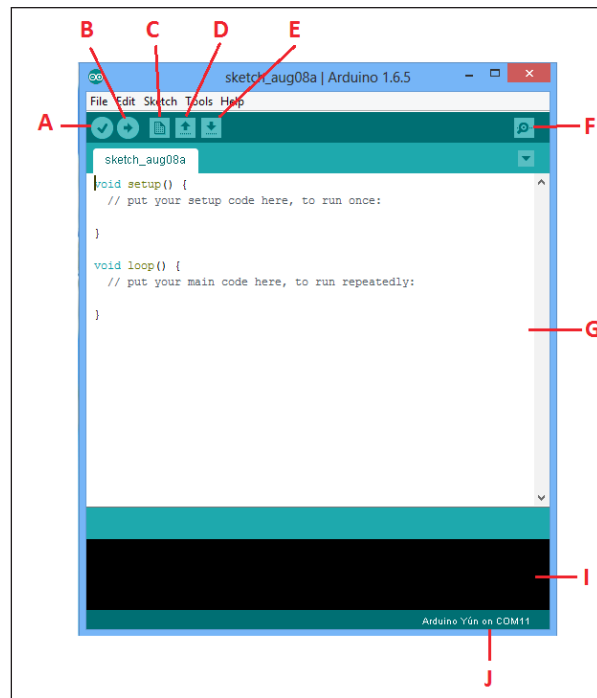
Arduino IDE поддерживается всеми основными операционными системами, изначально Windows, Mac и Linux.

В Windows

1. Перейдите на <https://www.arduino.cc/en/Main/Software>.
2. Загрузите «Установщик Windows» или «ZIP-файл Windows для установки без администратора».
3. Если вы скачали «Установщик Windows», дважды щелкните по нему, и он будет установлен.
4. Если вы скачали «ZIP-файл Windows для установки без администратора», распакуйте его, и вы найдете `arduino.exe`. Дважды щелкните по нему, чтобы начать работу с Arduino IDE.

Разумный IDE

IDE *Arduino* очень проста. В основном она состоит из редактора, компилятора, загрузчика и монитора последовательного порта (МПП). У него нет дополнительных функций, таких как отладчик или автозавершение кода:



Рассмотрим каждую кнопку отдельно:

- Кнопка «Проверить»: ... скомпилирует программу, которая в данный момент находится в редакторе. Она не только проверяет программу синтаксически, а также превращает это в представление, подходящее для платы Arduino.
- **В:** Кнопка загрузки: ... скомпилирует и загрузит текущую программу на подключенную плату Arduino.
- **С:** Новая кнопка: создает новую программу, открывая чистый лист редактора.
- **D:** Кнопка «Открыть»: с помощью этой кнопки вы можете открыть существующую программу из файловой системы.
- **E:** кнопка «Сохранить»: сохраняет текущую программу.
- **F:** Монитор последовательного порта (**МПП**): Arduino может связываться с компьютером через последовательное соединение. При нажатии кнопки Serial Monitor открывается окно последовательного монитора, которое позволяет вам наблюдать за данными, отправляемыми Arduino, а также отправлять данные обратно.
- **G:** Окно редактора: здесь вы будете писать код.
- **H:** Консоль ошибок: здесь вы увидите все сообщения об ошибках вашего кода.
- **I:** Строка состояния: здесь будет отображаться имя подключенной платы Arduino вместе с номером COM-порта.

Прежде чем вы начнете

После ознакомления с Arduino UNO и IDE, есть несколько вещей, которые вам нужно знать, прежде чем погрузиться в мир Arduino.

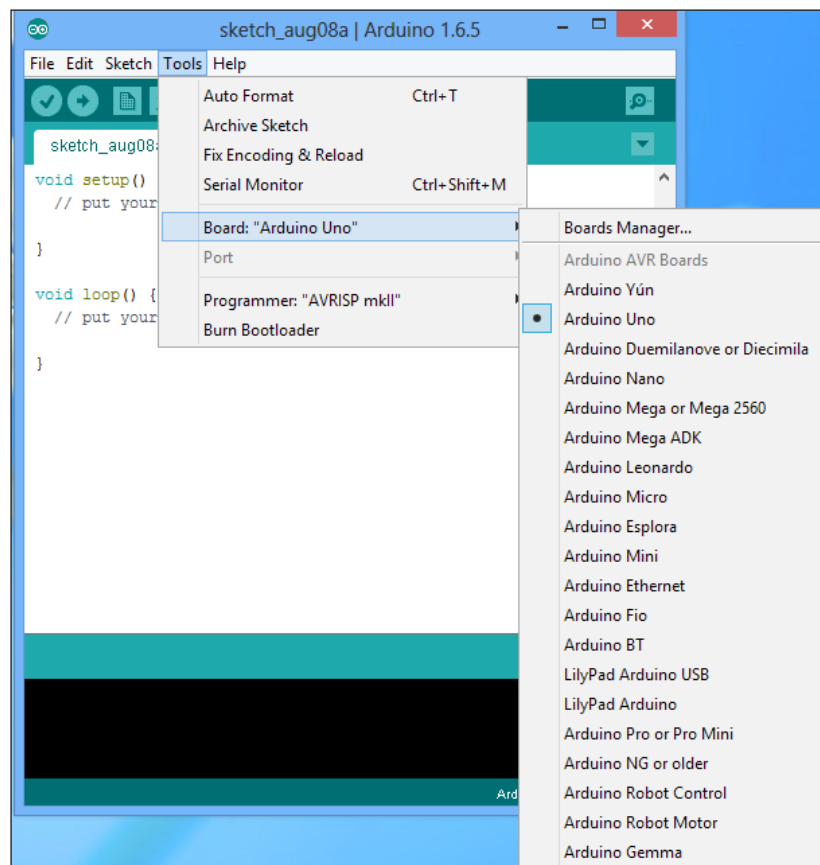
Источник питания

Как упоминалось в разделе Arduino UNO, есть два способа включить Arduino UNO. Первый - с помощью USB-кабеля, подключенного к вашему компьютеру, а второй - от внешнего источника питания 12 В. Пожалуйста, убедитесь, что вы не используете источник питания с напряжением более 20 вольт, так как вы можете сжечь вашу плату Arduino. Рекомендуемое напряжение для большинства моделей Arduino составляет от 6 до 12 вольт.

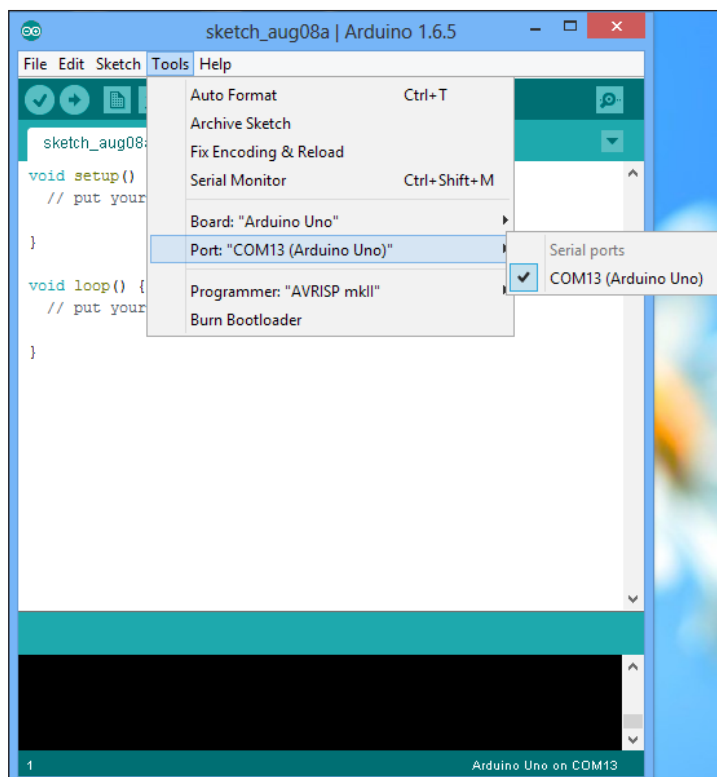
Проверка соединения

Это последний шаг перед написанием программы Hello World на Arduino:

1. Убедитесь, что вы выбрали Arduino UNO в меню Tools | Плата. Если у вас есть другая плата Arduino, убедитесь, что вы выбрали эту плату:



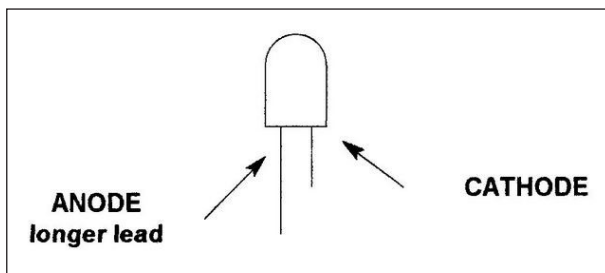
2. Выберите COM-порт в Инструменты | Порт, к которому подключена ваша плата Arduino UNO. На следующем изображении это COM13, но он будет \ может отличаться на вашем компьютере:



"Hello World"

Вы должны знать о выражении «Hello World» из области компьютерных наук, где вы пишете фрагмент кода, который будет отображать «Hello World» на мониторе. В пространстве платы Ардуино «Hello World» означает мигание светодиода путем написания простой программы.

Более длинный вывод светодиода - это анод, а другой конец - катод, как показано на рисунке:



Подключите более длинный вывод светодиода к контакту 13 платы *Arduino*, а другой вывод - к GND и напишите следующий код в новом окне редактора:

```
// функция настройки запускается один раз, когда вы нажимаете
кнопку сброса или включаете плату
void setup() {
  // инициализируем цифровой вывод 13 как выход.
  pinMode(13, OUTPUT);
}

// функция цикла запускается снова и снова навсегда
void loop() {
  digitalWrite(13, HIGH); // включаем светодиод
  delay(1000);           // ждем секунду (1000 миллисекунд)
  digitalWrite(13, LOW); // выключаем светодиод
  delay(1000);           // ждем секунду (1000 миллисекунд)
}
```

Из предыдущего кода вы заметите, что в каждой программе есть две важные функции. Первый - это `void setup()`, которая запускается только один раз, когда вы включаете плату или нажимаете кнопку сброса. Вторая функция - это `void loop()` цикл, который повторяется бесконечно.

В функции `void setup()` вы должны написать код, который нужно выполнить один раз, например, определение переменной, инициализацию порта как INPUT или OUTPUT. В предыдущем коде цифровой вывод 13 определен как ВЫХОД. В функции цикла `void loop()` первая строка подает HIGH (ВЫСОКОЕ) напряжение на контакт 13, которая включит светодиод, подключенный к контакту 13.

Программа «Hello World» включит светодиод на одну секунду и выключит светодиод на одну секунду. Функция задержки (1000) вызовет задержку одну секунду, и после этого `digitalWrite (13, LOW)` подаст низкое напряжение (0 В) на контакт 13, что выключит светодиод. Опять же, прежде чем включить светодиод, вам нужно подождать одну секунду, поставив задержку (1000) в конце кода.

Использование последовательной связи

Последовательная связь используется для связи между платой Arduino и компьютером или другими устройствами. Все платы Arduino имеют как минимум один последовательный порт, также известный как UART. Последовательная передача данных - это когда мы передаем данные по одному биту, один за другим.

Последовательная запись

В этом примере плата Arduino будет взаимодействовать с компьютером через Монитор Последовательного Порта (МПП).

Напишите следующий код в редакторе Arduino:

```
void setup()                // запускаем один раз при запуске скетча
{
  Serial.begin(9600);       // настраиваем на 9600 бит / с

  Serial.println("Hello world!"); // печатает привет с окончанием перевода строки
}

void loop()                 // запускаем снова и снова
{
  // ничего не делать!
}
```

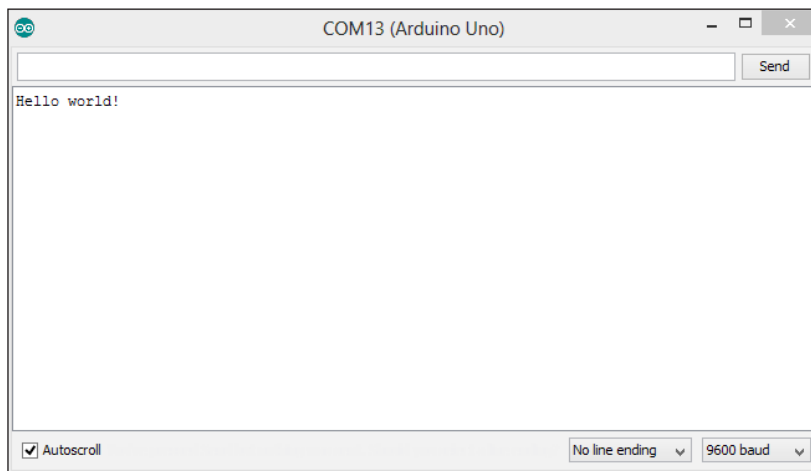


Даже если у вас нет ничего в процедурах настройки или цикла, *Arduino* требует, чтобы они присутствовали. Таким образом, он будет знать, что вы действительно ничего не собираетесь делать, вместо того, чтобы забыть включить их!

`Serial.begin` настраивает *Arduino* на желаемую скорость передачи данных, в данном случае 9600 бит в секунду. `Serial.println` отправляет данные с *Arduino* на компьютер. После компиляции и загрузки на подключенную плату *Arduino* откройте **МПП** из *Arduino IDE*. Вы должны увидеть `Hello world!` Текст отправляется с платы *Arduino*:



Если у вас возникли проблемы с поиском **МПП**, прочитайте [Раздел *Arduino IDE*](#) в этой главе.



Чтение с **МПП**

В предыдущем примере для отправки команды из *Arduino* на ваш компьютер использовался **МПП**. В этом примере вы отправите команду с компьютера, и *Arduino* выполнит определенную операцию (включит / выключит светодиод) на основе полученной команды:

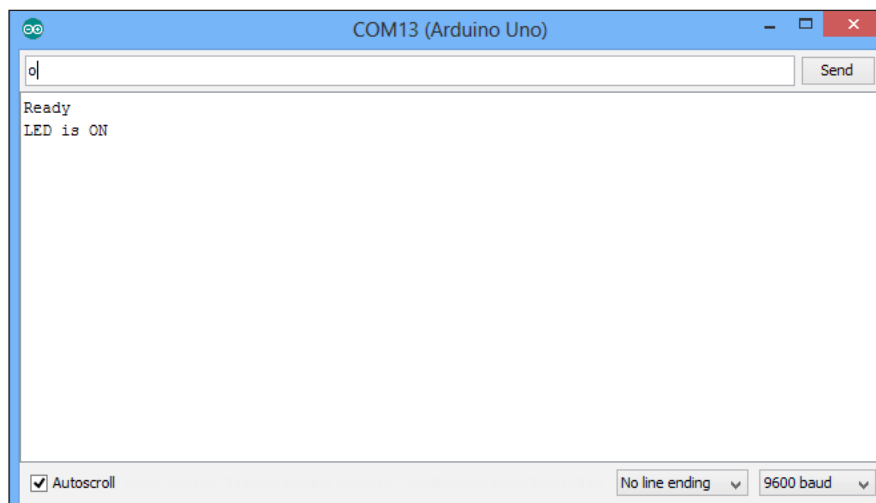
```
int inByte; // Сохраняет входящую команду

void setup() {
```

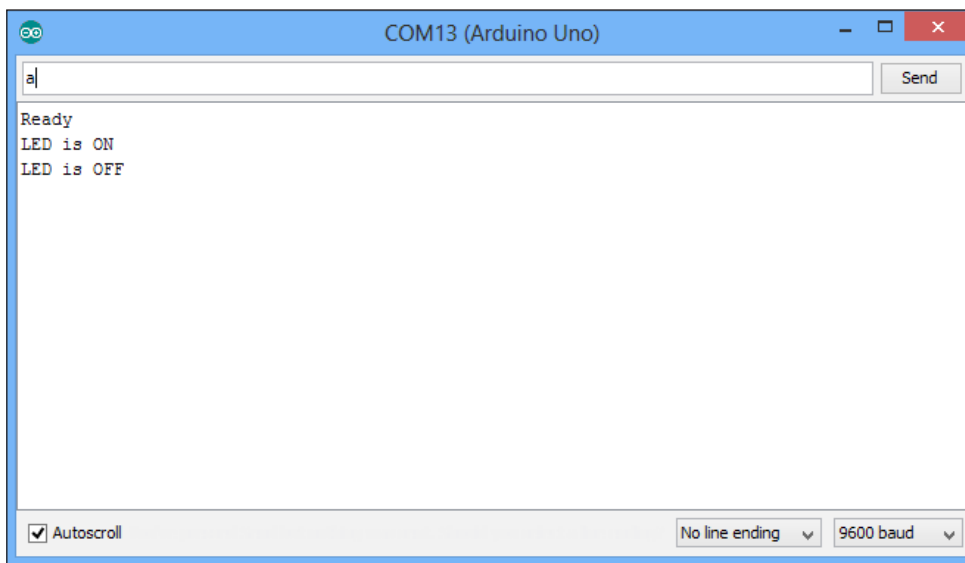


```
Serial.begin(9600);
pinMode(13, OUTPUT); // LED pin
Serial.println("Ready"); // Готовы принимать команды
}
void loop() {
  if(Serial.available() > 0) { // Байт готов к приему
    inByte = Serial.read();
    if(inByte == 'o') { // байт равен 'o'
      digitalWrite(13, HIGH);
      Serial.println("LED is ON");
    }
    else
    {
      // байт не 'o'
      digitalWrite(13, LOW);
      Serial.println("LED is OFF");
    }
  }
}
```

Функция `inByte` сохранит входящий последовательный байт. Из предыдущего примера вы должны быть знакомы с командами, написанными в функции настройки. В функции цикла сначала вам нужно знать, когда байт доступен для чтения. Функция `Serial.available()` возвращает количество байтов, доступных для чтения. Если он больше 0, `Serial.read()` прочтает байт и сохранит его в переменной `inByte`. Допустим, вы хотите включить светодиод, когда доступна буква «o». Для этого вы будете использовать условие `if`, и вы будете проверять, равен ли полученный байт «o» или нет. Если это «o», включите светодиод, установив контакт 13 в HIGH. Arduino также отправит на компьютер сообщение о включении светодиода, которое можно просмотреть в МПП:



Если это какой-либо другой символ, выключит светодиод, установив контакт 13 в положение LOW. *Arduino* также отправит на компьютер сообщение LED is OFF, которое можно просмотреть в МПП:



Мир светодиодов

Светодиод (LED) означает светоизлучающий диод, поэтому он излучает свет, когда на анод и катод LED подается достаточное напряжение. Сегодняшние LED доступны во многих различных типах, формах и размерах - прямой результат огромных улучшений в полупроводниковой технологии за последние годы. Эти достижения привели к лучшему освещению, увеличению срока службы и снижению энергопотребления. Они также привели к более сложному принятию решений, поскольку существует очень много типов LED на выбор.

LED можно разделить на миниатюрные светодиоды, светодиоды высокой мощности и светодиоды для конкретных приложений:

- **Миниатюрные светодиоды:** эти светодиоды очень маленькие и обычно доступны в одном цвете / форме. Их можно использовать в качестве индикаторов на таких устройствах, как сотовые телефоны, калькуляторы и пульты дистанционного управления.
- **LED высокой мощности:** , они являются прямым результатом усовершенствованной диодной технологии. Они имеют гораздо больший световой поток, чем стандартные светодиоды. Обычно эти светодиоды используются в автомобильных фарах.

- **Светодиоды для конкретных приложений:** как следует из названия, в эту категорию попадают многие светодиоды. Это мигающие светодиоды, светодиоды RGB, семисегментный дисплей, светодиодные лампы и светодиодные полосы.

Резюме

В этой главе был рассмотрен обзор различных плат Arduino с подробным объяснением платы Arduino UNO. Объясняется IDE Arduino с инструкциями по установке для Windows. Был написан скетч «Hello World» для последовательной связи.

В следующей главе мы разработаем светодиодную лампу настроения, и вы познакомитесь с некоторыми художественными вещами, помимо программирования светодиодов.

2

Проект 1 - Светодиодный НОЧНИК

В главе 1 «Начало работы с Arduino и светодиодами» вы узнали о «Привет, мир» физических вычислений. Теперь, когда у вас есть базовые знания об Arduino и ее IDE, мы можем перейти к некоторым интересным вещам, связанным с управлением светодиодами. В этой главе будут рассмотрены следующие темы:

- Введение в макетную плату
- Управление несколькими светодиодами
- Затухание светодиода
- Создание лампы настроения
- Разработка светодиодного ночника.

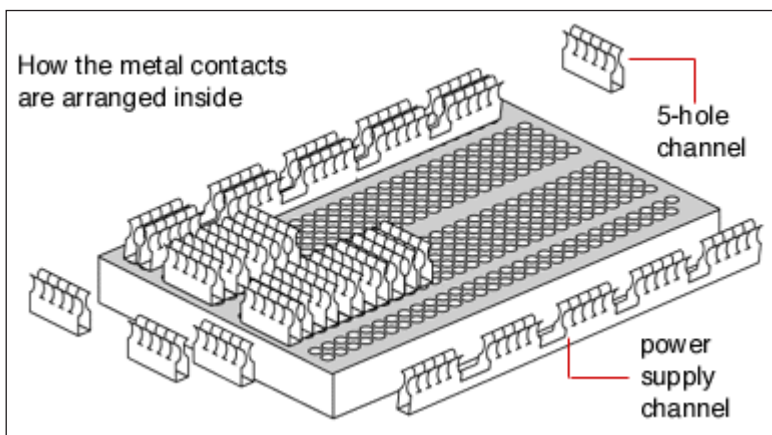
К концу этой главы вы научитесь художественно управлять различными светодиодами.

Введение в макетную плату

Прототипирование - это процесс проверки идеи путем создания предварительной модели, на основе которой могут быть разработаны или сформированы другие продукты. Макетная плата - одна из самых фундаментальных частей для создания прототипов электронных схем. Поскольку она не требует пайки, ее также называют «беспаячной платой».

