Community Experience Distilled

Проекты домашней автоматики на Arduino

Автоматизируйте свой дом с помощью мощной платформы Arduino

Marco Schwartz

Проекты домашней автоматики на Arduino

Автоматизируйте свой дом с помощью мощной платформы Arduino

Marco Schwartz

об авторе

Марко Шварц - инженер-электрик, предприниматель и блогер. Он имеет степень магистра электротехники и информатики от Supélec, Франция, и степень магистра микроэлектроники от EPFL, Швейцария.

Имеет более 5 лет опыта работы в области электротехники. Его интересы связаны с электроникой, домашней автоматики, платформами Arduino и Raspberry Pi, проектами оборудования с открытым исходным кодом и 3D-печатью.

Он управляет несколькими веб-сайтами на базе Arduino, включая веб-сайт Open Home автоматики, посвященный созданию систем домашней автоматики с использованием оборудования с открытым исходным кодом.

Он написал еще одну книгу по домашней автоматике и Arduino под названием «Домашняя автоматика с Arduino,создание пространства ». Он также опубликовал книгу о том, как создаватьпроекты Интернета вещей с помощью Arduino, под названием «Интернет вещей с Arduino Yun, Packt Publishing».

Фрэнсис Перее - профессор профессионального образования Consejería de Educación de la Junta de Andalucía в Испании с более чем 14-летним опытом.

Он специализируется на системном администрировании, веб-разработке и системах управления контентом. В свободное время он работает фрилансером и сотрудничает, в частности, с - multimedia, небольшой дизайн-студией в Кордове, где он работает системным администратором и главным веб-разработчиком.

Он также был техническим обозревателем SketchUp 2014 для архитектурной визуализации, Томаса Блейхера и Робина де Йонга, и Интернета вещей с Arduino Yún, Марко Шварца, оба - Packt Publishing.

Когда он не сидит за компьютером и не возится в своей мастерской, он может бегать или ездить на велосипеде по рельсам и холмам или, как пчеловод, ухаживать за своими пчелами в округе Аксаркия, где он живет.

Я хотел бы поблагодарить мою жену Саломе и троих наших детей - Паулу, Альваро и Хави - за всю поддержку, которую они мне оказывают, хотя мы все очень заняты. Нет слов, чтобы выразить им свою благодарность.

Также хочу поблагодарить своих коллег по мультимедиа и терпеливых студентов. Необходимость быть на требуемом уровне - вот что заставляет меня двигаться вперед.

Содержание

Предисловие 1

Глава 1: Создание беспроводных детекторов движения XBee 7

- Компоненты и программы 8
- Конфигурация оборудования 9
- Взаимодействие датчика PIR с Arduino 11
- Программирование детектора движения XBee 13
- Графический интерфейс для детекторов движения XBee 15
- Резюме 19

Глава 2: Управление светом с вашего телефона или планшета 20

- Компоненты и программы 21
- Конфигурация оборудования 24
- Тестирование реле и Wi-Fi соединения 27
- Графический интерфейс для управления реле 32
- Тестирование графического интерфейса 33
- Резюме 35

Глава 3: Измерение температуры с помощью Bluetooth 37

- Компоненты и программы 38
- Конфигурация оборудования 40
- Создание скетча Arduino 44
- Тестирование датчика температуры и влажности 46
- Измерение температуры и влажности удаленно 47
- Резюме 50

Глава 4: Метеостанция в облаке с Xively 53

- Компоненты и программы 54
- Подключение различных компонентов 55
- Тестирование датчиков 56
- Настройка учетной записи Xively 58