Структура макета

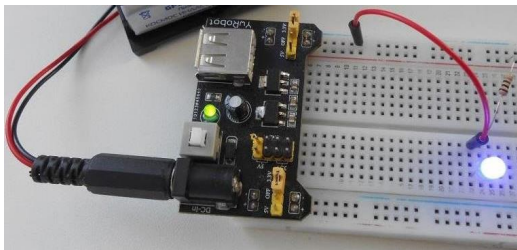
Практически все современные макеты сделаны из пластика. Современный макет представляет собой перфорированный пластиковый блок с многочисленными металлическими зажимами под перфорацией. На макетной плате есть металлические полосы под платой и отверстия наверху платы. На следующем изображении вы можете увидеть структуру макета:

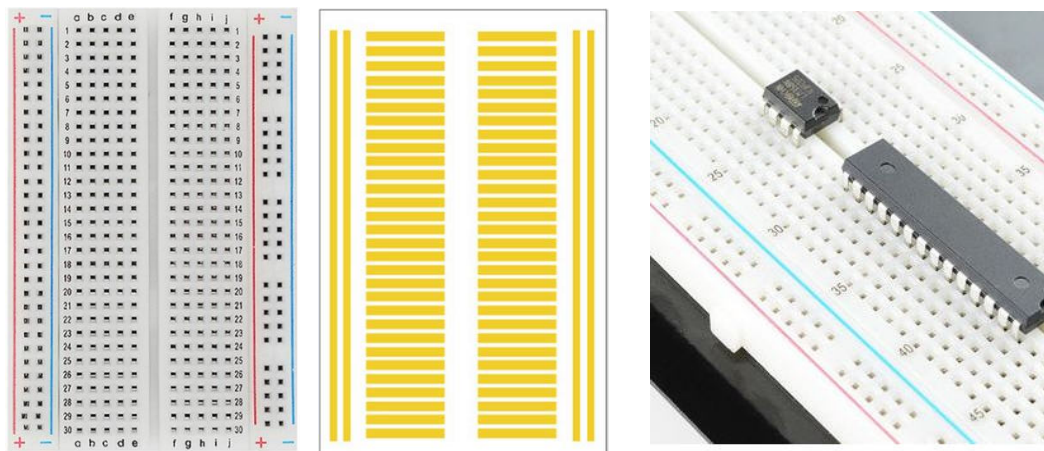


Основная структура макета состоит из главной центральной области, которая представляет собой блок из двух наборов столбцов, где каждый столбец состоит из множества строк. Все эти ряды соединяются построчно.

Использование макета

На макетной плате есть много полос меди под платой, которые соединяют отверстия. Верхние синие линии не связаны с нижними. В случае электронных схем требуется питание на различных контактах. Таким образом, вместо того, чтобы делать много подключений к источнику питания, можно подключить источник питания к одному из отверстий на макетной плате, а источник питания - через его внешние отверстия или же использовать плату внешнего источника питания:





Несколько макетов могут быть соединены вместе, чтобы образовать большие экспериментальные площадки для макетных плат. Если вы хотите использовать какой-либо чип / IC (интегральную схему), вы можете разместить его как показано на следующем рисунке. Нам будет очень легко понять макетную плату, поскольку мы будем широко использовать макетную плату для всех наших проектов.

Управление несколькими светодиодами

Во встроенном мире управление одним светодиодом - это код «Hello World», который мы узнали в первой главе. Теперь, когда мы знакомы с концепцией светодиодов, мы можем начать управлять несколькими светодиодами с помощью Arduino. Здесь мы начнем с создания простого модуля светофора.

Простой контроллер светофора

Как мы все знаем, светофор состоит из трех светодиодов: красного, желтого и зеленого. Для реализации этого проекта нам понадобятся красные, зеленые и желтые светодиоды, зачистные провода и несколько резисторов на 255 Ом.

В предыдущей главе в нашей программе «Hello World» мы подключили светодиод напрямую к контакту 13. Здесь мы подключим красный, желтый и зеленый светодиоды к контактам 9, 10 и 11 соответственно. В случае контакта 13 он имеет встроенный подтягивающий резистор. Подтягивающие резисторы используются для ограничения тока, подаваемого на светодиод. Мы не можем подавать на светодиоды ток более нескольких мА. Но с выводом 13 ток в Arduino ограничен внутренним подтягивающим резистором. Если мы хотим подключиться к контактам, отличным от 13, нам нужно добавить резисторы в схему.

Подключите более длинную головку красного светодиода к контакту 9, а более короткую - к резистору, а затем резистор к земле. Аналогичным образом подключите желтый и зеленый светодиоды к контактам 10 и 11 соответственно.

После того, как вы установили соединение, напишите следующий код в окне редактора Arduino IDE:

```
// Инициализируем светодиоды для использования в коде.
// Инициализация вне любой функции, чтобы доступ к глобальной
переменной можно было получить по всему коду

int redLed = 9;
int yellowLed = 10;
int greenLed = 11;

// функция настройки запускается один раз, когда вы нажимаете
кнопку сброса или включаете плату
void setup() {
  // инициализируем цифровой вывод для красного, желтого и
зеленого светодиода в качестве выхода.
  pinMode(redLed, OUTPUT);
  pinMode(yellowLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}

// функция цикла запускается снова и снова до бесконечности
void loop() {
  digitalWrite(redLed, HIGH);      // Выстав. для крас.LED выс.уровень
  digitalWrite(yellowLed, LOW);    //... для желтого-низк.уровень
  digitalWrite(greenLed, LOW);    //...для зеленого - низк. уровень
  delay(10000);                    // Ждем 10 секунд

  digitalWrite(redLed, LOW);
  // Making red led low
  digitalWrite(yellowLed, LOW);    //... для желтого-низк.уровень
  digitalWrite(greenLed, HIGH);   //...для зеленого - высок. уровень
  delay(10000);                    // Ждем 10 секунд

  digitalWrite(redLed, LOW);      //... для красн.LED-низк. уровень
  digitalWrite(yellowLed, HIGH);  //... для желтого-выс. уровень
  digitalWrite(greenLed, LOW);    //...для зеленого - низк. уровень
  delay(3000);                     // Ждем 3 секунды

}
```

В приведенном выше коде вы можете видеть, что она очень похожа на программу «Hello World», за исключением того, что здесь мы управляем несколькими светодиодами. Здесь мы инициализируем переменные как целые числа и используем эту же переменную во всем коде. Итак, в будущем, если нам понадобится изменить вывод Arduino, нам просто нужно будет внести изменения в одном месте, вместо того, чтобы вносить изменения в нескольких местах кода. Рекомендуется использовать переменные вместо прямого использования номеров выводов в коде. В функции настройки мы устанавливаем выводы как ВЫХОД. Если мы не инициализируем порт как INPUT или OUTPUT, порт может находиться в неопределенном состоянии. Таким образом, он выдаст случайный результат.

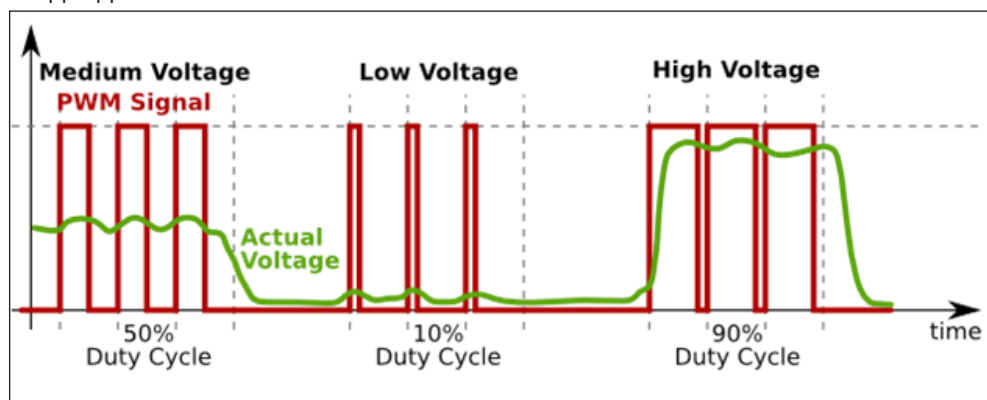
Мы завершили код для одного направления нашего светофора. Аналогичным образом вы можете создать свой код для других оставшихся направлений.

Светодиодное затухание

Вы можете создать затухание LED, используя функцию `Arduino analogWrite (pin, value)`. Прежде чем мы перейдем к использованию функции `analogWrite ()`, мы разберемся с концепцией, лежащей в ее основе. Для создания аналогового сигнала Arduino использует метод, называемый **широтно-импульсной модуляцией (PWM)**.

Широтно-импульсная модуляция (ШИМ)

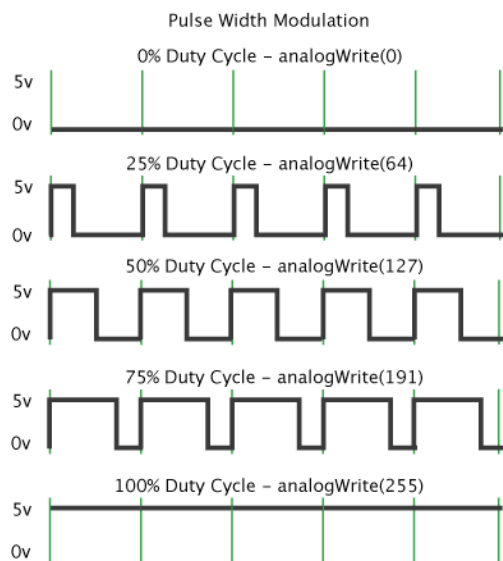
ШИМ - это метод получения аналогового сигнала с помощью цифровых средств. Изменяя рабочий цикл (рабочий цикл - это процент периода, когда сигнал активен), мы можем имитировать «среднее» аналоговое напряжение. Как вы можете видеть на следующем изображении, когда нам нужно среднее напряжение, мы сохраняем рабочий цикл равным 50%. Точно так же, если мы хотим достичь низкого и высокого напряжения, мы сохраним рабочий цикл на уровне 10% и 90% соответственно. В этом приложении ШИМ - это процесс управления напряжением, подаваемой на светодиод:



Использование ШИМ на Arduino

Arduino UNO имеет 14 цифровых контактов ввода / вывода. Как упоминалось в главе 1 «Начало работы с Arduino и светодиодами», мы можем использовать шесть выводов (3, 5, 6, 9, 10 и 11) в качестве выводов PWM. Эти выводы управляются встроенными таймерами, которые автоматически переключают выводы с частотой около 490 Гц. Как обсуждалось ранее, мы будем использовать функцию `analogWrite()`.

На следующем рисунке функция `analogWrite()` принимает в качестве параметра номер вывода и значение вывода. Здесь, как вы можете видеть на изображении, значение вывода может быть от 0 до 255, а рабочий цикл сопоставлен с 0% и 100%:



Подключите анод (более длинный вывод) светодиода к выводу 11 (вывод ШИМ платы) через резистор 220 Ом, а катод к земле и напишите следующий код в окне редактора или вставьте из папки с кодами:

```
int led = 11;           // вывод, к которому подключен светодиод
int brightness = 0;    // насколько яркий светодиод
int steps = 5;        // на сколько точек погаснет светодиод

void setup() {
  pinMode(led, OUTPUT);
}
```

```
void loop() {
  // Установка яркости светодиода:
  analogWrite(led, brightness);

  // изменяем яркость в следующий раз через цикл:
  brightness = brightness + steps;

  // Когда значение яркости достигает 0 или 255, обратное направление

  if (brightness == 0 || brightness == 255) {
    steps = -steps ;
  }
  // ждем 30 миллисекунд, чтобы увидеть эффект затухания
  delay(30);
}
```

Мы используем вывод 11 (вывод ШИМ) для затухания светодиода. Мы сохраняем яркость светодиода в переменной яркости. Изначально мы устанавливаем яркость светодиода 0.

Когда функция цикла снова запускается, мы увеличиваем значение с шагом 5. Как и в функции `analogWrite()`, мы можем установить значение от 0 до 255. Когда яркость достигает максимума, мы уменьшаем значение. Точно так же, когда яркость достигает 0, мы начинаем увеличивать яркость с шагом 1. Чтобы увидеть эффект затемнения, мы помещаем задержку в 30 миллисекунд в конце кода.

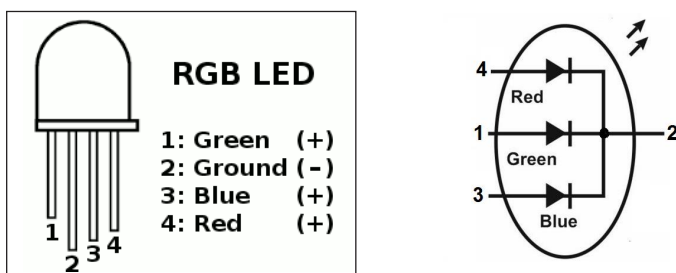
Создание лампы настроения

Освещение - одна из самых больших возможностей для домовладельцев эффективно влиять на атмосферу в своем доме, будь то для комфорта и удобства или для создания настроения для гостей. В этом разделе мы сделаем простую, но эффективную лампу настроения, используя нашу собственную Arduino. Мы будем использовать светодиод RGB для создания нашей лампы настроения. Светодиод RGB (красный, зеленый и синий) имеет все три светодиода в одном корпусе, поэтому нам не нужно использовать три разных светодиода для получения разных цветов. Кроме того, смешивая значения, мы можем имитировать множество цветов, используя сложное программирование. Говорят, что мы можем производить 16,7 миллиона различных цветов.

Использование светодиода RGB

Светодиод RGB - это просто три светодиода, помещенные в один корпус.

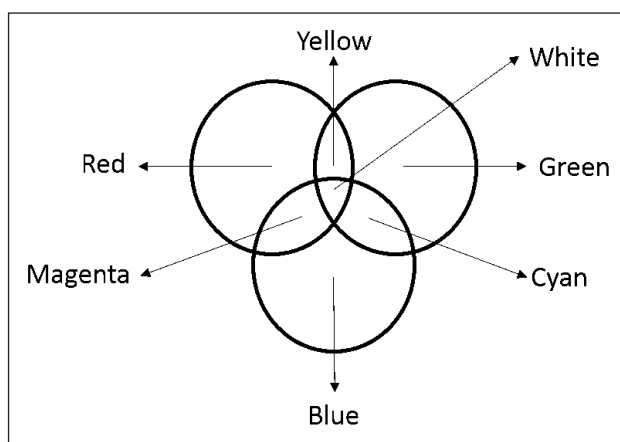
Поскольку у нашего светодиода RGB все катоды закорочены друг с другом, его также обычно называют анодным светодиодом RGB:



Нумерация выводов представлена на рисунке. Обязательно используйте токоограничивающие резисторы для защиты светодиодов от перегорания. Здесь мы будем смешивать цвета, как мы смешиваем краски на палитре или микшируем звук с помощью микшера. Но чтобы получить нужный цвет, нам придется подавать разные аналоговые напряжения на выводы светодиода.

Почему светодиоды RGB меняют цвет?

Поскольку ваш глаз имеет три типа перехватчиков света (красный, зеленый и синий), вы можете смешивать любой цвет, который вам нравится, варьируя количество красного, зеленого и синего света. Ваши глаза и мозг обрабатывают количество красного, зеленого и синего и преобразуют их в цвет спектра:



Если мы установим одинаковую яркость всех наших светодиодов, общий цвет света будет белым. Если мы выключим красный светодиод, то будут гореть только зеленый и синий светодиоды, которые станут голубым цветом. Мы можем управлять яркостью всех трех светодиодов, что позволяет сделать любой цвет. Кроме того, три разных светодиода внутри одного светодиода RGB могут иметь разные уровни напряжения и тока; вы можете узнать о них в таблице данных. Например, для красного светодиода обычно требуется 2 В, а для зеленого и синие светодиоды могут упасть до 3-4 В.

Создание лампы настроения

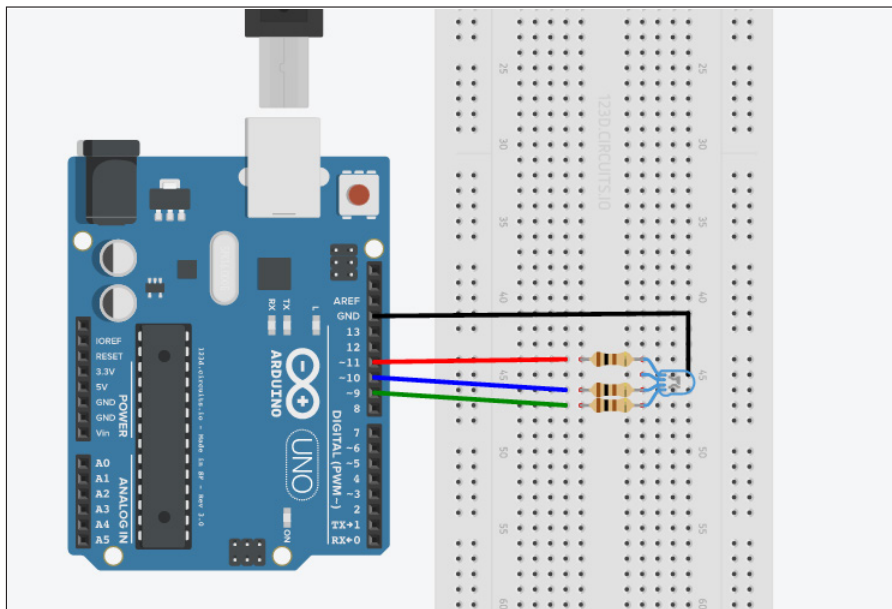
Теперь мы готовы использовать наш светодиод RGB в нашей лампе настроения. Мы начнем с разработки схемы нашей лампы настроения. В нашей лампе настроения мы сделаем плавный переход между несколькими цветами.

Для этого нам потребуются следующие компоненты:

- Светодиод RGB
- Резисторы 270 Ом
- Макетная плата

наш светодиод RGB состоит из трех светодиодов. Итак, нам нужны три разных вывода управления для управления тремя светодиодами. Точно так же для каждого светодиода требуется три токоограничивающих резистора с сопротивлением от 220 Ом до 470 Ом. Мы выбрали 270 Ом.

Как обсуждалось в предыдущем разделе, мы хотим управлять напряжением, подаваемое на каждый анод RGB LED. Поэтому нам придется использовать выводы ШИМ (9, 10 и 11). Следующая схема управляет красным светодиодом с вывода 11, синим светодиодом с вывода 10, зеленым светодиодом с вывода 9. Вывод GND подключите к самому длинному выводу светодиода. Произведите соединения на макетной плате согласно приведенному рисунку:



После того, как вы произвели соединения, напишите следующий код в окне редактора

```
int redLed = 11;
int blueLed = 10;
int greenLed = 9;

void setup()
{
  pinMode(redLed, OUTPUT);
  pinMode(blueLed, OUTPUT);
  pinMode(greenLed, OUTPUT);
}
void loop()
{
  setColor(255, 0, 0); // Red
  delay(500);
```



```

    setColor(255, 0, 255); // пурпурный
    delay(500);
    setColor(0, 0, 255); // Синий
    delay(500);
    setColor(0, 255, 255); // Голубой
    delay(500);
    setColor(0, 255, 0); // Зеленый
    delay(500);
    setColor(255, 255, 0); // Желтый
    delay(500);
    setColor(255, 255, 255); // Белый
    delay(500);
}
void setColor(int red, int green, int blue)
{
    // Для светодиода с обычным анодом нам нужно вычесть значение из 255.
    red = 255 - red;
    green = 255 - green;
    blue = 255 - blue;
    analogWrite(redLed, red);
    analogWrite(greenLed, green);
    analogWrite(blueLed, blue);
}

```

Мы используем очень простой код для изменения цвета светодиода синтервалом в одну секунду. Здесь мы устанавливаем цвет каждую секунду. Таким образом, этот код не даст вам плавного перехода между цветами. Но с помощью этого кода вы сможете запустить светодиод RGB. Теперь мы изменим этот код для плавного перехода между цветами. Для плавного перехода между цветами, воспользуемся следующим кодом:

```

int redLed = 11;
int greenLed = 10;
int blueLed = 9;

int redValue = 0;
int greenValue = 0;
int blueValue = 0;

void setup() {
    randomSeed(analogRead(0));
}

void loop() {
    redValue = random(0,256); // Произвольно генерировать от 1 до 255
    greenValue = random(0,256); // Произвольно генерировать от 1 до 255
    blueValue = random(0,256); // Произвольно генерировать от 1 до 255
}

```

```
    analogWrite(redLed, redValue);
    analogWrite(greenLed, greenValue);
    analogWrite(blueLed, blueValue);

// Увеличение всех значений одно за другим после установки
случайных значений.
    for(redValue = 0; redValue < 255; redValue++){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }
    for(greenValue = 0; greenValue < 255; greenValue++){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }
    for(blueValue = 0; blueValue < 255; blueValue++){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }

//Уменьшение всех значений по одному для выключения всех
светодиодов.
    for(redValue = 255; redValue > 0; redValue--){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }
    for(greenValue = 255; greenValue > 0; greenValue--){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }
    for(blueValue = 255; blueValue > 0; blueValue--){
        analogWrite(redLed, redValue);
        analogWrite(greenLed, greenValue);
        analogWrite(blueLed, blueValue);
        delay(10);
    }
}
```

Мы хотим, чтобы наша лампа настроения снова и снова повторяла одну и ту же последовательность цветов. Итак, мы используем функцию `randomSeed ()`. Функция `randomSeed ()` инициализирует генератор псевдослучайных чисел, который запускается в произвольной точке и будет повторяться в той же последовательности снова и снова. Эта последовательность очень длинная и случайная, но всегда будет одинаковой.

Итак, когда мы запускаем нашу последовательность с помощью `analogRead (0)`, она выдаст некоторое случайное число, которое справедливо при инициализации генератора случайных чисел. Функция `random (min, max)` генерирует случайное число между минимальным и максимальным значениями, указанными в качестве параметров. В функции `analogWrite ()` число должно быть от 0 до 255. Итак, мы устанавливаем `min` и `max` как 0 и 255 соответственно. Мы устанавливаем случайное значение для `redPulse`, `greenPulse` и `bluePulse`, которые мы устанавливаем для контактов. После генерации случайного числа мы увеличиваем или уменьшаем значение, сгенерированное с шагом 1, что сгладит переход между цветами.

Теперь мы готовы использовать это как лампу настроения в нашем доме. Но перед этим нам нужно спроектировать внешний корпус нашей лампы. Мы можем использовать белую бумагу (сложенную в форме куба), чтобы разместить вокруг нашего светодиода RGB. Белая бумага действует как рассеиватель, что гарантирует смешивание света. В качестве альтернативы вы можете использовать все, что рассеивает свет и сделать вещи красивыми! Если вы хотите сделать уменьшенную версию лампы настроения, сделайте отверстие в шарике для пинг-понга и вставьте этот светодиод в шарик.

Разработка светодиодного ночника

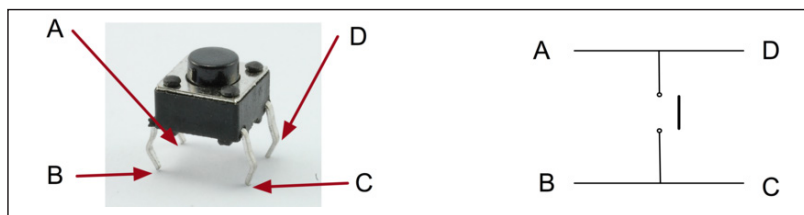
Итак, теперь мы разработали нашу лампу настроения, но она будет включаться только тогда, когда мы подключим блок питания к Arduino. Она не будет включаться или выключаться в зависимости от темноты окружающей среды. Кроме того, чтобы выключить его, мы должны отключить наш блок питания от Arduino. В этом разделе мы узнаем, как использовать переключатели с Arduino.

Введение в выключатель

Выключатели - один из самых простых и незаменимых компонентов. Выключатели делают только одно: либо размыкают, либо замыкают цепь. В основном есть два типа выключателей:

- Кнопки: клавиши клавиатуры и кнопка сброса на плате Arduino.
- Выключатели: после нажатия остаются включенными как настенный выключатель

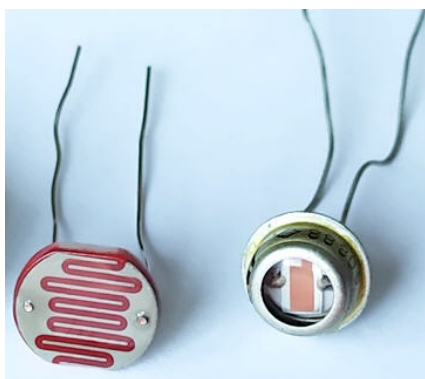
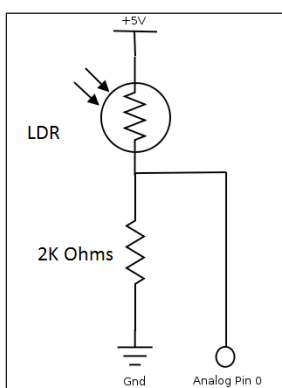
Обычно все кнопки являются нормально разомкнутые типа NO. Таким образом, когда кнопка приводится в действие, она замыкает цепь и действует как идеальный кусок проводящего провода. Помимо этого, в зависимости от их работы в мире существует множество кнопок, таких как тумблерные, поворотные, DIP, кулисные, мембранные и т. д. Здесь мы будем использовать обычную кнопку с четырьмя контактами:



В нашей кнопке контакты A-D и B-C замкнуты. Мы подключим нашу цепь между A и C. Итак, всякий раз, когда вы нажимаете кнопку, цепь замыкается, и через нее будет течь ток. Мы будем читать ввод от кнопки, используя функцию `digitalRead()`. Мы подключим один вывод (вывод A) к 5 В, а другой вывод (вывод C) - к цифровому входному выводу Arduino (вывод 2). Так что всякий раз, когда нажата кнопка, она отправит 5 В на вывод 2.

Лампа Pixar

Мы добавим еще несколько вещей в обсуждаемую нами лампу настроения, чтобы сделать ее более надежной и простой в использовании. Наряду с кнопкой мы добавим светочувствительный компонент, чтобы сделать включение автоматическим. Мы будем использовать светорезистор (LDR), (его еще называют фоторезистором) для восприятия света и управления лампой. По сути, LDR - это резистор, сопротивление которого изменяется при изменении интенсивности падающего на него света. В основном сопротивление LDR падает с увеличением света. Чтобы получить изменения значений в соответствии с уровнями освещенности, нам необходимо подключить наш LDR по следующей схеме:



Здесь мы используем схему делителя напряжения для измерения изменения интенсивности света. При изменении интенсивности света сопротивление LDR изменяется, что, в свою очередь, изменяет напряжение на резисторе. Мы можем считать напряжение с любого аналогового вывода с помощью `analogRead()`.

После того, как вы подключили схему, как показано, напишите в редакторе следующий код:

```
int LDR = 0; //будет входить с вывода A0
int LDRValue = 0;
int light_sensitivity = 400; //Это приблизительное значение света
падающего на ваш LDR
surrounding your LDRint
LED = 13;
void setup()
{
  Serial.begin(9600); //запускаем МПП с 9600 бод
  pinMode(LED, OUTPUT);
}

void loop()
{
  LDRValue = analogRead(LDR); //Считываем значение LDR через вывод A0
  pin A0
  Serial.println(LDRValue); //Выводим значения LDR на МПП
  monitor

  if (LDRValue < light_sensitivity)
  {
    digitalWrite(LED, HIGH);
  }
  else
  {
    digitalWrite(LED, LOW);
  }
  delay(50); //Задержка перед повторным чтением значения LDR
}
```

В предыдущем коде мы считываем значение с нашего LDR на аналоговом выводе A0. Когда значение, считываемое с вывода A0, ниже определенного порогового значения, мы включаем светодиод или в нашем случае, лампу настроения.

Точно так же мы добавим кнопку в нашу лампу настройки, чтобы она полностью работала как лампа Pixar.

Подключите один контакт кнопки к 5 В, а другой контакт к цифровому выводу 2. Мы будем включать лампу только и только тогда, когда в комнате темно и выключатель включен. Итак, мы внесем следующие изменения в предыдущий код. В функции настройки инициализируйте контакт 2 как вход, а в цикле добавьте следующий код:

```
buttonState = digitalRead(pushSwitch); //pushSwitch инициализируется как 2.

If (buttonState == HIGH){

//Включаем лампу
}
Else {
//Выключаем лампу.
Поочередно выключаем все светодиоды для плавного выключения лампы.
  for(redValue = 255; redValue > 0; redValue--){
    analogWrite(redLed,redValue);
    analogWrite(greenLed,greenValue);
    analogWrite(blueLed,blueValue);
    delay(10);
  }
  for(greenValue = 255; greenValue > 0; greenValue--){
    analogWrite(redLed,redValue);
    analogWrite(greenLed,greenValue);
    analogWrite(blueLed,blueValue);
    delay(10);
  }
  for(blueValue = 255; blueValue > 0; blueValue--){
    analogWrite(redLed,redValue);
    analogWrite(greenLed,greenValue);
    analogWrite(blueLed,blueValue);
    delay(10);
  }
}
```

Итак, теперь мы включили LDR и включили нашу лампу, чтобы использовать ее как обычно.

Резюме

В этой главе мы начали с макетной платы. Программирование нескольких светодиодов объяснялось разработкой модели контроллера светофора. К концу этой главы были разработаны лампа настроения со светодиодной подсветкой RGB и лампа Pixar с чувствительной частью.

В следующей главе будет разработана подсветка телевизора с дистанционным управлением, где вы узнаете, как установить связь между Arduino и пультом от телевизора.

3

Проект 2 - Подсветка телевизора с дистанционным управлением

В предыдущей главе «Проект 1 - Светодиодный ночник» мы погрузились в удивительный мир светодиодов. Мы создали несколько крутых проектов с использованием разных типов светодиодов. Теперь мы будем использовать другой тип светодиода - инфракрасный светодиод. В этой главе мы начнем с изучения основ ИК-светодиодов и основ ИК-связи. Как только вы научитесь программировать ИК-датчик, мы будем использовать его для управления подсветкой телевизора с помощью пульта дистанционного управления. В этой главе вы узнаете о следующем:

- Знакомство с ИК-светодиодами и принцип их работы
- Программирование ИК-датчика
- Как управлять светодиодной матрицей
- Разработка подсветки телевизора с дистанционным управлением.

Знакомство с ИК-светодиодами

В мире беспроводных технологий ИК (инфракрасный порт) - один из самых распространенных, недорогих и простых в использовании способов связи. Возможно, вы всегда задавались вопросом, как работает пульт от телевизора. Пульт от телевизора использует ИК-светодиоды для отправки сигнала. Поскольку длина волны света, излучаемого ИК-светодиодом, больше, чем длина волны видимого света, его нельзя увидеть невооруженным глазом. Но если вы посмотрите в камеру своего мобильного телефона или любую другую камеру, вы увидите, как светится свет, когда вы нажимаете любую клавишу на пульте дистанционного управления. Давайте сначала разберемся, что такое ИК-светодиод и каковы различные применения ИК-светодиода.

Что такое ИК-светодиод?

ИК (инфракрасный) светодиод, также известный как ИК (инфракрасный) передатчик, излучает инфракрасные волны в диапазоне от 875 нм до 950 нм. Обычно ИК-светодиоды состоят из арсенида галлия или арсенида алюминия-галлия. Принцип работы ИК-светодиода такой же, как мы упоминали в предыдущих главах. Более длинный вывод светодиода - это анод, а более короткий - катод, как показано здесь:



Применение ИК-светодиодов / ИК-связи

Помимо использования ИК-связи в пультах дистанционного управления телевизора, существует ряд других приложений, использующих ИК-связь. Инфракрасный свет также можно использовать для передачи данных между электронными устройствами. Хотя инфракрасный свет невидим для глаза, цифровые камеры и другие датчики могут видеть этот свет, поэтому инфракрасный свет можно использовать для многих систем безопасности и технологий ночного видения. Помимо технического использования, Управление по санитарному надзору за качеством пищевых продуктов и медикаментов США одобрило несколько продуктов с ИК-светодиодами для использования в некоторых медицинских и косметических процедурах.

ИК-датчики

Мы изучили основы ИК-связи и ИК-светодиодов, поэтому теперь перейдем к созданию собственного ИК-датчика.

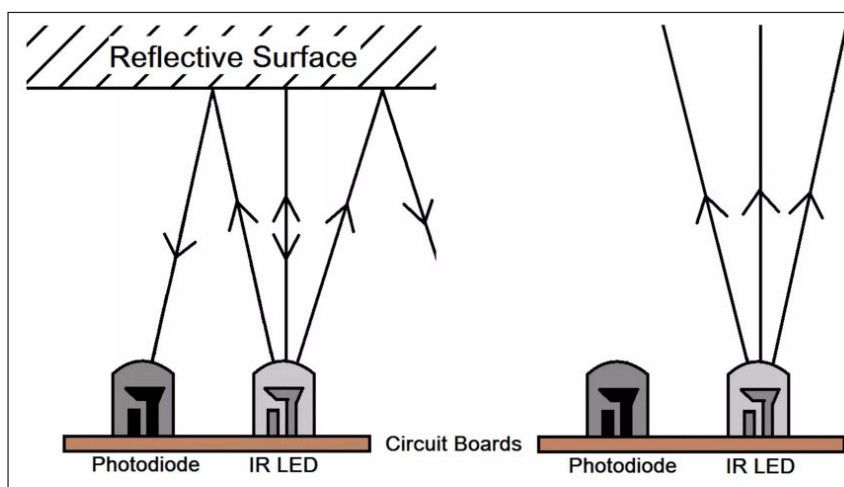
В зависимости от области применения существуют различные типы ИК-датчиков. Датчики приближения, датчики препятствий и датчики контраста (используемые в роботе-следящем за линией) - это несколько примеров, в которых используются ИК-датчики.

Рабочий механизм

ИК-датчик состоит из пары ИК-светодиода (который работает как передатчик) и фотодиода (который работает как приемник). ИК-светодиод излучает ИК-излучение, которое при приеме на фотодиод определяет выходной сигнал датчика.

Существуют разные варианты использования ИК-датчика. Например, если мы поместим ИК-светодиод прямо перед фотодиодом, так что почти все излучение достигнет фотодиода, это можно будет использовать как охранную сигнализацию. Таким образом, когда кто-либо прерывает прямую видимость между ИК-светодиодом и фотодиодом, это нарушает непрерывное излучение, исходящее от ИК-светодиода, и мы можем запрограммировать его на подачу сигнала тревоги. Этот тип механизма также называется прямым падением, поскольку мы не используем отражающую поверхность между передатчиком и приемником.

Другой вариант использования - с косвенным падением, в котором мы используем физический закон поглощения. Это означает, что когда свет направлен на черную поверхность, черная поверхность фактически поглощает свет. Точно так же, когда ИК-излучение направлено на черную поверхность, поверхность будет поглощать ИК-излучение. Но, когда он направлен на белую поверхность, поверхность будет отражать ИК-излучение. Основываясь на количестве света, полученного от поверхности, мы можем определить, следует ли роботу по линии или нет. Итак, принцип абсорбции можно использовать для робота-слеящего за линией:



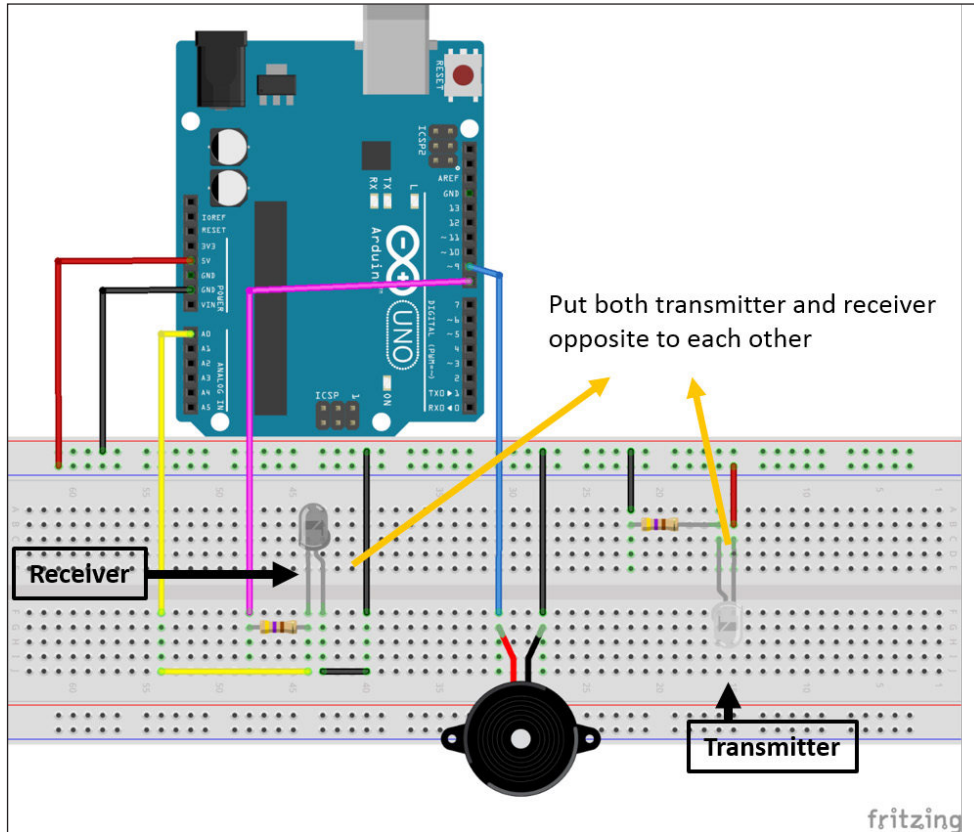
Как вы можете видеть на предыдущем изображении, всякий раз, когда на пути ИК-излучения обнаруживается какое-либо препятствие, часть ИК-излучения отражается обратно, что при получении ИК-волн дает выходной сигнал. Этот тип настройки также полезен для обнаружения любого объекта / препятствия на пути.

Программирование базового ИК-датчика

Разобравшись с основными принципами работы простого ИК-датчика, мы научимся программировать этот датчик.

Основываясь на принципе прямого попадания, мы разработаем простую охранную сигнализацию.

После того, как вы подключили схему, как показано на следующем рисунке, загрузите следующий код в Arduino:



```
int IRTransmitter = 8;
int IRReceiver = A0; //ИК-приемник подключен к выводу 5

int buzzer = 9; // Зуммер подключен к выводу 9.
int output = 0; // Переменная для хранения значения с ИК-датчика
int ambientLight = 500;

void setup()
{
  pinMode(IRReceiver, INPUT);
  pinMode(IRTransmitter, OUTPUT);
  pinMode(buzzer, OUTPUT);
  digitalWrite(IRTransmitter, HIGH);
}
```

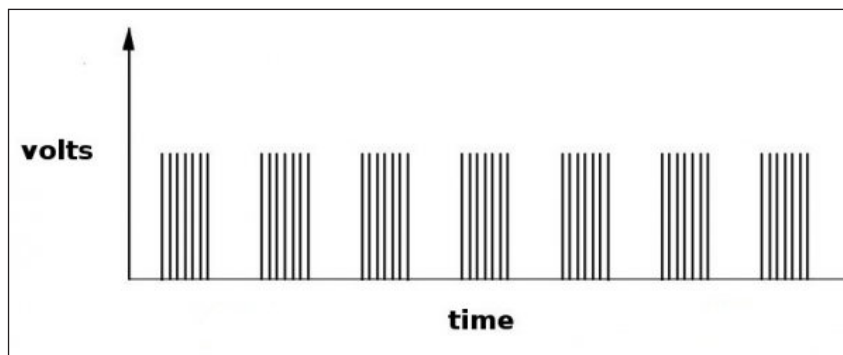
```
void loop()
{
  output = analogRead(IRReceiver);
  // Если что-то находится между ИК-передатчиком и ИК-приемником
  // ИК-приемник не выдаст выходной сигнал. Сделайте тревогу.
  if (output < ambientLight)
  {
    makeAlarm(50);
  }
}
void makeAlarm(unsigned char time)
{
  analogWrite(buzzer,170);
  delay(time);
  analogWrite(buzzer,0);
  delay(time);
}
```

В предыдущем коде мы постоянно включаем ИК-передатчик. Таким образом, ИК-приемник / фотодиод будет постоянно получать ИК-сигнал. Всякий раз, когда кто-либо пытается проникнуть в дом или сейф, ИК-сигнал прерывается, что, в свою очередь, снижает падение напряжения на ИК-приемнике и срабатывает сигнал тревоги. Здесь, в зависимости от окружающих вас условий, вам нужно будет изменить значение переменной `ambientLight` в коде.

Как получать данные с пульта от телевизора

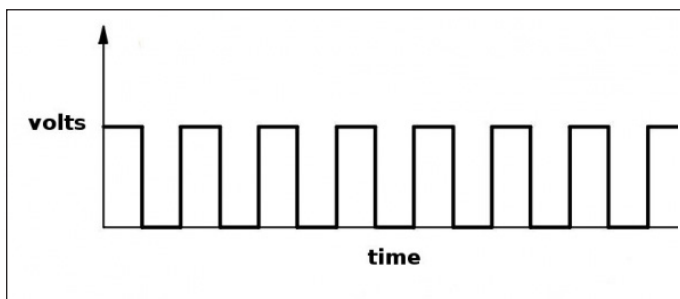
Как мы обсуждали ранее, пульт от телевизора оснащен ИК-светодиодом, который излучает ИК-свет с частотой 38 кГц. Итак, всякий раз, когда это излучение обнаруживается в приемной части телевизора (схема управления телевизора), нам необходимо отфильтровать этот сигнал.

Как вы можете видеть на следующем изображении, когда сигнал фактически передается с пульта дистанционного управления телевизора, он будет выглядеть как серия волн:



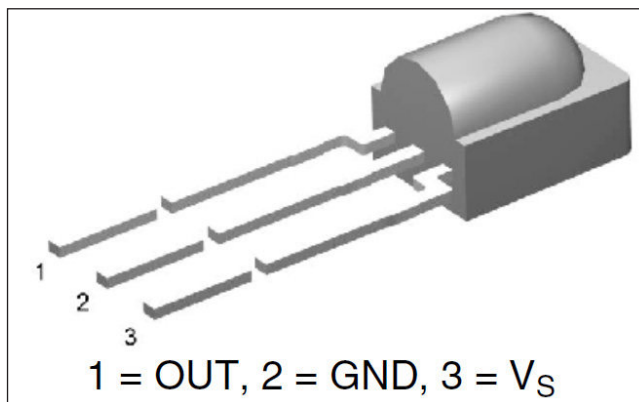
Для фильтрации реального сигнала с телевизионного пульта от окружающего шума мы будем использовать микросхему TSOP38238. TSOP38238 выглядит как транзистор, но на самом деле это устройство сочетает в себе ИК-чувствительный фотозлемент, полосовой фильтр 38 кГц и автоматический регулятор усиления. Здесь чувствительный к ИК-излучению фотозлемент работает как ИК-приемник (фотодиод), а полосовой фильтр 38 кГц требуется для сглаживания принятого модулированного сигнала.

После прохождения через полосовой фильтр 38 кГц выходной сигнал будет выглядеть, как показано ниже, что намного чище и легче для чтения:



Кроме того, TSOP38238 покрыт свинцовой эпоксидной рамой, которая действует как ИК-фильтр. Таким образом, мешающий свет (свет постоянного тока от вольфрамовой лампы или модулированный сигнал от люминесцентных ламп) не достигает фотодиода. Этот демодулированный сигнал может быть напрямую декодирован любым микропроцессором (в нашем случае Arduino). Из-за этого, кроме предусилителя, он игнорирует весь другой ИК-свет, если он не модулируется на определенной частоте (38 кГц).

Эта ИС настолько проста, насколько это возможно с тремя выводами. Есть два вывода питания и один вывод V_S для выходного сигнала:



TSOP38238 может питаться от 2,5 В до 5,5 В, что делает его пригодным для всех типов приложений. Кроме того, он может принимать сигналы практически от всех типов пультов дистанционного управления.

Для начала мы будем управлять одним светодиодом с помощью ИК-пульта. Для этого мы будем использовать библиотеку IRRemote, доступную по адресу <https://github.com/z3t0/Arduino-IRremote>. После того, как вы загрузили и извлекли код из папки библиотек Arduino, вы сможете увидеть примеры в Arduino IDE. Нам нужно записать значение всех кнопок для будущего использования.

Подключите первый вывод TSOP38238 к контакту 11 Arduino, второй вывод к контакту заземления Arduino, а третий вывод к контакту 5 В Arduino.