Создание скетча Arduino 61
Авторизация и отображение данных на Xively 64
Резюме 66
Глава 5: Следите за потреблением энергии в облаке 67
Компоненты и программы 68
Выполнение аппаратных подключений 70
Тестирование проекта 73
Настройка учетной записи Xively 75
Отправка данных об энергопотреблении в Xively 77
Резюме 80
Глава 6: Взлом коммерческого устройства домашней автоматики 81
Компоненты и программы 82
Конфигурация оборудования 85
Controlling the device from your computer 87
Building a graphical interface 89
Резюме 93
Глава 7: Создайте свою систему домашней автоматики 95
Компоненты и программы 96
Создание системы Arduino с нуля 98
Тестирование системы Arduino 100
Разработка печатной платы 101
Изготовление платы 107
3D-печать корпуса для вашего проекта домашней автоматики 108
Резюме 113

Предисловие

Домашняя автоматика - тема, которая существует уже много лет. Она включает в себя все, что вы можете себе представить, для управления и автоматики своего дома. Самый распространенный пример - это сигнализация вашего дома. Датчики движения, контактные датчики и центральное устройство, управляющее вашей системой сигнализации, обычно являются основными компонентами любой системы домашней автоматики.

Для домашней автоматики доступно бесчисленное множество устройств. Вы можете купить устройства домашней автоматики во многих магазинах и даже установить их у себя дома. Однако многие из этих систем очень дороги, их невозможно настроить под свои нужды, и они имеют устаревшие пользовательские интерфейсы.

С другой стороны, у нас есть платформа Arduino. Arduino - это платформа, которую вы можете использовать для быстрого создания прототипов электронных систем. Сейчас его используют миллионы людей по всему миру для создания более сложных систем. Это на самом деле идеальная платформа для создания систем домашней автоматики. Благодаря гибкости платформы Arduino, мы увидим, что она легко взаимодействует с различными датчиками и исполнительными механизмами, которые обычно используются во многих системах домашней автоматики. Она также может взаимодействовать со многими беспроводными системами, такими как Wi-Fi, Bluetooth или XBee.

В этой книге мы увидим, как создавать системы домашней автоматики с помощью Arduino. Сначала мы построим системы на основе имеющихся в продаже плат Arduino. Например, мы собираемся построить датчик температуры на базе Arduino и Bluetooth. Мы также собираемся интегрировать некоторые из этих систем в перспективу Интернета вещей, отправив некоторые данные непосредственно в облачную службу. Наконец, мы также увидим, как взламывать коммерчески доступные устройства и создавать собственные домашние системы автоматики с нуля.

О чем эта книга

Глава 1, Создание беспроводных детекторов движения XBee, посвящена очень распространенной теме домашней автоматики: детекторам движения. Мы собираемся построить рой детекторов движения на основе известного протокола XBee и Arduino. Мы также собираемся создать серверный интерфейс для отслеживания состояния детекторов движения XBee.

В главе 2 «Управление освещением с телефона или планшета» рассматривается еще одна популярная тема в домашних системах автоматики: дистанционное управление освещением. Мы собираемся связать реле с Arduino и чипом Wi-Fi, чтобы вы могли управлять светом в своем домашнем очаге не только со своего компьютера, но и с телефона или планшета.

Глава 3 «Измерение температуры с помощью Bluetooth» посвящена измерению температуры и влажности с помощью Arduino и передаче результатов обратно на ваш компьютер с помощью Bluetooth. Мы также собираемся создать простой интерфейс с использованием Python, чтобы вы могли получить доступ к измерениям, выполненным системой Arduino и Bluetooth.

Глава 4, Метеостанция в облаке с Xively, научит вас решать очень модную тему: Интернет вещей. Мы собираемся провести базовые измерения погоды на нашей плате Arduino, а затем передать эти данные через Wi-Fi в облако с помощью службы под названием Xively. Используя этот сервис, мы сможем удаленно отслеживать наши данные из любой точки мира.

Глава 5 «Мониторинг энергопотребления в облаке» начинается с использования той же облачной службы, которая использовалась в главе 4 «Метеостанция в облаке» с Xively. Однако в этом случае мы собираемся отправлять данные о потреблении энергии в облако. Таким образом, вы сможете отслеживать данные непосредственно из интерфейса Xively.