Подключите Arduino через USB и попробуйте скомпилировать следующий код:

```
#include <IRremote.h>
const int IR_RECEIVER = 11;
IRrecv receiver(IR_RECEIVER);
decode_results buttonPressed;
void setup()
{
  Serial.begin(9600);
  receiver.enableIRIn(); // Запускаем приемник
}

void loop()
{
  if (receiver.decode(&buttonPressed))
  {
    Serial.println(buttonPressed.value); //Печать значений, поступающих с ИК-пульта

    receiver.resume(); // Получаем следующее значение
  }
  delay(100);
}
```

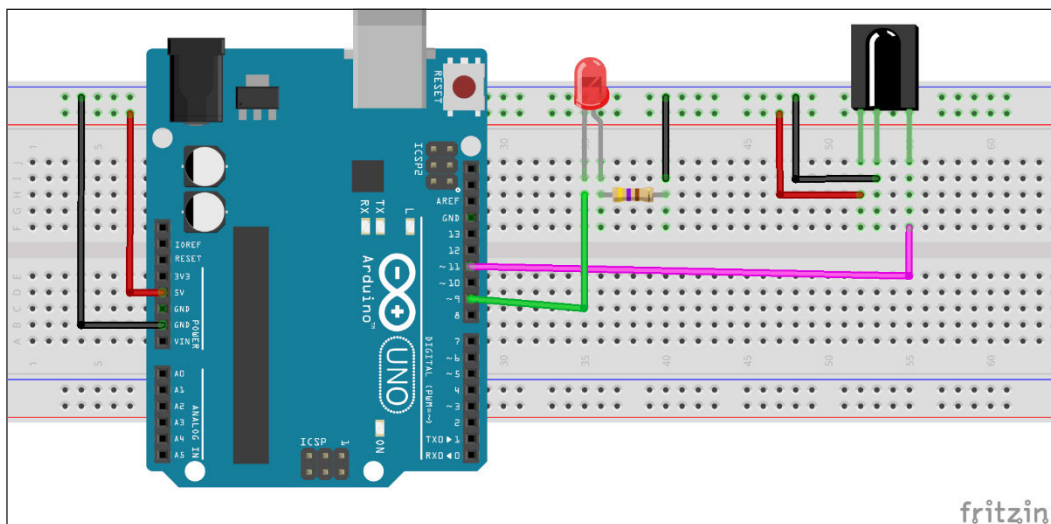


В последней версии Arduino библиотека IRremote.h уже установлена в папке RobotIRremote. Но у вас не будет всех примеров, доступных из загруженной библиотеки. Итак, удалите RobotIRremote и попробуйте снова скомпилировать код.

После удаления дубликата файла вы сможете успешно скомпилировать код. После загрузки предыдущего кода откройте МПП. Поскольку мы хотим управлять одним светодиодом с помощью ПДУ, нам необходимо знать значение нажатой кнопки. Итак, попробуйте записать значения всех кнопок одну за другой.

Проект 2 - Подсветка телевизора с дистанционным управлением

Например, цифра 1 содержит 54 528 кодов для определенного пульта ДУ. Возможно, вам придется проверить пульт, который у вас есть. Теперь мы будем управлять светодиодом с помощью ИК-пульта дистанционного управления. Вместе с ИК-приемником подключим один светодиод, как показано на следующей схеме: one LED, as shown in the following circuit:



Обновите код для управления светодиодом, основываясь на ваших показаниях из предыдущего упражнения:

```
#include <IRremote.h>
const int IR_RECEIVER = 11;
IRrecv receiver(IR_RECEIVER);
decode_results buttonPressed;
long int buttonValue = 0;
const long int buttonOne = 54528; //Обновляем одно значение в соответствии с
вашиими показаниями
readings
const long int buttonTwo = 54572; //Обновляем значение два в
соответствии с вашими показаниями

int LEDpin = 9;

void setup()
{
  Serial.begin(9600);
  receiver.enableIRIn(); // Запускаем приемник
  pinMode(LEDpin, OUTPUT);
}

void loop()
{
```



```
if (receiver.decode(&buttonPressed))
{
  buttonValue = buttonPressed.value;
  Serial.println(buttonValue);
  switch (buttonValue){
    case buttonOne:
      digitalWrite(LEDpin,HIGH);
      break;
    case buttonTwo:
      digitalWrite(LEDpin,LOW);
      break;
    default:
      Serial.println("Waiting for input. ");
  }
  receiver.resume(); // Получаем следующее значение
}
delay(100);
}
```

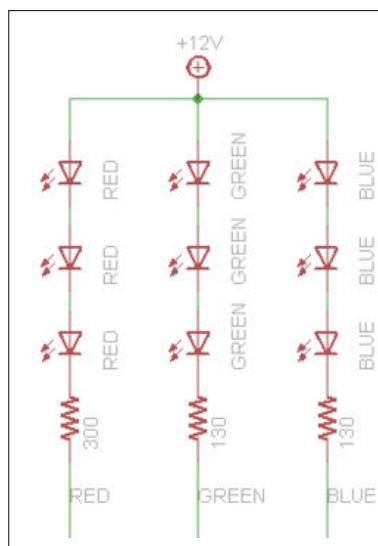
Теперь вы сможете управлять одним светодиодом с помощью ИК-пульта. Точно так же вы можете управлять всем, чем хотите, с помощью ИК-пульта. В конце этой главы мы хотим управлять подсветкой телевизора с помощью ИК-пульта дистанционного управления. Разработаем подсветку телевизора с помощью светодиодной ленты. Итак, теперь мы узнаем, как управлять светодиодной матрицей.

Светодиодные ленты

Светодиодные ленты представляют собой гибкие печатные платы с припаянными на них полноцветными светодиодами. Есть два основных типа светодиодных лент: аналоговые и цифровые. Аналоговые полоски имеют все светодиоды, подключенные параллельно. Таким образом, они действуют как один огромный трехцветный светодиод. В случае аналоговой светодиодной ленты мы не можем установить цвет / яркость каждого светодиода. Таким образом, они просты в использовании и недороги. Цифровые светодиодные ленты работают по-другому. Чтобы использовать светодиод, мы должны отправить цифровой код, соответствующий каждому светодиоду в случае цифровой светодиодной ленты. Поскольку они обладают большей модульностью, они довольно дороги. Также цифровые светодиодные ленты сложнее использовать по сравнению с аналоговыми светодиодными лентами:



Внутри светодиоды RGB подключены друг к другу параллельно. Одна часть ленты содержит все три светодиода, включенных параллельно. Полная светодиодная лента состоит из нескольких параллельно соединенных последовательно светодиодов RGB. Подключение в одном блоке / секции показано на следующем рисунке:



Поскольку одна секция содержит несколько светодиодов, для ее работы требуется больше тока. Он не может работать на токе, предоставляемом Arduino. В случае полностью белого цвета для каждого сегмента потребуется приблизительно 60 мА тока на блок / секцию. Таким образом, для работы всей светодиодной ленты нам может потребоваться ток до 1,2 А на метр. Для запуска светодиодной ленты требуется внешний источник питания 12 В.

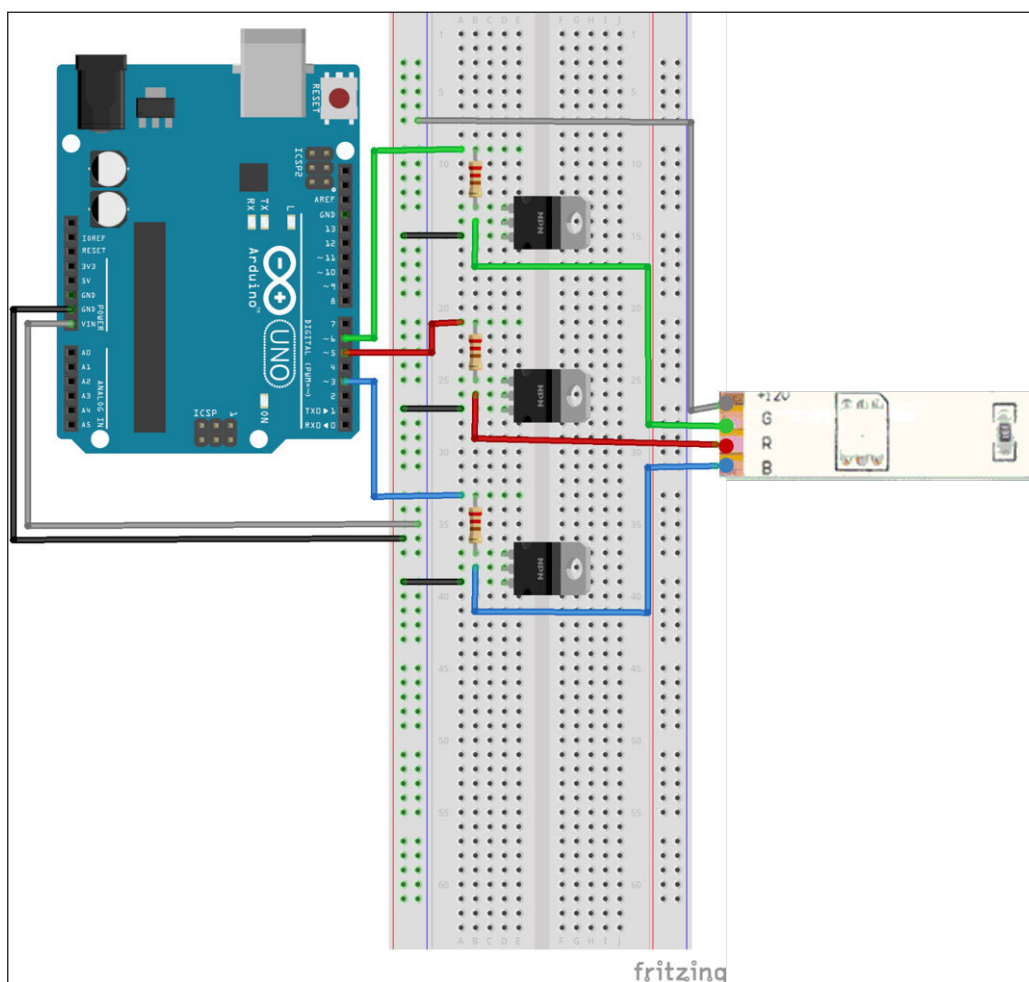
Управление светодиодной лентой с помощью Arduino

Как мы знаем, мы можем потреблять максимальный ток до 300 мА, если соединим все контакты ввода / вывода вместе. Но, поскольку нам нужен более большой ток, нам придется использовать внешний источник питания.

А теперь самое сложное. Мы будем использовать Arduino для управления светодиодной лентой, а для источника питания - внешний источник питания. Для подключения светодиодной ленты к Arduino мы будем использовать транзистор MOSFET для ограничения тока, потребляемого Arduino и LED лентой.

Здесь мы будем использовать N-канальный MOSFET IRF540, который недорог и работает слогикой от 3,3 В до 5 В. Эти полевые транзисторы могут переключать более 60 А и 30В.

Подключиться к полосе относительно просто. Нам нужно припаять провода к четырем контактам / медным контактным площадкам нашей светодиодной ленты. Можно использовать термоусадку для обеспечения изоляции, устойчивости к истиранию и защиты окружающей среды. Чтобы использовать нашу светодиодную ленту с Arduino, нам также потребуются резисторы для ограничения тока:



Подключите силовой транзистор с резистором между выходным контактом ШИМ и базой. В случае транзисторов NPN вывод 1 является базовым. Вывод 2 - коллектор, а вывод 3 - эмиттер. Теперь для подачи питания на Arduino и светодиодную ленту подключите источник питания 9-12 В к Arduino, чтобы Vin подал высокое напряжение на светодиод.

В конце подключения проводки обязательно подключите заземление источника питания к заземлению Arduino / MOSFET. Подключите выходной контакт Arduino к базе полевого МОП-транзистора (контакт 1) с резистором 100-220 Ом между ними. Подключите контакт 2 полевого МОП-транзистора к входному контакту светодиодной ленты и подключите контакт 3 полевого МОП-транзистора к земле. Проверьте все соединения и напишите следующий код в окне редактора Arduino. Здесь мы используем контакты 5, 6 и 3 Arduino для управления красным, зеленым и синим светодиодами светодиодной ленты соответственно:

```
int redLed = 5;
int greenLed = 6;
int blueLed = 3;

int redValue = 0;
int greenValue = 0;
int blueValue = 0;

void setup() {
  randomSeed(analogRead(0));
}

void loop() {
  redValue = random(0,256); // Произвольно генерируем от 1 до 255
  greenValue = random(0,256); // Произвольно генерируем от 1 до 255
  blueValue = random(0,256); // Произвольно генерируем от 1 до 255

  analogWrite(redLed,redValue);
  analogWrite(greenLed,greenValue);
  analogWrite(blueLed,blueValue);

  // Увеличиваем все значения одно за другим после установки случайных
  значений.
  for(redValue = 0; redValue < 255; redValue++){
    analogWrite(redLed,redValue);
    analogWrite(greenLed,greenValue);
    analogWrite(blueLed,blueValue);
    delay(10);
  }
  for(greenValue = 0; greenValue < 255; greenValue++){
    analogWrite(redLed,redValue);
    analogWrite(greenLed,greenValue);
    analogWrite(blueLed,blueValue);
```

```
        delay(10);
    }
    for(blueValue = 0; blueValue < 255; blueValue++){
        analogWrite(redLed,redValue);
        analogWrite(greenLed,greenValue);
        analogWrite(blueLed,blueValue);
        delay(10);
    }

    //Уменьшение всех значений по одному для выключения всех
    светодиодов.
    for(redValue = 255; redValue > 0; redValue--){
        analogWrite(redLed,redValue);
        analogWrite(greenLed,greenValue);
        analogWrite(blueLed,blueValue);
        delay(10);
    }
    for(greenValue = 255; greenValue > 0; greenValue--){
        analogWrite(redLed,redValue);
        analogWrite(greenLed,greenValue);
        analogWrite(blueLed,blueValue);
        delay(10);
    }
    for(blueValue = 255; blueValue > 0; blueValue--){
        analogWrite(redLed,redValue);
        analogWrite(greenLed,greenValue);
        analogWrite(blueLed,blueValue);
        delay(10);
    }
}
```

Если все на месте, вы должны увидеть, как светодиодная лента включается и меняет свой цвет.

Теперь мы узнали обо всем, что необходимо для управления подсветкой телевизора с помощью ИК-пульта дистанционного управления, поэтому мы объединим все, что мы узнали в этой главе:

- Как работает IR
- Как считывать значения с ИК-ПДУ
- Как управлять светодиодной лентой

Мы также хотим управлять яркостью светодиодной ленты. Мы будем использовать кнопку питания, чтобы включать и выключать подсветку. С громкостью плюс и громкость минус будем увеличивать и уменьшать яркость подсветки.

Как мы делали ранее в этой главе, подключите TSOP38238 к контакту 11(5 В) и контакту заземления Arduino. После того, как вы выполнили все подключения, загрузите следующий код в Arduino:

```
#include <IRremote.h>
const int IR_RECEIVER = 11; // Подключаем выходной контакт TSOP38238 к контакту 11
IRrecv receiver(IR_RECEIVER);
decode_results buttonPressed;
long int buttonValue = 0;
```

Упоминаем коды, полученные в предыдущем упражнении

```
// Кнопка питания для включения или выключения
const long int PLUS_BUTTON = 54536; // Увеличиваем яркость LED ленты

const long int MINUS_BUTTON = 54608; // Уменьшаем яркость LED ленты

const long int CHANGE_COLOR = 54584; // Уменьшаем яркость LED ленты

const int FADE_AMOUNT = 5; // Для быстрого увеличения /
уменьшения яркости увеличьте это значение
boolean isOn = false;

int redLed = 5;
int greenLed = 6;
int blueLed = 3;

int redValue = 0;
int greenValue = 0;
int blueValue = 0;

int colors[3];

// Включаем светодиодную ленту случайным цветом
void powerUp(int *colors)
{
    redValue = random(0, 256); // Произвольно генерируем от 1 до 255
    greenValue = random(0, 256); // Произвольно генерируем от 1 до 255
    blueValue = random(0, 256); // Произвольно генерируем от 1 до 255

    analogWrite(redLed, redValue);
    analogWrite(greenLed, greenValue);
```

```
    analogWrite(blueLed, blueValue);

    colors[0] = redValue;
    colors[1] = greenValue;
    colors[2] = blueValue;
}

// Выключаем светодиод
void powerDown(int *colors)
{
    redValue = colors[0];
    greenValue = colors[1];
    blueValue = colors[2];

    //Уменьшение всех значений одно за другим для выключения всех
    светодиодов.
    for (; redValue > 0; redValue--) {
        analogWrite(redLed, redValue);
        delay(10);
    }
    for (; greenValue > 0; greenValue--) {
        analogWrite(greenLed, greenValue);
        delay(10);
    }
    for (; blueValue > 0; blueValue--) {
        analogWrite(blueLed, blueValue);
        delay(10);
    }
    colors[0] = redValue;
    colors[1] = greenValue;
    colors[2] = blueValue;
}

void increaseBrightness(int *colors)
{
    redValue = colors[0];
    greenValue = colors[1];
    blueValue = colors[2];

    redValue += FADE_AMOUNT;
    greenValue += FADE_AMOUNT;
    blueValue += FADE_AMOUNT;

    if (redValue >= 255) {
```

```
    redValue = 255;
}

if (greenValue >= 255) {
    greenValue = 255;
}

if (blueValue >= 255) {
    blueValue = 255;
}
analogWrite(redLed, redValue);
analogWrite(greenLed, greenValue);
analogWrite(blueLed, blueValue);

colors[0] = redValue;
colors[1] = greenValue;
colors[2] = blueValue;
}

void decreaseBrightness(int *colors)
{
    redValue = colors[0];
    greenValue = colors[1];

    blueValue = colors[2];

    redValue -= FADE_AMOUNT;
    greenValue -= FADE_AMOUNT;
    blueValue -= FADE_AMOUNT;

    if (redValue <= 5) {
        redValue = 0;
    }

    if (greenValue <= 5) {
        greenValue = 0;
    }

    if (blueValue <= 5) {
        blueValue = 0;
    }
    analogWrite(redLed, redValue);
    analogWrite(greenLed, greenValue);
}
```



```
analogWrite(blueLed, blueValue);

colors[0] = redValue;
colors[1] = greenValue;
colors[2] = blueValue;
}

// Случайная генерация цвета и плавный переход к этому цвету
void changeColor(int *colors)
{
    int newRedValue = random(0, 256); // Произвольно генерируем от 1 до 255
    int newGreenValue = random(0, 256); // Произвольно генерируем от 1 до 255
    int newBlueValue = random(0, 256); // Произвольно генерируем от 1 до 255

    redValue = colors[0];
    greenValue = colors[1];
    blueValue = colors[2];

    if (newRedValue > redValue) {
        for (; redValue >= newRedValue; redValue++) {
            analogWrite(redLed, redValue);
            delay(10);
        }
    }
    else {
        for (; redValue <= newRedValue; redValue--) {
            analogWrite(redLed, redValue);
            delay(10);
        }
    }

    if (newGreenValue > greenValue) {
        for (; greenValue >= newGreenValue; greenValue++) {
            analogWrite(greenLed, greenValue);
            delay(10);
        }
    }
    else {
        for (; greenValue <= newGreenValue; greenValue--) {
            analogWrite(greenLed, greenValue);
            delay(10);
        }
    }
}
```

```
    if (newBlueValue > blueValue) {
        for (; blueValue >= newBlueValue; blueValue++) {
            analogWrite(blueLed, blueValue);
            delay(10);
        }
    }
    else {
        for (; blueValue <= newBlueValue; blueValue--) {
            analogWrite(blueLed, blueValue);
            delay(10);
        }
    }

    colors[0] = redValue;
    colors[1] = greenValue;
    colors[2] = blueValue;
}

void setup() {
    Serial.begin(9600);
    receiver.enableIRIn(); // Запустить приемник

    randomSeed(analogRead(0));

    pinMode(redLed, OUTPUT);
    pinMode(greenLed, OUTPUT);
    pinMode(blueLed, OUTPUT);
}

void loop() {

    if (receiver.decode(&buttonPressed))
    {
        buttonValue = buttonPressed.value;
        Serial.println(buttonValue);
        switch (buttonValue) {
            case POWER_BUTTON:
                if (!isOn) {
                    powerUp(colors);
                    isOn = true;
                }
                else {
                    powerDown(colors);
                }
            }
        }
    }
}
```

```
        isOn = false;
    }
    break;
case PLUS_BUTTON:
    decreaseBrightness(colors);
    break;
case MINUS_BUTTON:
    increaseBrightness(colors);
    break;
case CHANGE_COLOR:
    changeColor(colors);
    break;
default:
    Serial.println("Waiting for input.
                   (Ожидание ввода) ");
}
receiver.resume(); // Получаем следующее значение
}
delay(100);
}
```

В приведенном выше коде мы используем кнопку питания для включения и выключения подсветки, а кнопки увеличения и уменьшения громкости - для увеличения и уменьшения яркости соответственно. Я использовал кнопку со стрелкой вверх, чтобы изменить цвет LED ленты. Вы можете добавить больше функций в этот проект, настроив его в блоке `switch case`.

Резюме

В этой главе мы начали с основ ИК-светодиодов и ИК-связи. После этого мы узнали о программировании ИК-датчиков и их применении. К концу этой главы мы узнали об управлении светодиодной лентой и завершили нашу главу разработкой подсветки телевизора с дистанционным управлением.

В следующих главах мы начнем с более сложных вещей, таких как разработка светодиодного куба, звуковая визуализация и постоянное зрение.

4

Проект 3 – Светодиодный куб

Если вы успешно реализовали два последних проекта, вы бы заметили, что мы обходимся без пайки. Тем не менее, я бы сказал, что вы не работали с электроникой, если вы не паяли платы. В этой главе вы подробно познакомитесь с пайкой. Вы также поймете, как создать светодиодный куб $4 * 4 * 4$ с помощью платы Arduino UNO. Вы что вы узнаете:

- Введение в пайку
- Разработка светодиодного куба
- Программирование светодиодного куба $4 * 4 * 4$

Начало работы с пайкой

Пайка - это процесс создания надежного электрического и механического соединения между некоторыми металлами путем соединения их мягким припоем. Это сплав свинца и олова с низкой температурой плавления. Стык нагревается паяльником до нужной температуры. Для эффективной пайки требуется хорошая передача тепла отпаяльника к паяемым компонентам. Чем дольше применяется нагрев, тем выше риск теплового повреждения компонента, поэтому важно быстро завершить работу.

Что вам понадобится

Прежде чем переходить к следующему разделу главы, убедитесь, что у вас есть следующие инструменты:

- Паяльник
- Подставка для паяльника
- Приспособление для удаления припоя
- Картон

Следующий рисунок представлен необходимым набором инструментов:



Советы по технике безопасности

Пайка представляет различные опасности, поэтому всегда следуйте этим советам безопасности при пайке:

- Не прикасайтесь к жалу паяльника.
- Не прикасайтесь паяльником к сетевому шнуру.

- Всегда возвращайте паяльник на подставку, когда он не используется.
- Не кладите паяльник на рабочий стол!
- Работайте в хорошо вентилируемом помещении.
- Дым, образующийся при плавлении припоя, в основном возникает из-за флюса и вызывает сильное раздражение. Избегайте вдыхания.
- Мойте руки после использования припоя.
- Припой содержит свинец, который является ядовитым металлом.
- Никогда не паяйте схему под напряжением.

Конструирование светодиодного куба

Как упоминалось ранее, основное внимание в этой главе уделяется пайке. В этом разделе вы узнаете, как спроектировать светодиодный куб, где будет много пайки и креативные элементы, такие как управление светодиодами и программирование Arduino.

Необходимые компоненты

Прежде чем приступить к проектированию куба, убедитесь, что у вас есть следующие компоненты для этого проекта:

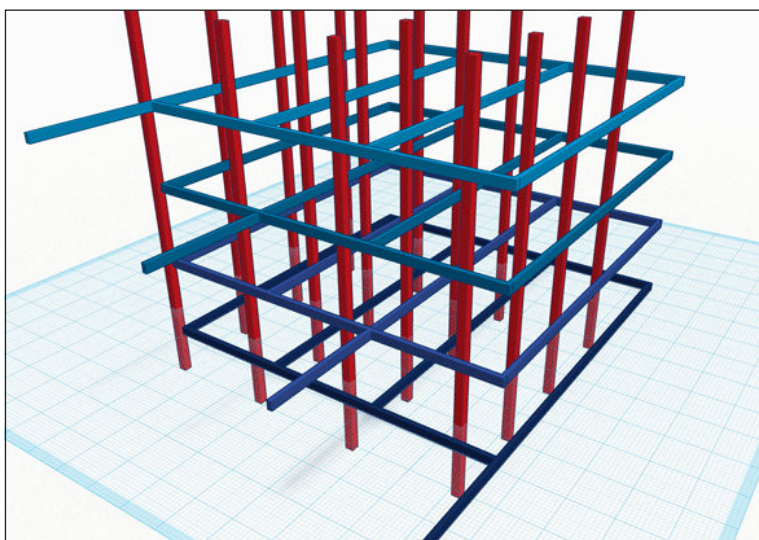
- Arduino UNO
- 64 светодиода: вы можете использовать любой цветной светодиод. Хотя для этого проекта требуется 64 светодиода, я бы порекомендовал вам купить не менее 100 светодиодов на случай, если некоторые светодиоды сгорят в процессе пайки.
- 16 резисторов: они должны соответствовать вашим светодиодам. Если вы не уверены какой резистор купить, возьмите резисторы 500 Ом / 1 кОм.
- Соединительные провода
- Печатная плата
- Пенопласт
- Паяльник и припой

Принцип, лежащий в основе изготовления куба

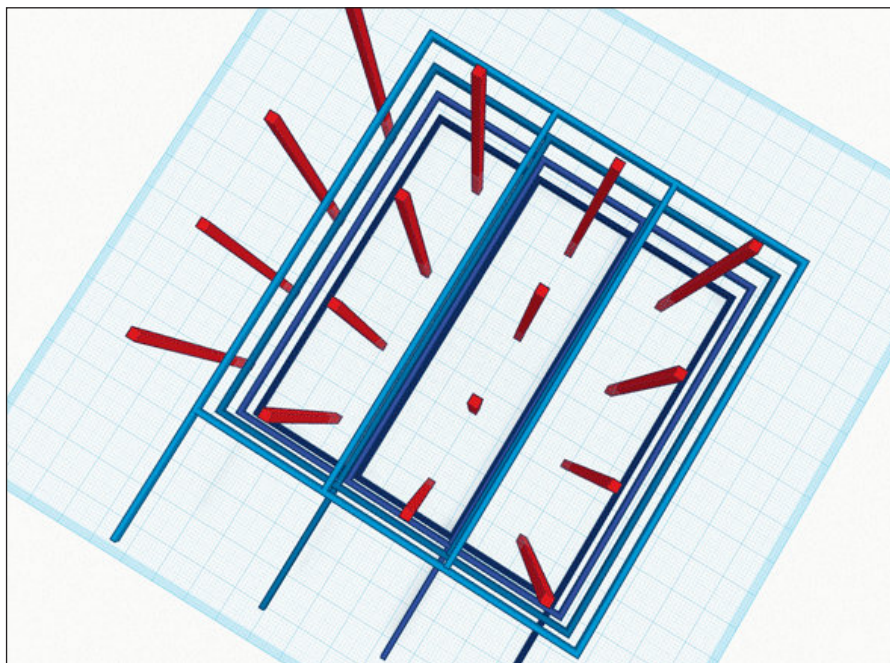
Прежде чем приступить к проекту, убедитесь, что у вас есть перечисленные компоненты. Этот раздел является наиболее важной частью этой главы, поскольку он объясняет ключевой принцип, лежащий в основе проекта. Он дает полный обзор системы. В Интернете вы найдете множество руководств по созданию светодиодного куба, однако в большинстве из них будет использоваться один выходной контакт для каждого отдельного светодиода. Если вы используете этот подход для светодиодного куба $4 * 4 * 4$, вам понадобится 64 контакта, которых нет в Arduino UNO. Один из подходов - использовать регистры сдвига.

Если вы посмотрите на плату Arduino UNO, вы заметите, что максимум можно использовать 20 контактов. Таким образом, для управления всеми этими светодиодами на 20 выводах будет использоваться метод мультиплексирования. Если вы разделите куб на четыре слоя, вам понадобится 16 управляющих выводов для обращения к отдельным светодиодам. Чтобы включить конкретный светодиод куба, необходимо активировать все слои. Всего для этого проекта вам понадобится $16 + 4 = 20$ контактов.

Для каждого слоя будет общий катод (отрицательный вывод), поэтому все отрицательные выводы светодиодов подключены к одному выводу для этого слоя. Что касается анода (положительный вывод), каждый светодиод будет подключен к соответствующему слою выше и ниже. Таким образом, у вас будет 16 столбцов положительных выводов и четыре слоя отрицательных выводов. Ниже приведены некоторые трехмерные изображения конструкции, которые помогут вам лучше понять ее. Красные - положительные клеммы, синие - отрицательные:



Вот еще одно изображение

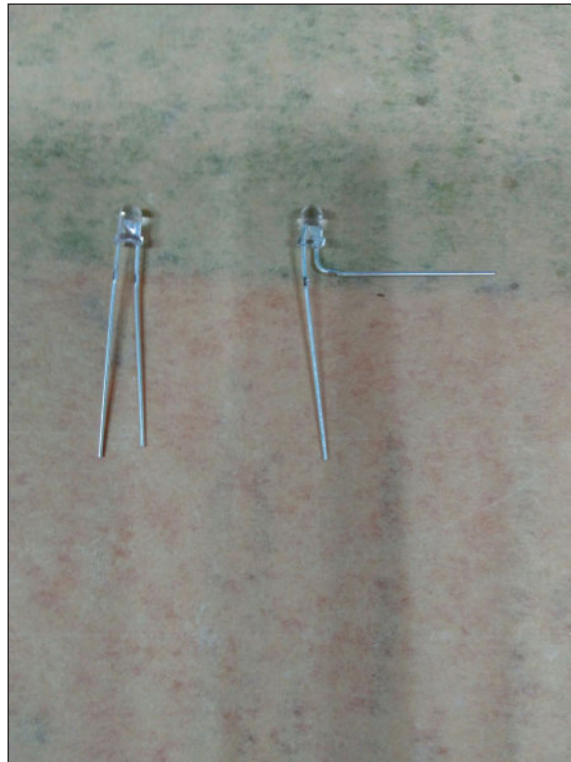


Изготовление

Для конструкции вы можете использовать полностью металлическую конструкцию, чтобы придать конструкции жесткость за счет простоты.

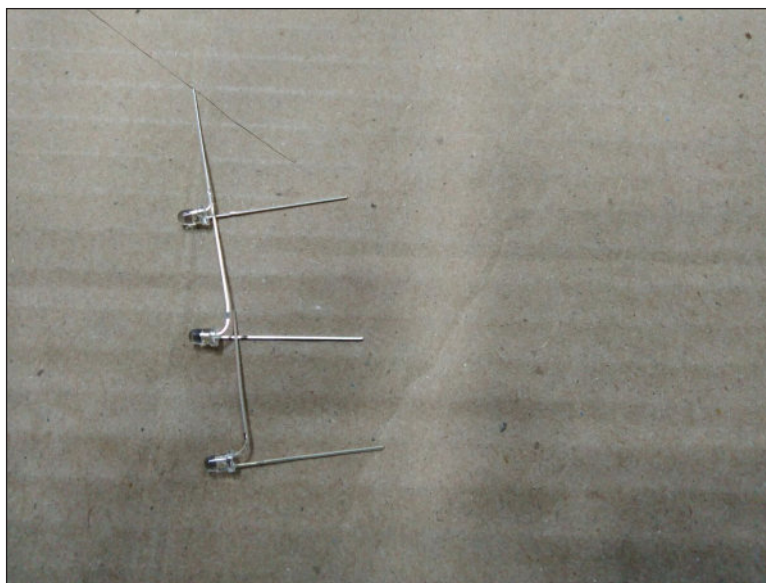


Вы можете сгибать катод влево или вправо. Просто убедитесь, что они одинаковы по всей длине и не касаются анода

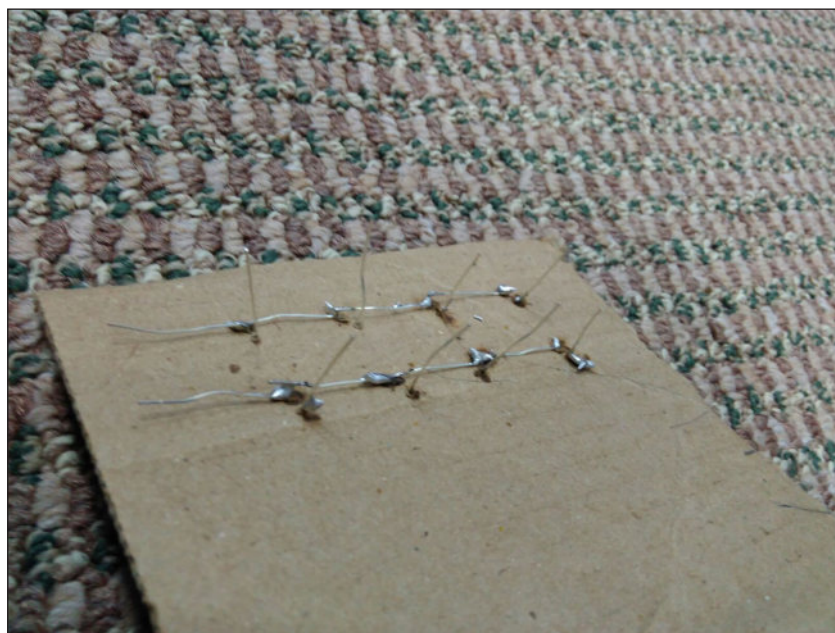


Одна из важнейших частей этого проекта - создание конструкции для удержания светодиода во время пайки. Сделать такую конструкцию можно, сделав деревянную подставку или используя пенопласт . Вариант с пенопластом будет немного проще. Вот несколько вещей, которые вы должны помнить при работе над этой частью:

- Четверть ножки светодиода должна перекрывать соседний светодиод. При необходимости воспользуйтесь линейкой.
- Если вы используете деревянное приспособление, убедитесь, чтобы отверстие в ней было немного больше корпуса светодиода, чтобы вы смогли удалить полностью припаянный слой.



Теперь припаиваем катоды четырех рядов светодиодов. После заполнения первых четырех рядов ваша структура будет выглядеть примерно так, как на следующем изображении:



На предыдущем изображении отображаются только два ряда слоев. Если вы работаете с медными светодиодными кабелями, вы должны знать, что конструкции, созданные с использованием медных кабелей, не будут такими прочными. Чтобы усилить жесткость слоя, отрежьте и припаяйте два прямых отрезка проволоки с обоих концов. Убедитесь, что они соединяются с каждым рядом. Вот и все. Ваш первый слой готов.

Перед пайкой других слоев вы должны протестировать слой, который вы только что создали, и исправить любые ошибки, если вы их допустили, чтобы вы не повторили ту же ошибку при работе с другими слоями. Для этого откройте Arduino IDE и загрузите код мигания, которое вы написали в главе 1 «[Начало работы со светодиодами Arduino](#)». Для тестирования каждого светодиода подключите земляной контакт к рамке слоя с подключенным резистором и прижмите положительный провод к каждому светодиоду.

Если все пойдет хорошо, загорится каждый светодиод. Если нет, проверьте соединение этого светодиода и убедитесь, что вы где-то не пропустили паяное соединение. Если пайка и соединение в порядке, замените светодиод и снова проверьте его. Есть вероятность, что светодиоды могли сгореть, если вы долго держали паяльник подключенным к ним.

После того, как вы протестировали первый слой, удалите этот слой с деревянного приспособления / пенопласта и повторите процесс еще три раза. Не забудьте протестировать все четыре слоя светодиодов с помощью программы мигания.

После того, как вы закончили и протестировали четыре слоя, вам нужно соединить все вертикальные ножки вместе. Один из приемов - вырезать кусок картона. Это поможет сохранить все слои на одной высоте. Вы заметите, что даже после этого есть много ножек, которые не совпадают идеально. Другой вариант - медными проводами припаять их в тех местах, где ножки не выровнены должным образом.

Ошибки, которых следует избегать

Когда вы будете работать с этим проектом, иногда все может стать довольно сложным. Вот несколько вещей, о которых следует помнить, поскольку большинство новичков делают с этим ошибки:

- Если вы используете картон для получения одинаковой высоты для всех слоев, сделайте его длиннее сбоку и соедините части картона за пределами куба, чтобы, когда вы закончите слой, вы могли легко его вытащить.
- Общее практическое правило заключается в том, что анод светодиода соединяется с анодом другого светодиода, и аналогично катод светодиода соединяется с катодом другого светодиода. Не соединяйте / не спаивайте анод одного светодиода с катодом другого светодиода.

Убедитесь, что вы проверили все четыре слоя после их соединения. Самый простой способ проверить - прикоснуться только к самому верхнему слою. Если это работает, значит, у вас хорошее соединение на всех уровнях.

Не забудьте вырезать лишние кусочки металлического каркаса и ножек, прежде чем подключать их к монтажной плате или картону. Здесь важно учитывать, что нельзя вырезать вертикальные ножки, поскольку их нужно вставить в макетную плату.

Крепление к плате

Прежде чем перейти к части Arduino, последний шаг - закрепить эту структуру на макетной плате. Если вы хотите создать макетную плату, убедитесь, что вы разместили каждый светодиод на одинаковом расстоянии, а для удержания каждой ножки светодиода вы можете использовать несколько зажимов "крокодил".



Лично мне это показалось немного трудным, так как после того, как вы присоединились ко всем четырем слоям, все ножки светодиода не будут находиться на одинаковом расстоянии, поэтому вместо этого я использовал картонный лист и припаял светодиоды на картон.

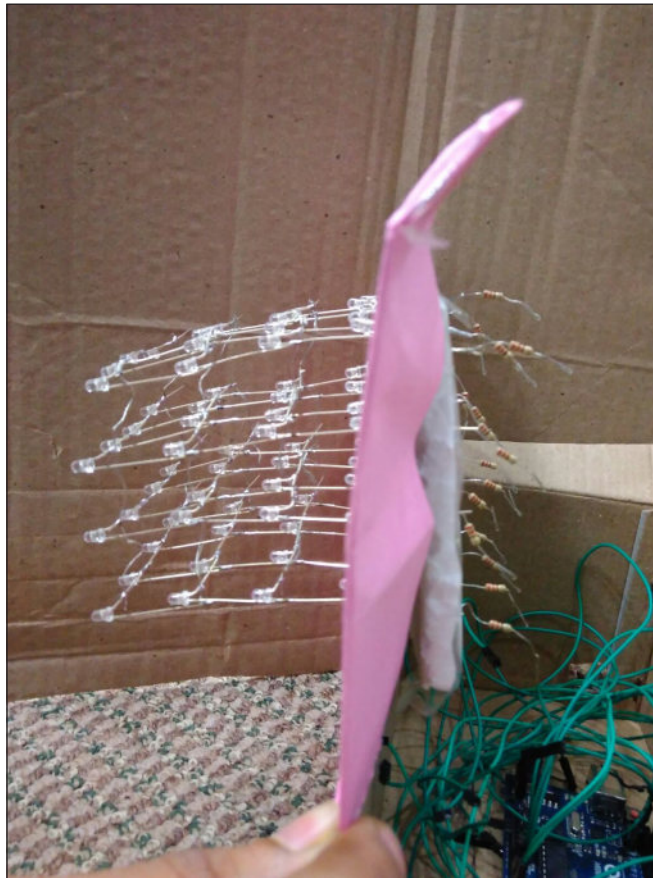
Следующим шагом будет соединение светодиодов с резисторами. В идеале резисторы должны быть припаяны к макетной плате.



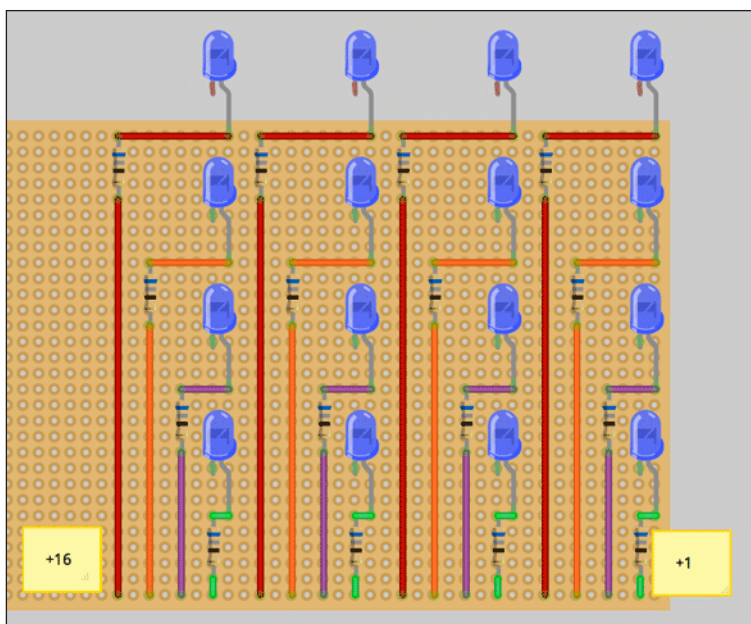


Если вы используете картон, как я, вы можете использовать пенопласт для подключения резистора к светодиоду.

После того, как вы соединили резисторы с кубом, он будет выглядеть, как показано на следующем рисунке. Обратите внимание, для этого прототипа я использовал пенопласт и картон. Если вы используете деревянную основу и макетную плату, ваш прототип будет выглядеть гораздо более аккуратным:



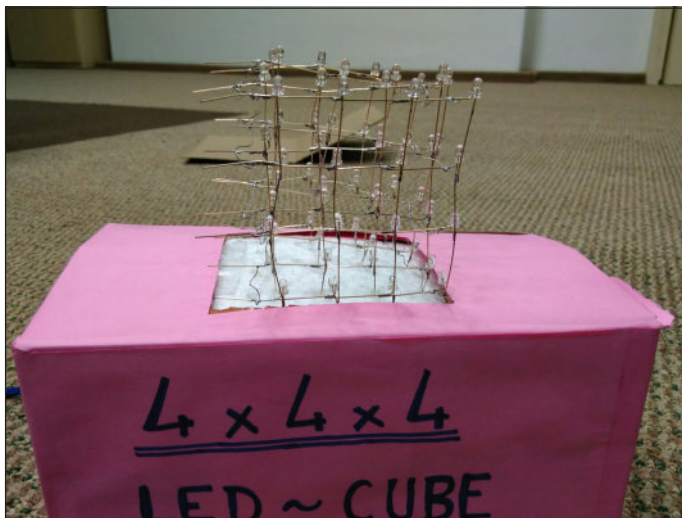
Если вы не планировали размещение резистора заранее, вот что у вас будет, когда вы выполните все шаги. Лучше всего расположить их одинаково ступенчато, чтобы затем вы могли использовать одну целую сторону куба для всех окончательных подключений к Arduino. Вот принципиальная схема:



Наконец, подключите несколько соединительных проводов, которые можно подключить к соответствующим контактам *Arduino*, и убедитесь, что вы используете для этого длинные провода. Вы можете использовать цветовой код для различения соединительных кабелей:

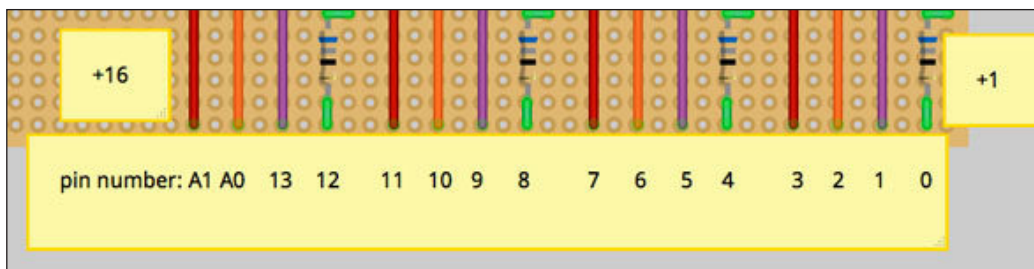


Если у вас возникли проблемы с получением кабелей разного цвета, и в конце концов вы почувствуете, что пайка и соединения будут немного беспорядочными, накройте куб изнутри картонной коробкой, как показано на следующем рисунке:



Программирование светодиодного куба $4 * 4 * 4$

Завершив сложную часть пайки, давайте перейдем к подключению и программированию Arduino. Перед подключением положительных выводов подключите четыре отрицательных слоя к аналоговым портам ввода / вывода Arduino с A2 (нижний слой) по A5 (верхний слой). После этого к плате Arduino необходимо подключить 16 выводов управления светодиодами. Подключите первые 14 контактов к цифровым портам ввода / вывода Arduino с 0 по 13. Остальные контакты 15 и 16 необходимо подключить к аналоговым контактам A0 и A1. См. Следующую схему для справки по подключению:



Перед программированием куба вам следует понять несколько вещей:

- Для адресации одного светодиода используйте номер плоскости (слоя) 0–3 и контактный номер светодиода 0–15. Поверните плоскость на выход LOW (отрицательная ветвь), а номер контакта светодиода на HIGH (положительная ветвь), чтобы активировать светодиод.

Перед активацией одного светодиода убедитесь, что все остальные плоскости выключены, установив их выход как HIGH. Если этого не сделать, вместо одного светодиода загорится весь столбец светодиодов.