В главе 6, «Взлом коммерческого устройства домашней автоматики», рассматривается идея действовать по-другому. Вместо того, чтобы создавать новую систему домашней автоматики на базе Arduino, мы собираемся взломать существующее устройство, чтобы вы могли управлять им со своего компьютера. Мы собираемся создать выключатель питания с USB-управлением, чтобы вы могли управлять любым устройством прямо с интерфейса, работающего на вашем компьютере.

Глава 7 «Создание собственной системы домашней автоматики» идет еще дальше и показывает вам, как создать собственную систему домашней автоматики на базе Arduino. Мы расскажем, как создать систему Arduino с нуля, как разработать собственную печатную плату и наконец, как спроектировать и напечатать на 3D-принтере индивидуальный корпус для вашего проекта.

Что вам понадобится для этой книги

Для создания всех проектов, представленных в этой книге, вам потребуется несколько аппаратных и программных компонентов. Конечно, вы можете просто прочитать описание проектов и так научиться. Тем не менее, я действительно рекомендую заняться проектами, чтобы действительно научиться создавать собственные системы домашней автоматики на базе Arduino.

Необходимые аппаратные компоненты подробно описаны в начале каждой главы, однако для всех проектов вам действительно понадобится плата Arduino. Все проекты книги основаны на плате Arduino Uno, описание которой приведено по следующему URL-адресу:

`http://arduino.cc/en/Main/arduinoBoardUno`

Программы, которое мы будем использовать во всех главах книги - следующие:

- Arduino IDE (`http://arduino.cc/en/main/software`).
- Вам понадобится несколько библиотек. Они подробно описаны в каждой главе, где они необходимы.
- Для некоторых проектов вам также понадобится веб-сервер, работающий на вашем компьютере. Я рекомендую использовать какое-нибудь программное обеспечение, которое объединяет веб-сервер и базу данных и обрабатывает все детали за вас.

  Если вы используете Windows, я рекомендую использовать EasyPHP. Вы можете скачать его по следующему URL-адресу:

  `http://www.easyphp.org/`

  Для OS X я рекомендую использовать MAMP. Вы можете скачать его по следующему URL-адресу:

  `http://www.mamp.info/`

  Для Linux вы можете следовать инструкциям, приведенным в следующем URL-адресе, чтобы установить веб-сервер:

  `http://doc.ubuntu-fr.org/lamp`

  Убедитесь, что на этом этапе сервер работает. Мы собираемся использовать его в нескольких проектах этой книги.

Для кого эта книга

Эта книга предназначена для всех, кто хочет создавать собственные системы домашней автоматики на базе Arduino. На самом деле вам не нужно ничего знать об платформе Arduino,поскольку все проекты будут объяснены пошагово с четкими инструкциями. Единственное, что вам нужно, чтобы следовать проектам, описанным в этой книге, - это базовые знания электроники и программирования.

Эта книга также предназначена для любителей электроники, которые хотят узнать больше о выполнении проектов на платформе Arduino. Выполняя эти проекты по домашней автоматики, вы узнаете обо всех аспектах платформы Arduino: как взаимодействовать с датчиками и исполнительными механизмами с Arduino, как использовать беспроводные модули и даже как создать свою собственную систему Arduino с нуля.

Наконец, эта книга также предназначена для людей, желающих узнать больше об Интернете вещей с помощью Arduino. Почти все главы книги посвящены беспроводной связи, а две главы книги посвящены отправке некоторых данных в облако, чтобы их можно было отслеживать откуда угодно.

Соглашения

В этой книге вы найдете несколько стилей текста, которые различают разные виды информации. Вот несколько примеров этих стилей и объяснение их значения.

Кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, фиктивные URL-адреса, пользовательский ввод и дескрипторы Twitter показаны следующим образом:
«Что нас интересует, так это поле return_value, которое содержит результат функции digitalRead ()».