Теперь вы готовы приступить к программированию.

Скопируйте код [ledcube2.ino](#) из папки с кодами и загрузите его на вашу плату Arduino:

```
#include <avr/pgmspace.h> // позволяет использовать PROGMEM для
хранения шаблонов во флеш-памяти

#define CUBESIZE 4
#define PLANESIZE CUBESIZE*CUBESIZE
#define PLANETIME 3333 // время отображения каждой плоскости в us ->
обновление 100 Гц
#define TIMECONST 5 // умножает DisplayTime, чтобы получить мс

/*
** Определение контактов в массиве упрощает изменение порядка
подключения куба. Отрегулируйте числа здесь, пока светодиоды не начнут
мигать по порядку - от L к R, T к B
** Обратите внимание, что аналоговые входы 0-5 также являются цифровыми
выходами 14-19!
** Выводы DigitalOut0 (последовательный RX) и AnalogIn5 оставлены
открытыми для будущих приложений.
*/int LEDPin[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15};
int LEDPinCount = 16;
int PlanePin[] = {16, 17, 18, 19};
int PlanePinCount = 4;

// инициализация
void setup()
{
  int pin; //счетчик цикла
  // устанавливаем выводы светодиода как выход (активный ВЫСОКИЙ)
  for (pin=0; pin<PLANESIZE; pin++) {
    pinMode( LEDPin[pin], OUTPUT );
```

```
    }
    // устанавливаем контакты плоскости как выходы (активный LOW)
    for (pin=0; pin<CUBESIZE; pin++) {
        pinMode( PlanePin[pin], OUTPUT );
    }
}

void loop(){
    loopFor();
}

// принципы использования 4-х плоскостей и 16-ти пинов - здесь мы
// перебираем каждую плоскость, включаем и выключаем по очереди
void loopFor()
{
    for(int thisPlane = 0; thisPlane < PlanePinCount; thisPlane++){
        for(int thisPin = 0; thisPin < LEDPinCount; thisPin++){

            planesOff();
            digitalWrite(LEDPin[thisPin],HIGH);
            digitalWrite(PlanePin[thisPlane],LOW);

            delay(50);

            digitalWrite(LEDPin[thisPin],LOW);
            digitalWrite(PlanePin[thisPlane],HIGH);

        }
    }
}

void planesOff(){
    for(int thisPlane = 0; thisPlane < PlanePinCount; thisPlane++){
        digitalWrite(PlanePin[thisPlane],HIGH);
    }
}
```

Наконец, подключите несколько соединительных проводов, которые можно подключить к соответствующим контактам Arduino, и убедитесь, что вы используете для этого длинные провода.

Вы можете использовать цветовой код для различения соединительных проводов:

Резюме

До сих пор в этой книге основное внимание уделялось пониманию программирования Arduino, а не электронике и пайке. Если вы дошли до конца этой главы, это означает, что вы усвоили очень важный навык - пайку. Разобравшись в пайке и программировании Arduino, теперь вы можете воплотить свои собственные идеи в реальность / прототип. Поделитесь своим прототипом и опытом в социальных сетях, используя хэштег #ArduinoBLINK.

Получив хорошие знания о программировании и пайке Arduino, в следующей главе вы узнаете о визуализации звука и о том, как использовать различные датчики с Arduino.

Звуковая визуализация и светодиоды Рождественская елка

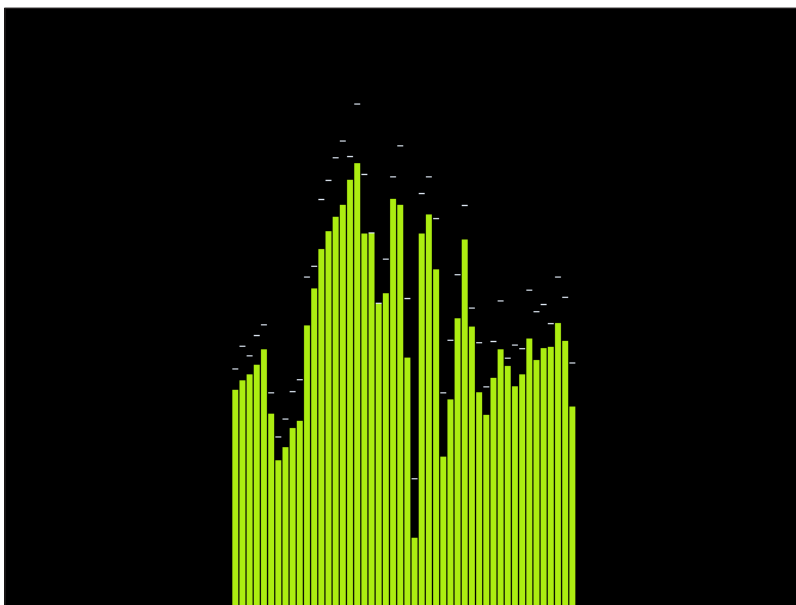
5

Все становится довольно просто, если вы понимаете основы Arduino и то, как управлять «вещами» с помощью Arduino. В предыдущих главах мы разработали несколько полезных проектов с использованием светодиодов и датчиков света. Мы также узнали о пайке в предыдущей главе. В этой главе мы поймем, как визуализировать звук, используя Arduino, а затем мы изготовим светодиодную елку.

- Введение в звуковую визуализацию
- Звуковая визуализация с использованием Arduino
- Создание рождественской елки с управляемым звуком.

Введение в звуковую визуализацию

Звуковая визуализация, или визуализация музыки, была неотъемлемой частью музыкальной индустрии с момента появления медиаплееров. Например, на вашем компьютере, когда вы воспроизводите любую музыку, вы можете видеть визуализацию в медиаплеере по умолчанию или медиаплеере VLC, как показано на следующем изображении:



Вы заметили, что при изменении громкости музыки или частоты звука меняется и визуализация. Мы будем использовать тот же принцип для визуализации звука с помощью Arduino.

Как визуализировать звук

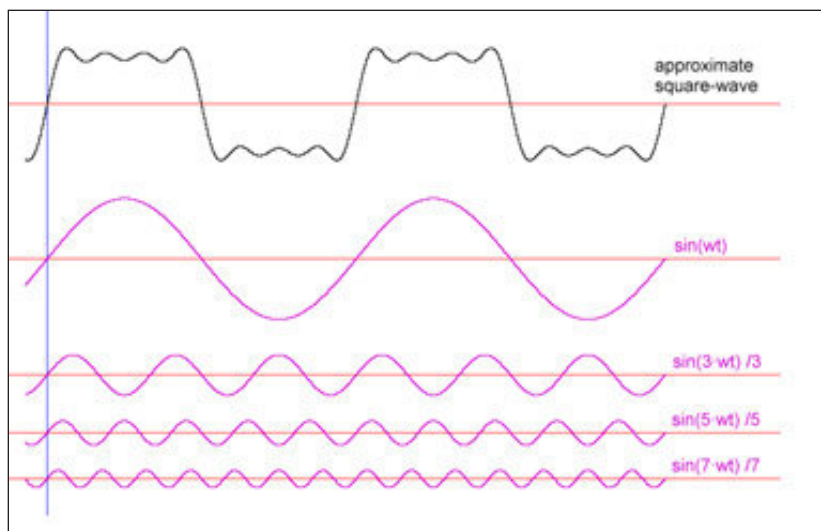
Проще говоря, звук / музыка - это не что иное, как серия сигналов с определенной частотой и определенной амплитудой. Мы можем получить значение в каждой точке, используя `analogRead ()`. Но эта функция будет слишком медленной для выборки звука. Итак, мы будем использовать аналого-цифровой преобразователь микроконтроллера, который автоматически принимает повторяющиеся аналоговые интервалы через определенные интервалы. Мы изучим оба способа отбора проб аудио.

Для анализа дискретизированного звука доступен ряд алгоритмов. Но для повышения производительности мы будем использовать алгоритм БПФ (быстрое преобразование Фурье).

Что такое БПФ (быстрое преобразование Фурье)

Быстрое преобразование Фурье - один из основных и наиболее важных численных алгоритмов в области обработки сигналов. Он преобразует сигнал из его временной области (временная область относится к анализу сигнала относительно времени) в частотную область. Частотная область относится к анализу математической функции или, в данном случае, сигнала относительно частоты.

Вы можете понять временную и частотную области с помощью следующего изображения:



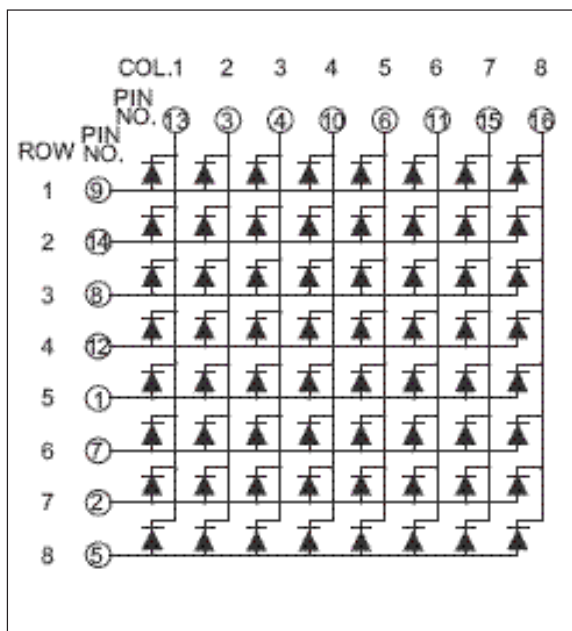
Вы можете разложить любой сигнал на набор синусоид (разной амплитуды, частоты и фазы). Аналогия этому - изображения, которые вы видите на экране своего компьютера, которые состоят из красных, зеленых и синих точек (разной величины и частоты).

БПФ разлагает синусоидальные сигналы из произвольного сигнала путем умножения произвольного сигнала на синусоидальный сигнал определенной частоты. Чем ближе совпадение, тем выше итоговый результат. Когда у нас есть сигнал, который преобразован в частотную область, становится легче его анализировать.

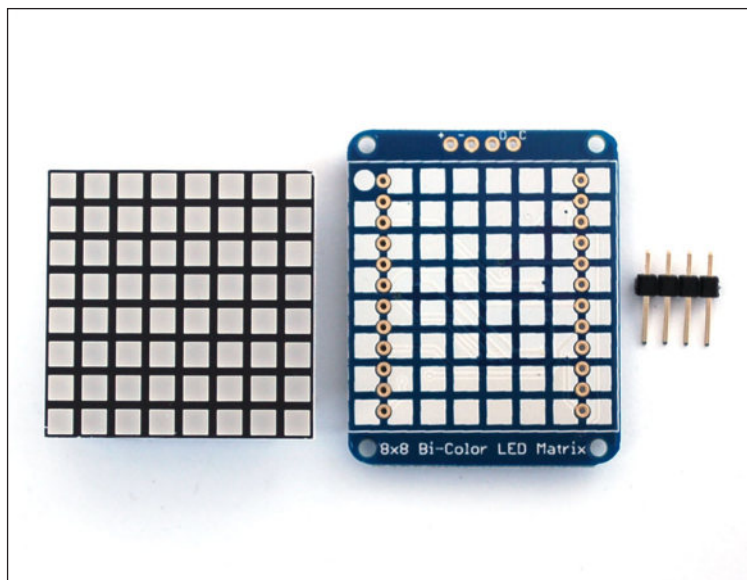
Звуковая визуализация с использованием Arduino

Разобравшись с основами визуализации звука, мы перейдем к реализации визуализации звука с помощью Arduino. Перед тем, как разработать светодиодную елку, разработаем звуковую визуализацию на светодиодной матрице.

Светодиодная матрица представляет собой комбинацию из 64 светодиодов, соединенных вместе, как показано на следующей схеме. Поскольку у Arduino нет 64 контактов, мы не можем подключить отдельные контакты для управления каждым светодиодом. Вместо этого мы будем использовать концепцию мультиплексирования:



Используя мультиплексирование, мы можем управлять любым количеством светодиодов с помощью Arduino. Для управления светодиодной матрицей 8 x 8 нам понадобится одна схема мультиплексора. Мы можем использовать рюкзак от adafruit для мультиплексора или MAX7219 Dot Matrix MCU Control для мультиплексорной части:



Итак, теперь нам требуется всего четыре контакта, а не множество выводов для управления этой светодиодной матрицей.

Для светодиодной матрицы с музыкальным управлением вам потребуются следующие компоненты:

- Аналоговый микрофонный датчик
- Светодиодная матрица 8 x 8

Подключите микрофон и светодиодную матрицу, как показано на схеме. Подключите вывод 3,3 В к AREF Arduino и к выводу Vcc на микрофоне. Это важно для создания опорного напряжения 3,3 В на аналоговый вход с микрофона. Подключите вывод Arduino 5 V к выводу + светодиодной матрицы. Подключите аналоговый вывод A0 Arduino к микрофонному выходу. Подключите контакты Arduino SDA и SCL к контактам D (данные) и C (такты) рюкзачка. Более ранние версии Arduino могут не иметь этих контактов.


```

complex_t fft_buffer[FFT_N];
uint16_t output_spectrum[FFT_N / 2];
volatile byte buffer_position = 0;

byte peakValue[8], dotCount = 0, colCount = 0;
int col[8][10], minAvgLevel[8], maxAvgLevel[8], colDiv[8];

static const uint8_t PROGMEM
// Шум должен быть удален из каждой колонки. Отрегулируйте значения в
// соответствии с требованиями
noiseToDeduct[64] = { 8, 6, 6, 5, 3, 4, 4, 4, 3, 4, 4, 3, 2, 3, 3, 4,
                    2, 1, 2, 1, 3, 2, 3, 2, 1, 2, 3, 1, 2, 3, 4, 4,
                    3, 2, 2, 2, 2, 2, 2, 1, 3, 2, 2, 2, 2, 2, 2, 2,
                    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 4
                    },
// Эквалайзер для удаления шума и нейтрализации шума в области низких
// частот.
equalizer[64] = {
    255, 175, 218, 225, 220, 198, 147, 99, 68, 47, 33, 22, 14, 8, 4,
    2,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
},
// Мы хотим уместить вывод спектра БПФ в 8 столбцов.
// Все бины из выходного спектра бесполезны.
// мы будем следовать ячейкам для вывода столбца.
// Таблица ниже содержит подробную информацию о количестве
// бункеров, которые нужно взять и проиндексировать
from where it will start
// along with the nin numbers.
column0[] = { 2, 1,
             111, 8
             },
column1[] = { 4, 1,
             19, 186, 38, 2
             },
column2[] = { 5, 2,
             11, 156, 118, 16, 1
             },
column3[] = { 8, 3,
             5, 55, 165, 164, 71, 18, 4, 1
             },
column4[] = { 11, 5,

```

Звуковая визуализация и светодиодная новогодняя елка

```
        3, 24, 89, 169, 178, 118, 54, 20, 6, 2, 1
    },
column5[] = { 17, 7,
              2, 9, 29, 70, 125, 172, 185, 162, 118, 74,
              41, 21, 10, 5, 2, 1, 1
            },
column6[] = { 25, 11,
              1, 4, 11, 25, 49, 83, 121, 156, 180, 185,
              174, 149, 118, 87, 60, 40, 25, 16, 10, 6,
              4, 2, 1, 1, 1
            },
column7[] = { 37, 16,
              1, 2, 5, 10, 18, 30, 46, 67, 92, 118,
              143, 164, 179, 185, 184, 174, 158, 139, 118, 97,
              77, 60, 45, 34, 25, 18, 13, 9, 7, 5,
              3, 2, 2, 1, 1, 1, 1
            },
// Он содержит список всех бункеров данных для всех 8 столбцов.
* const binsToUse[] = {
    column0, column1, column2, column3,
    column4, column5, column6, column7
};

Adafruit_BicolorMatrix LEDmatrix = Adafruit_BicolorMatrix();

void setup() {
    uint8_t i, j, nBins, binNum, *outputData;

    memset(peakValue, 0, sizeof(peakValue));
    memset(col, 0, sizeof(col));

    for (i = 0; i < 8; i++) {
        minAvgLevel[i] = 0;
        maxAvgLevel[i] = 512;
        outputData = (uint8_t *)pgm_read_word(&binsToUse[i]);
        nBins = pgm_read_byte(&outputData[0]) + 2;
        binNum = pgm_read_byte(&outputData[1]);
        for (colDiv[i] = 0, j = 2; j < nBins; j++)
            colDiv[i] += pgm_read_byte(&outputData[j]);
    }

    LEDmatrix.begin(0x70);
```

```

Инициализировать автономный режим АЦП; f = (16 МГц / предварительный
делитель) / 13 циклов /
  ADMUX = ADC_INPUT; // Channel sel, right-adj, использовать вывод AREF
  ADCSRA = _BV(ADEN) | // включение АЦП
           _BV(ADSC) | // запуск АЦП
           _BV(ADATE) | // Автоматический запуск
           _BV(ADIE) | // Разрешение прерывания
           _BV(ADPS2) | _BV(ADPS1) | _BV(ADPS0); // 128:1 / 13 = 9615
Hz
  ADCSRB = 0;
  DIDR0 = 1 << ADC_INPUT;
  TIMSK0 = 0;

  sei(); // Разрешить прерывания
}

void loop() {
  uint8_t i, x, L, *outputData, nBins, binNum, weighting, c;
  uint16_t minLvl, maxLvl;
  int level, y, sum;

  while (ADCSRA & _BV(ADIE)); // Дождемся завершения семплирования звука

  fft_input(audio_capture, fft_buffer); // Samples -> complex #s
  buffer_position = 0; // Сброс счетчика отсчетов
  ADCSRA |= _BV(ADIE); // Возобновляем прерывание выборки
  fft_execute(fft_buffer); // Обработка сложных данных
  fft_output(fft_buffer, output_spectrum); // Complex -> spectrum

  Убрать шум и применить эквалайзеры
  for (x = 0; x < FFT_N / 2; x++) {
    L = pgm_read_byte(&noiseToDeduct[x]);
    output_spectrum[x] = (output_spectrum[x] <= L) ? 0 :
      (((output_spectrum[x] - L) * (256L - pgm_read_
byte(&equalizer[x]))) >> 8);
  }

  // Заливаем фон цветами, тогда незанятые части столбцов стираются
  LEDmatrix.fillRect(0, 0, 8, 3, LED_RED); // Верхняя секция
  LEDmatrix.fillRect(0, 3, 8, 2, LED_YELLOW); // Середина
  LEDmatrix.fillRect(0, 5, 8, 3, LED_GREEN); // Нижняя секция

  // Субдискретизация вывода спектра до 8 столбцов:
  for (x = 0; x < 8; x++) {

```

```
outputData = (uint8_t *)pgm_read_word(&binsToUse[x]);
nBins = pgm_read_byte(&outputData[0]) + 2;
binNum = pgm_read_byte(&outputData[1]);
for (sum = 0, i = 2; i < nBins; i++)
    sum += output_spectrum[binNum++] * pgm_read_
byte(&outputData[i]); // Weighted
col[x][colCount] = sum / colDiv[x]; // Взвешенный
minLvl = maxLvl = col[x][0];
for (i = 1; i < 10; i++) { // Получить диапазон предыдущих 10 кадров
    if (col[x][i] < minLvl)
    {
        minLvl = col[x][i];
    }
    else if (col[x][i] > maxLvl)
    {
        maxLvl = col[x][i];
    }
}
// minLvl и maxLvl указывают используемые пределы вывода БПФ
// для динамической установки минимального и максимального уровня столбца.

if ((maxLvl - minLvl) < 8)
{
    maxLvl = minLvl + 8;
}
minAvgLevel[x] = (minAvgLevel[x] * 7 + minLvl) >> 3; // затухание
min/max levels
maxAvgLevel[x] = (maxAvgLevel[x] * 7 + maxLvl) >> 3; // (ложное
скользящее среднее)

// Вторая шкала с фиксированной точкой, основанная на динамических минимальных /
максимальных уровнях:
level = 10L * (col[x][colCount] - minAvgLevel[x]) /
(long)(maxAvgLevel[x] - minAvgLevel[x]);
// Обрезать вывод и преобразовать в байт:
if (level < 0L)
{
    c = 0;
}
else if (level > 10)
{
    c = 10; // Разрешить точке отойти на пару пикселей сверху
}
else
```

```
{
    c = (uint8_t)level;
}

if (c > peakValue[x])
{
    peakValue[x] = c; // Оставляем точку сверху
}

if (peakValue[x] <= 0) // Нет вывода
{
    LEDmatrix.drawLine(x, 0, x, 7, LED_OFF);
    continue;
}
else if (c < 8) // Частичный столбец?
{
    LEDmatrix.drawLine(x, 0, x, 7 - c, LED_OFF);
}

// Цвет точки пика меняется, но не обязательно совпадает
// три области экрана ... желтый имеет небольшое дополнительное влияние.
y = 8 - peakValue[x];
if (y < 2)
{
    LEDmatrix.drawPixel(x, y, LED_RED);
}
else if (y < 6)
{
    LEDmatrix.drawPixel(x, y, LED_YELLOW);
}
else
{
    LEDmatrix.drawPixel(x, y, LED_GREEN);
}
}

LEDmatrix.writeDisplay();

// В каждом третьем кадре уменьшаем пиковые пиксели на 1:
if (++dotCount >= 3)
{
    dotCount = 0;
    for (x = 0; x < 8; x++)
    {
```

```
        if (peakValue[x] > 0) peakValue[x]--;
    }
}

if (++colCount >= 10)
{
    colCount = 0;
}
}

ISR(ADC_vect) { // Прерывание дискретизации звука
    static const int16_t noiseThreshold = 4;
    int16_t sample = ADC; // значения от 0 до 1023

    audio_capture[buffer_position] =
        ((sample > (512 - noiseThreshold)) &&
         (sample < (512 + noiseThreshold))) ? 0 :
        sample - 512; // Sign-convert for FFT; -512 to +511

    if (++buffer_position >= FFT_N) ADCSRA &= ~_BV(ADIE);
    // Преобразование знака для БПФ; От -512 до +511
}
```

Давайте разберемся с кодом подробнее. Здесь, как мы обсуждали ранее, мы используем встроенный АЦП для дискретизации звука. При работе с дискретизацией / обработкой сигналов одна из самых полезных функций Arduino, которая может пригодиться, - это прерывания. Прежде чем мы поймем, как работают прерывания, нам нужно знать о прерываниях.

Прерывания позволяют выполнять определенные важные задачи в фоновом режиме и прерывают поток выполнения при возникновении определенного события. В нашем случае прерывания используются для дискретизации звука. После того, как звук обработан с помощью алгоритма БПФ и отображается на дисплее, он снова производит выборку звука.

Хотя мы не слышим все звуки, вокруг нас присутствует некоторый фоновый шум. Мы должны удалить окружающий шум перед обработкой музыки для рендеринга на светодиодной матрице. Итак, мы сохранили 64 значения в массиве `noiseToDeduct`, которые впоследствии будут вычтены из сигнала. Если вы получаете больше шума на выходе, вы можете отрегулировать шум, установив разные значения в массиве.

Помимо шума, нам также необходимо выровнять звук в сторону низких частот. Итак, мы храним разные значения в эквалайзере массива. После того, как шум будет вычтен из выходного сигнала фильтра, мы нормализуем спектр звука / выходного сигнала.

Кроме того, выходной спектр БПФ будет иметь гораздо больше блоков / выборок, чем мы сможем использовать. Самый нижний и несколько верхних участков спектра либо зашумлены, либо находятся вне диапазона. Итак, мы будем использовать выходные значения спектра БПФ, хранящиеся в массивах `column0`, `column1`, `column2`, `column3`, `column4`, `column5`, `column6` и `column7`. Массив содержит количество используемых выходных значений и начальный индекс ячеек вместе с их весами. Хотя эти значения могут быть изменены в соответствии с нашими требованиями, эти значения получены после тщательного тестирования `adafruit`.

Здесь мы использовали две новые функции: `pgm_read_word` и `pgm_read_byte`. Обе функции предоставляются библиотекой `avr / pgmspace.h`. `pgm_read_word` используется для чтения слова из области программы с 16-битным (близким) адресом. Точно так же `pgm_read_byte` используется для чтения байта из области программы с 16-битным (близким) адресом:

```
outputData = (uint8_t *)pgm_read_word(&binsToUse[x]);
nBins = pgm_read_byte(&outputData[0]) + 2;
binNum = pgm_read_byte(&outputData[1]);
```

Для получения данных из массива `binsToUse` мы использовали `pgm_read_word`. При чтении определенного значения из `outputData` мы использовали `pgm_read_byte`. Здесь мы используем 128-битный буфер для хранения дискретизированного звука. Когда буфер заполняется, он обрабатывает данные, передавая их фильтру БПФ. После выходного спектра БПФ сигнал снова подвергается понижающей дискретизации:

```
sum += output_spectrum[binNum++] * pgm_read_
byte(&outputData[i]); // Взвешенный
col[x][colCount] = sum / colDiv[x]; // В среднем
```

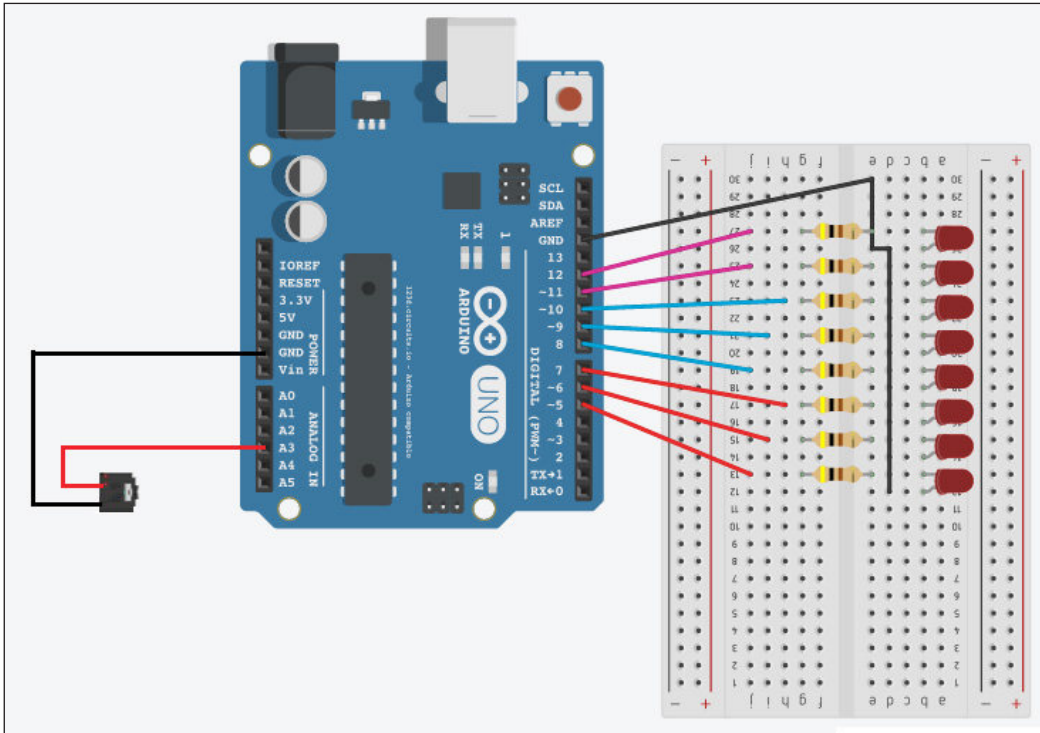
Мы находим средневзвешенное значение для каждого столбца, выбирая значение из предопределенной таблицы, инициализированной в начале. Найдя средневзвешенное значение для восьми столбцов, мы записываем эти значения на светодиодный матричный дисплей после сохранения минимального и максимального значений выходного спектра. Эти динамические минимальные и максимальные значения полезны для того, чтобы сделать дисплей интересным, даже на малой громкости.

После понимания звуковой визуализации с использованием БПФ, мы теперь разработаем светодиодную рождественскую елку, которая синхронизирует свое освещение с ритмами.

Разработка светодиодной елки

Теперь мы знакомы с концепцией звуковой визуализации. Мы также узнали об управлении светодиодной матрицей с помощью музыки. Теперь мы разработаем светодиодную рождественскую елку, которая будет мигать светодиодами в соответствии с ритмами музыки.

Чтобы разработать базовую схему, которая реагирует на биения, подключите схему, как показано на следующем рисунке:



Мы подключим аудиовход / микрофон к аналоговому контакту 3 Arduino. Мы подключили светодиоды к контактам с 5 по 12.

После того, как вы подключили схему, как указано, загрузите следующий код в Arduino:

```
#include <fix_fft.h>

int LEDPins[] = {5, 6, 7, 8, 9, 10, 11, 12};
int x = 0;
char imaginary[128], inputSignal[128];
char outputAverage[14];
int i = 0, inputValue;
#define AUDIOPIN 1

void setup()
{
  for (int i = 0; i < 8; i++)
```

```
{
  pinMode(LEDpins[i], OUTPUT);
}
Serial.begin(9600);
}

void loop()
{
  for (i = 0; i < 128; i++) {
    inputValue = analogRead(AUDIOPIN);
    inputSignal[i] = inputValue;
    imaginary[i] = 0;
  };
  fix_fft(inputSignal, imaginary, 7, 0);
  for (i = 0; i < 64; i++) {
    inputSignal[i] = sqrt(inputSignal[i] * inputSignal[i] +
imaginary[i] * imaginary[i]); // это получает абсолютное
значение значений в массиве, поэтому мы имеем дело только
с положительными числами
  };
  // средние столбцы вместе
  for (i = 0; i < 14; i++) {
    outputAverage[i] = inputSignal[i * 4] + inputSignal[i * 4 + 1] +
inputSignal[i * 4 + 2] + inputSignal[i * 4 + 3]; // в среднем вместе
    outputAverage[i] = map(outputAverage[i], 0, 30, 0, 9);
  }
  int value = outputAverage[0]; // 0 для баса
  writetoLED(value);
}

void writetoLED(int mappedSignal)
{
  if (mappedSignal > 8)
  {
    for (int i = 0; i < 8; i++)
    {
      digitalWrite(LEDpins[i], HIGH);
    }
  }
  else if (mappedSignal > 7)
  {
    for (int i = 0; i < 7; i++)
    {
```

```
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 7; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 6)
{
    for (int i = 0; i < 6; i++)
    {
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 6; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 5)
{
    for (int i = 0; i < 5; i++)
    {
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 5; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 4)
{
    for (int i = 0; i < 4; i++)
    {
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 4; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 3)
{
    for (int i = 0; i < 3; i++)
    {
```

```
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 3; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 2)
{
    for (int i = 0; i < 2; i++)
    {
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 2; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else if (mappedSignal > 1)
{
    for (int i = 0; i < 1; i++)
    {
        digitalWrite(LEDpins[i], HIGH);
    }
    for (int i = 1; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
else
{
    for (int i = 0; i < 8; i++)
    {
        digitalWrite(LEDpins[i], LOW);
    }
}
}
```

Здесь мы напрямую считываем сигнал с микрофона с помощью `analogRead ()`. Мы снимаем 128 отсчетов. После снятия всех показаний мы обрабатываем эти показания с помощью библиотеки `fix_fft.h`, доступной по адресу <http://forum.arduino.cc/index.php/topic,38153.0.html>. После сохранения на вашем компьютере и `fix_fft.h`, и `fix_fft.cpp`, вам необходимо импортировать эту библиотеку в свой код Arduino.

После получения выходного спектра от БПФ мы берем среднее значение выходного сигнала для включения или выключения светодиодов на основе среднего значения. Поскольку басовая партия чаще всего обеспечивает гармоники и ритмическую поддержку, мы используем значение низких частот для управления светодиодами. Мы получаем значение низких частот как `outputAverage [0]`. Отображение среднего значения от 0 до 9 помогает легко включить или выключить светодиоды.

В художественных целях мы соединим несколько светодиодов вокруг модели елки. Таким образом, елка будет синхронизировать свое освещение с ритмами музыки.

Резюме

В этой главе мы начали с основ звуковой визуализации. Поняв алгоритм БПФ, мы переходим к визуализации звука с помощью *Arduino*, используя его на светодиодном дисплее с точечной матрицей 8 x 8. В конце этой главы мы разработали светодиодную рождественскую елку, которая синхронизирует свой свет в соответствии с ритмами музыки. В следующей главе мы перейдем к развитию «настойчивости видения».

6

Постоянство зрения

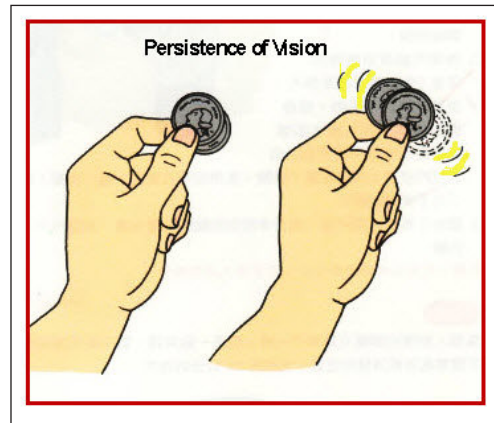
До сих пор в этой книге мы создавали все объекты, которые по своей природе неподвижны, то есть не могут двигаться. В заключительном проекте этой книги мы создадим жезл Persistence of Vision, используя светодиодную матрицу и двигатель. Но, прежде всего, вы познакомитесь со светодиодными матрицами и двигателями. В этой главе мы рассмотрим следующие темы:

- Постоянство зрения
- Программирование светодиодной матрицы
- Управление двигателем с помощью Arduino
- Синхронизация времени светодиодной матрицы в зависимости от скорости двигателя.

Создайте свое собственное постоянство видения

Один из пяти органов чувств нашего тела, глаз - замечательный инструмент, который помогает нам обрабатывать свет таким образом, чтобы наш разум мог создавать из него образ. Постоянство зрения относится к оптической иллюзии, когда несколько разных изображений сливаются в один образ в человеческом сознании. Иллюзия постоянства зрения играет роль в том, чтобы мир не становился черным как смоль каждый раз, когда мы моргаем. Всякий раз, когда на сетчатку попадает свет, она сохраняет отпечаток света в течение примерно десятой секунды после того, как источник света удален. Из-за этого глаз не может различить изменения, которые происходят быстрее, чем этот период хранения. Это похожее явление используется в фильмах. Кинофильм создает иллюзию за счет быстрого упорядочивания отдельных фотографий. Обычно для движущихся изображений частота кадров составляет 24 кадра в секунду, что приводит к отсутствию мерцания изображения. Если удерживать количество кадров в секунду ниже 16, разум может различать изображения, что приводит к миганию изображений или мерцанию.

Посмотрите на следующее изображение; когда вы двигаете руками вперед и назад, как показано, вы увидите мерцание во всех положениях:



Известный профессор Университета Центрального Арканзаса процитировал это: «Понятие « постоянство видения », по-видимому, было заимствовано из психологии первого десятилетия века, периода, в течение которого возникло кино. Но в то время как большинство кинологов принимали « постоянство видения » как перцептивную основу медиума и приступили к теоретическим рассуждениям о природе, значении и функционировании кино на этой основе, перцептивные психологи продолжали подвергать сомнению механизмы, участвующие в восприятии движения; и они достигли понимания, которое требует переосмысления многих выводов, сделанных кинологами во время последние 50 лет ".

После знакомства с концепцией **Persistence of Vision** давайте погрузимся в то, как мы можем создать собственный **PoV**. Для этого нам понадобятся следующие компоненты:

- Arduino
- Светодиодная матрица
- Двигатель постоянного тока
- Резистор
- Драйвер двигателя L293D
- Деревянный материал для изготовления основы PoV.

В следующем разделе вы узнаете, как использовать эти компоненты для создания собственного Persistence of Vision (постоянства видения).

Программирование светодиодной матрицы

Светодиодная матрица - это не что иное, как несколько светодиодов, соединенных вместе. Размер светодиодной матрицы обычно состоит из восьми или 16 светодиодов. Управлять массивом светодиодов можно напрямую с помощью функции `digitalWrite ()`. Помимо использования функции `digitalWrite ()`, вы можете управлять светодиодами напрямую, используя связь на уровне порта. На Arduino у нас есть три порта: порты В, С и D:

- Порт В: цифровые выводы с 8 по 13
- Порт С: аналоговые выводы
- Порт D: цифровые выводы с 0 по 7

Каждый порт управляется тремя регистрами DDR. Эти регистры являются определенными переменными в Arduino как DDRB, DDRC и DDRD. Используя эти переменные, мы можем сделать выводы либо входными, либо выходными в функции настройки.

Вы можете использовать следующий синтаксис для инициализации выводов:

```
DDRy = Vxxxxxxxx
```

Здесь *y* - это имя порта (В / С / D), а *x* - значение вывода, которое определяет, является ли вывод входным или выходным. Мы будем использовать 0 для ввода и 1 для вывода. LSB (младший значащий байт) - это наименьший номер вывода для этого регистра.

Чтобы управлять выводами с помощью этой манипуляции с портом, мы можем использовать следующий синтаксис:

```
PORTy = Vxxxxxxxx
```

Здесь *y* - это имя порта (В / С / D), и сделайте *x* равным 1, чтобы сделать вывод HIGH (ВЫСОКИМ) и 0, чтобы сделать вывод LOW (НИЗКИМ).

Вы можете использовать следующий код для управления светодиодами с помощью связи на уровне порта:

```
void setup()
{
    DDRD = B11111111; // устанавливаем PORTD (цифровой 7-0) на выходы
}

void loop()
{
    PORTD = B11110000; // цифровой 4 ~ 7 ВЫСОКИЙ, цифровой 3-0 НИЗКИЙ
    delay(2000);
}
PORTD = B00001111; // цифровой 4 ~ 7 НИЗКИЙ, цифровой 3-0 ВЫСОКИЙ
delay(2000);
```

Есть несколько плюсов и минусов использования техники манипулирования портами. Ниже приведены недостатки использования техники манипулирования портами:

- Код становится сложнее отлаживать и поддерживать, и требуется много времени, чтобы понять код.
- Код становится менее переносимым. Если вы используете `digitalWrite ()` и `digitalRead ()`, гораздо проще написать код, который будет работать на всех микроконтроллерах, тогда как порты и регистры могут быть разными для каждого типа микроконтроллера.
- Вы можете вызвать непреднамеренные сбои при прямом доступе к порту, поскольку контакт 0 является линией приема для последовательного порта. Если вы случайно введете его, это может привести к тому, что ваш последовательный порт перестанет работать.

Техника манипулирования портами имеет несколько преимуществ по сравнению с обычными методами работы с кодом:

- В случае ограниченного по времени использования Arduino, вам нужно будет очень быстро включать и выключать выводы. Используя прямой доступ к порту, вы можете сэкономить много машинных циклов.
- Кроме того, если у вас заканчивается программная память, вы можете использовать эти методы, чтобы уменьшить размер кода.

Различные типы двигателей

В зависимости от потребностей вашего проекта вы можете выбирать из множества двигателей, доступных на рынке. Для хобби-электроники вы в основном будете использовать двигатель постоянного тока, серводвигатель или шаговый двигатель. На следующем изображении показаны все три типа двигателей. Слева направо, двигатель постоянного тока, серводвигатель и шаговый двигатель:



Так как в проекте мы будем использовать двигатель DC, то остальные типы двигателей мы рассматривать не будем.

Двигатели постоянного тока (DC)

Двигатели DC представляют собой двигатели непрерывного вращения. Когда на двигатель подается питание, он начинает работать и останавливается после отключения питания. Скорость двигателя постоянного тока регулируется с помощью ШИМ (как описано в главе 2, проект 1 - светодиодный ночник). Рабочий цикл определяет скорость двигателя.

Управление двигателем DC с помощью Arduino

В этой главе мы узнаем, как управлять двигателем DC с помощью Arduino

Вы можете управлять двигателем DC, используя тот же код, что и для мигающего светодиода. Двигатель DC - это двухполюсный компонент. Один вывод - это плюс, другой - минус. Если мы подключим +двигателя к положительной клемме аккумулятора, а минус двигателя - к отрицательной, двигатель будет работать в одном направлении, а при изменении полярности, вал двигателя будет вращаться в обратном направлении.

Подключив двигатель к двум цифровым выводам Arduino, мы можем управлять направлением двигателя. В следующем базовом коде мы будем запускать двигатель в одном направлении в течение пяти секунд, а затем изменим направление вращения вала двигателя на противоположное. Соедините выводы 3 и 4 Arduino с двумя клеммами двигателя:

```
int motorPos = 3;
int motorNeg = 4;

void setup() {
  pinMode(motorPos, OUTPUT);
  pinMode(motorNeg, OUTPUT);
}

void loop() {
  //запустить двигатель в одном направлении
  digitalWrite(motorPos, HIGH);digitalWrite
  (motorNeg, LOW);
  delay(5000); //ждем 5 секунд
  // Изменить направление двигателя
```

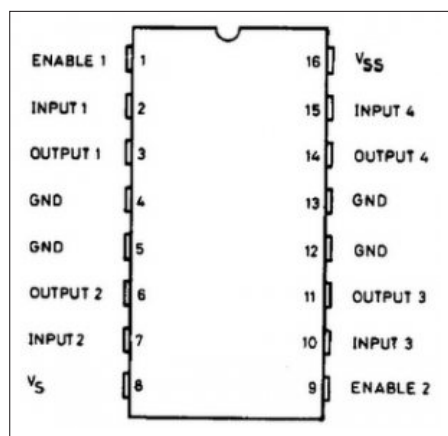
```

digitalWrite(motorPos, LOW);
digitalWrite(motorNeg, HIGH);
delay(5000);
}

```

В предыдущем коде мы подаем противоположные выходы на оба контакта 3 и 4, что нужно для определения направления двигателя.

Хотя этот метод управления двигателями постоянного тока напрямую с помощью Arduino кажется очень простым, у него есть свои недостатки. С пина ввода-вывода Arduino можно получить максимальный ток 20 мА. Итак, если мы подключим нагрузку к Arduino, Arduino может выйти из строя. Для этого мы используем микросхему L293D, совместимую с H-мостовой схемой. H-мост - это схема, которая может приводить двигатель в движение в обоих направлениях. Прежде чем подключать L293D к Arduino, проверьте все подключения к выводам по следующему рисунку:



Соедините *Enable1* и *Enable2* с +5 В. L293D спроектирована таким образом, что левые выводы IC могут использоваться для управления одним двигателем, а правые выводы IC могут использоваться для управления другим двигателем в обоих направлениях. Для управления одним двигателем подайте вход на контакты 2 и 7, которые будут давать выход на контакты 3 и 7. Контакт 8 является источником питания для двигателей. Подключите +5 В к выводу 16.

В большинстве случаев требуется стабилизатор напряжения, поскольку контроллер не может выдерживать напряжение более 5 В. Но в Arduino UNO есть встроенный стабилизатор напряжения. Хотя на Arduino UNO можно подать до 20 В, рекомендуется подавать входное напряжение до 12 В.

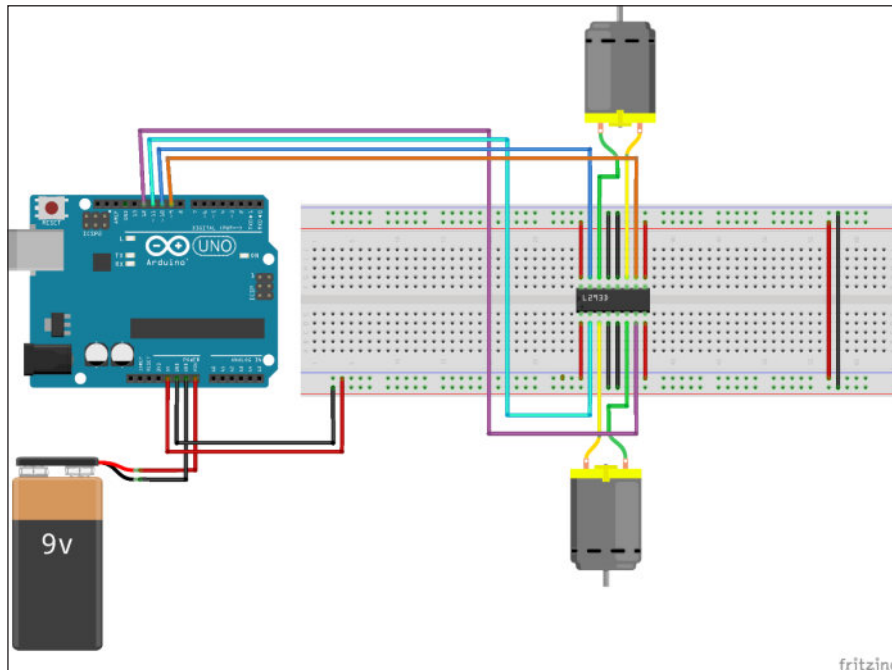
Как вы можете видеть на предыдущем изображении, двумя двигателями можно управлять с помощью одной микросхемы с нормальным напряжением (9 В).

Мы подключим Arduino UNO в качестве контроллера и подключим входы к контактам 3 и 4. Мы подключим один двигатель к контактам 3 и 4, а другой двигатель - к контактам 7 и 8. Как показано на следующем изображении, мы можем сделать простой робот, подключив ролик в качестве третьего колеса после установки этих двигателей на шасси:



Мы заставим этого робота выполнять простую повторяющуюся задачу, используя L293D и Arduino. Подключите управляющий сигнал / выход Arduino к двигателям к контактам 2, 7, 9 и 15 L293D.

Подключите двигатель между контактами 3 и 6 L293D. После того, как вы выполнили такое же соединение для другой стороны L293D, то есть контактов с 9 по 15, ваша схема будет выглядеть, как на следующем изображении:



Еще раз проверив все соединения, загрузите следующий код в Arduino:

```
int leftMotorPos = 10;
int leftMotorNeg = 9;
int rightMotorPos = 12;
int rightMotorNeg = 13;

void setup()
{
  pinMode(leftMotorPos, OUTPUT);
  pinMode(rightMotorPos, OUTPUT);
  pinMode(leftMotorNeg, OUTPUT);
  pinMode(rightMotorNeg, OUTPUT);
}

void loop()
{
  forward();
  delay(5000);
  right();
  delay(5000);
  left();
  delay(5000);
  reverse();
  delay(5000);
  stopAll();
  delay(5000);
}

void forward() {
  digitalWrite(rightMotorPos, HIGH);
  digitalWrite(leftMotorPos, HIGH);
  digitalWrite(rightMotorNeg, LOW);
  digitalWrite(leftMotorNeg, LOW);
}

void left() {
  digitalWrite(rightMotorPos, HIGH);
  digitalWrite(leftMotorPos, LOW);
  digitalWrite(rightMotorNeg, LOW);
  digitalWrite(leftMotorNeg, LOW);
}

void right() {
```

```
digitalWrite(rightMotorPos, LOW);  
digitalWrite(leftMotorPos, HIGH);  
digitalWrite(rightMotorNeg, LOW);  
digitalWrite(leftMotorNeg, LOW);  
}  
  
void reverse() {  
digitalWrite(rightMotorPos, LOW);  
digitalWrite(leftMotorPos, LOW);  
digitalWrite(rightMotorNeg, HIGH);  
digitalWrite(leftMotorNeg, HIGH);  
}  
  
void stopAll() {  
digitalWrite(rightMotorNeg, LOW);  
digitalWrite(leftMotorNeg, LOW);  
digitalWrite(rightMotorPos, LOW);  
digitalWrite(leftMotorPos, LOW);  
}
```

В соответствии с предыдущей схемой и кодом мы не подаем данные напрямую на двигатели; скорее, мы вводим входные данные для L293D, который, в свою очередь, обеспечивает достаточную мощность для работы двигателей.

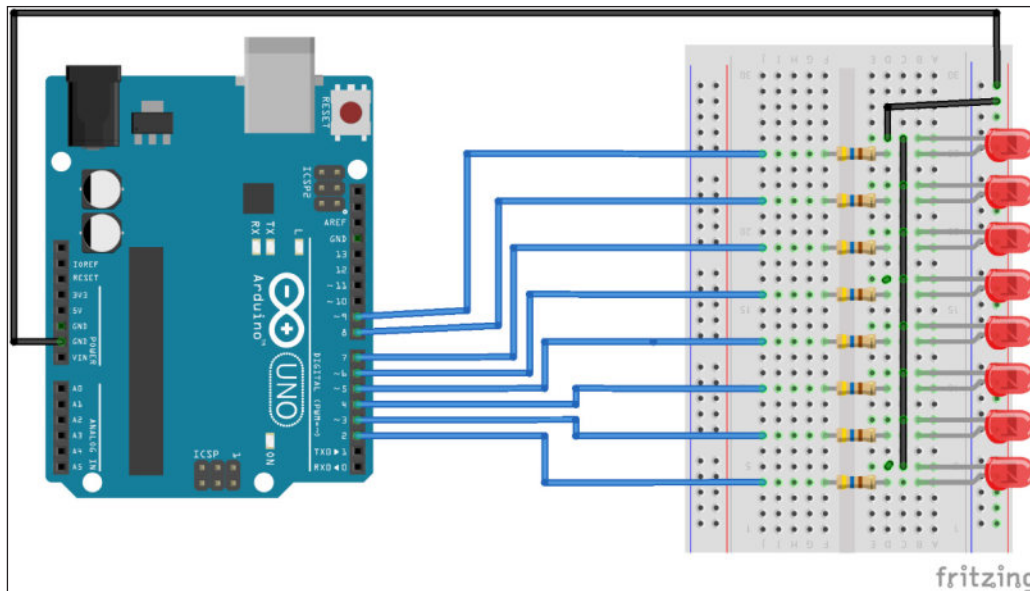


Обязательно подключите заземление между Arduino и L293D. В противном случае заземление будет в плавающем режиме, и двигатель будет работать резко, то есть двигатель может иногда работать, а иногда не запускаться.

Another way to control the motor using L293 is to give a signal to an enable pin. By controlling enable pins' input, you can control the motors. Here, we are giving control to the motor input and the enable pins are given high input continuously.

Синхронизация светодиодной матрицы с двигателем

В предыдущих разделах этой главы мы узнали об управлении светодиодной матрицей и двигателем постоянного тока с помощью Arduino:



После того, как вы подключили схему, как показано на предыдущем изображении, загрузите следующий код в Arduino. В следующем коде мы пишем «Привет, мир».

```
int LEDPins[] = {2, 3, 4, 5, 6, 7, 8, 9};
int noOfLEDs = 8;

//данные, соответствующие каждому алфавиту и несколько символов для отображения

byte H[] = {B11111111, B11111111, B00011000, B00011000, B00011000,
B00011000, B11111111, B11111111};
byte E[] = {B00000000, B11111111, B11011011, B11011011, B11011011,
B11011011, B11000011, B11000011};
byte L[] = {B00000000, B11111111, B11111111, B00000011, B00000011,
B00000011, B00000011, B00000011};
byte O[] = {B00000000, B11111111, B11111111, B11000011, B11000011,
B11000011, B11111111, B11111111};
byte fullstop[] = {B00000000, B00000000, B00000000, B00000011,
B00000011, B00000000, B00000000, B00000000};
byte comma[] = {B00000000, B00000000, B00000000, B00000110, B00000101,
B00000000, B00000000, B00000000};

// Настроить параметры в зависимости от необходимости
int timeBetweenColumn = 2.2;
int timeBtwnFrame = 20;
```

```
int frame_len = 8;

void setup()
{
  int i;
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(10, INPUT);
  for (i = 0; i < noOfLEDs; i++) {
    pinMode(LEDpins[i], OUTPUT);    // установить каждый вывод как
                                     ВЫХОД
  }
}

void loop()
{
  int b = 0;
  digitalWrite(12, HIGH);
  digitalWrite(13, HIGH);
  digitalWrite(11, HIGH);
  delay(timeBtwnFrame);
  show(H);
  delay(timeBtwnFrame);
  show(E);
  delay(timeBtwnFrame);
  show(L);
  delay(timeBtwnFrame);
  show(L);
  delay(timeBtwnFrame);
  show(O);
  delay(timeBtwnFrame);
}

void show( byte* image )
{
  int a, b, c;

  // просмотреть все данные для всех столбцов в каждом кадре.
  for (b = 0; b < frame_len; b++)
  {
    for (c = 0; c < noOfLEDs; c++)
    {
      digitalWrite(LEDpins[c], bitRead(image[b], c));
    }
  }
}
```

```
    delay(timeBetweenColumn);  
  }  
  for (c = 0; c < noOfLEDS; c++)  
  {  
    digitalWrite(LEDpins[c], LOW);  
  }  
}
```

Изначально настраиваем пины, которые нужно включить при написании любых букв. Здесь мы написали код для букв H, E, L, O, точки и запятой.

Здесь мы используем восемь светодиодов для отображения PoV. Обязательно подключите резистор между светодиодом и Arduino. Поскольку мы должны заставить светодиоды светиться в зависимости от скорости двигателя, мы можем контролировать это, изменяя значение переменных `timeBetweenColumn` и `timeBtwnFrame` в коде.

Изменяя значения этих двух переменных, вы сможете синхронизировать светодиоды со скоростью двигателя. Еще одна вещь, которую вы можете сделать, - это инициализировать переменную фиксированным значением и, используя последовательную связь, изменить значение этих переменных. Это легко сделать с помощью библиотеки `SoftwareSerial`.

Воплощая свои усилия в жизнь

Как только вы узнаете, как управлять двигателями и светодиодами с помощью Arduino, последний шаг - поместить все, что вы уже изучили, и сделать его автономным продуктом. Есть несколько способов добиться этого:

- Самый простой способ - использовать руки.
- Использование двух разных Arduino или внешних двигателей
- Использование существующих реальных устройств

Используйте руки для вращения

Несмотря на то, что вы узнали об управлении двигателями и светодиодами, если вы делаете это впервые, вам потребуется некоторое время, чтобы понять синхронизацию светодиодов и двигателей. Самый простой способ проверить Постоянство зрения - вращать руками. Перед тем, как начать вращать полную установку, убедитесь, что вы загрузили последний скетч в Arduino, подключили автономный аккумулятор / источник питания и надежно закрепили светодиоды на каком-либо основании. В качестве основы можно использовать лист термобаллона или картон и закрепить их изолентой.

После того, как вы исправили все материалы, вы можете повернуть PoV и увидеть, как ваши усилия оживают, как показано на следующем изображении:



Использование двух разных Arduino или внешних двигателей

После того, как вы починили Arduino, светодиоды и внешний аккумулятор / источник питания, вы можете использовать еще один Arduino и подключить двигатели для вращения всей конструкции. Сначала кажется, что вы можете управлять двигателями и светодиодами с помощью одного и того же Arduino, однако, если вы немного подумаете, вы поймете, почему вам не следует этого делать. Причина в вращении светодиодной структуры и соединительного провода между двигателем и Arduino. Если у вас есть внешний двигатель, вы можете использовать его и подключить к нему структуру Arduino-LED.

Используйте существующие реальные устройства

Еще одна интересная идея - использовать существующие реальные устройства, которые есть у вас дома. Конечно, есть много устройств, которыми вы можете пользоваться; мы протестировали его с двумя устройствами: велосипедным колесом и настольным вентилятором, как показано на следующем изображении. Нет никакой разницы в настройке по сравнению с двумя другими методами. Фактически, вы можете использовать ту же настройку / структуру, что и в методе 1:



Резюме

В этом последнем проекте этой книги вы узнали о двигателях и о том, как управлять светодиодной матрицей. Поняв «Постоянство зрения», мы разработали дисплей «Постоянство зрения». После разработки всех проектов, описанных в этой книге, вы сможете поиграть с различными типами светодиодов и сделать из них что-нибудь новаторское. В последней главе мы узнаем о некоторых проблемах, с которыми вы можете столкнуться при разработке проектов, описанных в этой книге.

7

Устранение неполадок и дополнительные ресурсы

В этой книге вы познакомились с Arduino Pi и его возможностями, вы разработали свою светодиодную ночную лампу, дистанционно управляемую подсветку телевизора, светодиодный куб, визуализацию звука и, наконец, постоянное зрение. Могли быть случаи, когда вы хотели узнать больше об определенных темах или застряли между ними. Эта глава отвечает на все эти вопросы.

В первом разделе упоминаются общие методы устранения неполадок. Во второй и последней части главы есть ресурсы, которые будут полезны, если вы хотите делать сложные вещи с помощью Arduino:

- Поиск проблемы
- Ресурсы - опытные пользователи

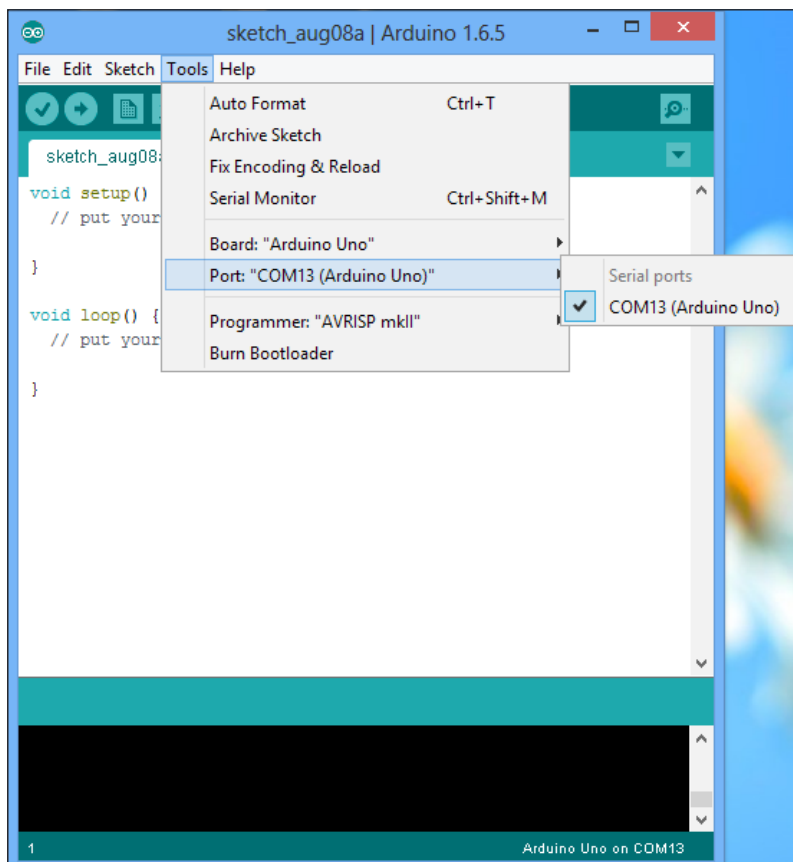
Поиск проблемы


В этом разделе есть ответы на некоторые из распространенных проблем, с которыми вы можете столкнуться при работе с Arduino.

Не могу загрузить программу

В программе Arduino Environment перейдите в Инструменты | Последовательный порт и выберите соответствующий последовательный порт. Чтобы узнать, какой последовательный порт использует плата, подключите плату к компьютеру с помощью кабеля USB. На рабочем столе Windows щелкните правой кнопкой мыши «Мой компьютер», затем «Свойства» | Диспетчер устройств | Порты (COM и LPT). Будет запись типа USB Serial Port (COM13) или ArduinoUNO (COM13).

Это означает, что используется порт 13 последовательной связи:



[ При загрузке вы можете получить такие сообщения об ошибках, как «Последовательный порт COM13» уже используется ". Попробуйте закрыть все программы, которые могут его использовать.]

Одна из возможных причин может заключаться в том, что вы используете несколько IDE Arduino на одной машине. Попробуйте закрыть все IDE Arduino и открыть скетч, который вы хотите загрузить на свою плату Arduino. В большинстве случаев это решит проблему. Даже после повторного открытия Arduino IDE, если вы получите такое же сообщение об ошибке, отключите Arduino и снова подключите его. Это должно решить вашу проблему. В некоторых случаях, если вы получаете такое же сообщение об ошибке снова, перезагрузите компьютер.

Светодиод тускло светит

Это самая частая ошибка новичков. Иногда светодиод, подключенный к выводу Arduino, тусклый. Это связано с тем, что вывод Arduino, подключенный к светодиоду, не объявлен как OUTPUT и не получает полную мощность от платы Arduino. Если вы объявите вывод светодиода Arduino как OUTPUT, это решит проблему, и светодиод будет правильно светиться. Вы также можете обратиться к разделу устранения неполадок на веб-сайте Arduino. Вы можете найти решения других проблем на <https://www.arduino.cc/en/Guide/>

Ресурсы - опытные пользователи

Этот раздел содержит несколько продвинутых проектов, которыми, на мой взгляд стоит заняться. В этом последнем разделе есть несколько интересных и полезных ресурсов, которые помогут вам перейти на новый уровень Arduino.

Проекты

Основываясь на всех имеющихся у вас базовых навыках программирования Arduino, светодиодов и датчиков, мы полагаем, что следующие четыре проекта могут быть вам интересны.

Twitter Mood Light

Это один из классных проектов, которые я видел. Это способ взглянуть на коллективное человеческое сознание. Это способ быть в курсе мировых событий по мере их развития или когда происходит что-то важное. Arduino подключается напрямую к любой беспроводной сети через модуль WiFi. Затем он ищет в Твиттере твиты с эмоциональным содержанием и сопоставляет твиты по каждой эмоции. Он также выполняет некоторую математику, например, тускнеет цвет светодиода, отражая текущее настроение в мире. Вот несколько примеров:

- Красный цвет означает гнев
- Желтый для счастья
- Розовый для любви
- Белый от страха
- Зеленый цвет от зависти
- Оранжевый для сюрприза
- Синий означает печаль

В этом проекте после получения твитов из дескриптора твиттера для пользователя с помощью метода извлечения настроек вы можете узнать об эмоциях / настроении мира.

Подробнее читайте на:

<http://www.instructables.com/id/Twitter-Mood-Light-The-Worlds-Mood-in-a-Box/>.

Дверной замок с секретным поиском

Теперь вы можете скрыть свое секретное убежище от злоумышленников с помощью замка, который откроется только тогда, когда он услышит секретный стук. Поначалу это не признавалось, но оказалось на удивление точным при оценке ударов. Если достигнута большая точность, можно даже различить людей, даже если они будут стучать одинаково!

Подробнее читайте на

<http://www.instructables.com/id/Secret-Knock-Detecting-Door-Lock/>.

LED велосипедная куртка

Это один из лучших проектов, демонстрирующих свои творческие способности другу и внешнему миру. В этом проекте показано, как создать куртку с сигналами поворота, которая позволит людям знать, куда вы едете, когда вы едете на велосипеде. В нем используются токопроводящие нити и сшиваемая электроника. В этом проекте вы научитесь использовать другой тип Arduino, который специально разработан для ношения - LilyPad.

Подробнее читайте на

<http://www.instructables.com/id/turn-signal-biking-jacket/>.

Кофейник с поддержкой Twitter

Tweet-a-pot - это следующая волна модных устройств с поддержкой твиттера. Этот кофейник позволяет вам приготовить кофе из любого места, где есть приемная сотового телефона, используя Twitter и плату Arduino. Tweet-a-pot - это простая реализация для удаленного управления устройством; используя немного кода и устройство, вы можете иметь свой собственный кофейник с поддержкой Twitter.

Подробнее на:

<http://www.instructables.com/id/Tweet-a-Pot-Twitter-Включено-Coffee-Pot/>.

Полезные ресурсы

Из этой книги вы узнали об Arduino, светодиодах и датчиках. Если вы посмотрите на мир движений производителей и Arduino, мы только прикоснемся к поверхности, и есть очень много вещей, которые вам нужно изучить. Вот некоторые ресурсы, которые, по нашему мнению, могут помочь вам поднять ваши навыки на новый уровень.

Hackaday

Это отличный ресурс для всевозможных чудес техники. В нем много проектов, связанных с Arduino, и простые руководства по большинству проектов. Однако этот веб-сайт не ограничивается только Arduino; у него есть различные другие ресурсы почти для всех технологий DIY. Он содержит отличную коллекцию постов и информации для разжигания воображения.

См. <http://hackaday.com/>.

Блог Arduino

Это отличный ресурс для всех новостей, связанных с Arduino. Он включает в себя все новейшее оборудование, связанное с Arduino, а также программные проекты. Это также одно из лучших мест, где можно быть в курсе работы, которую выполняет команда Arduino.

См. <https://blog.arduino.cc/>.

Журнал Make

Это журнал для любителей, посвященный всем видам технологий. В его блоге рассказывается о всевозможных интересных технологиях, которые можно сделать своими руками (DIY), и о проектах для вдохновения. Вы можете найти полезные ресурсы / проекты Arduino в разделе «Arduino» на сайте.

См. <http://blog.makezine.com/>.

Bildr

Bildr - отличный ресурс, который предоставляет подробные учебные пособия, опубликованные сообществом. Помимо понятных руководств, Bildr также предлагает отличные иллюстрации, упрощающие отслеживание подключений. Многие из руководств основаны на Arduino и содержат весь код и информацию о компонентах, которые вам понадобятся.

См. <Http://bildr.org/>.

Instructables

Это веб-платформа документации, которая позволяет людям делиться своими проектами с пошаговыми инструкциями по их созданию. Instructables - это не только про Arduino или даже технологии, поэтому вы можете найти там целый мир интересного материала.

См. <http://www.instructables.com/>.

Tronixstuff

Веб-сайт Джона Боксолла - отличный ресурс для изучения Arduino. На его сайте размещены десятки различных проектов и демонстраций Arduino.

См. <http://tronixstuff.com/tutorials/>.

Adafruit

Adafruit - это интернет-магазин, репозиторий и форум для всевозможных наборов, которые помогут вам заставить ваши проекты работать. Вероятно, это один из лучших онлайн-ресурсов для изучения Arduino и проверки некоторых интересных проектов.

См. <https://learn.adafruit.com/>.

Все о схемах

Если вы хотите узнать больше об электронике и схемотехнике, это может быть лучшим местом для вас.

См. <http://www.allaboutcircuits.com/>.

Хакерские пространства

Хакерские пространства - это физические пространства, где художники, дизайнеры, производители, хакеры, программисты, инженеры или кто-либо еще могут встречаться, чтобы учиться, общаться и сотрудничать в проектах. Если вы ищете вдохновения и хотите познакомиться с замечательными людьми, делающими потрясающую работу, найдите хакерское пространство поблизости от вашего района и учитесь у мастеров.

См. <http://hackerspaces.org/>.

Форум Arduino

Это отличное место, чтобы получить ответы на конкретные вопросы Arduino. Вы часто обнаруживаете, что другие люди работают над теми же проблемами, что и вы, поэтому при тщательном поиске вы, вероятно, найдете ответ практически на любую проблему.

См. [Http://arduino.cc/forum/](http://arduino.cc/forum/).

Резюме

В этой главе представлены решения некоторых распространенных проблем, с которыми вы можете столкнуться при работе с Arduino. В последнем разделе главы упоминается несколько проектов DIY, которые вы, возможно, захотите реализовать, используя предоставленные ресурсы.

Если вы столкнетесь с какой-либо проблемой в любом из проектов, упомянутых в книге, или заметите какие-либо опечатки / ошибки в любой из глав, не стесняйтесь писать нам по адресу Samarth@outlook.com и / или Utsav_shah01@outlook.com с темой как название книги.