Блок кода устанавливается следующим образом:

```
String data = "";
data = data + timeString + "," + String(temperature) + "," +
  String(humidity) + "," + String(lightLevel);
```

Когда мы хотим привлечь ваше внимание к определенной части блока кода, соответствующие строки или элементы выделяются жирным шрифтом:

```
[default]
if (client) {
  // Process request
  process(client);

  // Close connection and free resources.
  client.stop();
}
```

Любой ввод или вывод командной строки записывается следующим образом:

```
# /digital/7/1
```

Новые термины и **важные слова** выделены жирным шрифтом. Слова, которые вы видите на экране, например, в меню или диалоговых окнах, отображаются в тексте следующим образом: «В Windows вы можете найти его, перейдя в **Панель управления | Сеть и Интернет | Просмотр состояния сети и параметров совместного использования.** "

Предупреждения или важные примечания отображаются в таком поле.

Советы и хитрости выглядят так.

Скачивание примера кода

Коды примеров можно скачать из прилагаемой папки

1

Создание беспроводных детекторов движения XBee

В этой главе мы собираемся построить проект вокруг очень распространенного датчика домашней автоматики: датчика движения. Вы когда-нибудь замечали те небольшие модули из белого пластика, которые находятся в верхних углах в некоторых помещениях,модули, которые становятся красными, когда вы идете под ними? Это именно то, что мы собираемся сделать в этом проекте.

Однако вместо использования проприетарных технологий, которые обычно используются в этих модулях, мы собираемся основывать нашу систему на Arduino. А для коммуникационной части мы собираемся использовать модули XBee, которые представляют собой маломощные радиомодули, которые широко используются с платформой Arduino. Эти модули основаны на протоколе ZigBee, который также используется во многих коммерческих системах домашней автоматики.

Вот основные выводы, которые мы увидим в этой главе:

- В первую очередь перечислим все аппаратные и программные компоненты, которые нам нужно для этого проекта. С этими компонентами мы построим один модуль датчика движения, состоящий из платы Arduino, датчика движения и одного модуля XBee.

- Затем мы протестируем этот первый модуль; мы также протестируем коммуникационную часть, отправив команды через последовательный монитор программного обеспечения Arduino.

- Наконец, мы собираемся создать графический веб-интерфейс, который централизует все данные наших датчиков XBee. Благодаря простому интерфейсу, основанному на веб-технологиях, вы сможете мгновенно увидеть, обнаружено ли движение в вашем доме.

Компоненты и программы

Для этого первого проекта нам потребуются платы Arduino, датчики движения PIR, а также некоторые модули XBee и шилды XBee, в зависимости от количества датчиков, которые вы хотите установить в своей системе. Всего для одного датчика вам понадобятся следующие компоненты:

- Плата Arduino R3
- Датчик PIR
- Модуль XBee серии 1
- Arduino XBee шилд

Датчик движения должен иметь три контакта: два для источника питания и один сигнальный. Он также должен использовать уровень напряжения 5 В, чтобы быть совместимым с платой Arduino Unoboard, которая также работает при 5 В.

Для модуля XBee я использовал модуль XBee Series 1, 1 мВт, с трассирующей антенной (что означает, что для него не требуется внешняя антенна). Вы также можете использовать модуль с внешней антенной, но тогда вам придется подключить антенну к модулю. Я использовал модули Series 1XBee для этого проекта, так как они проще в использовании, чем Series 2, которые обладают функциями, которые нам не нужны для этого простого проекта. Этот модуль имеет радиус действия около 100 метров без препятствий.

Вам также потребуется подключить модуль XBee к плате Arduino. Для этого каждый из моих модулей датчиков движения будет использовать шилд Arduino XBee от Sparkfun, но вы также можете использовать другие бренды. Ему просто нужно установить соединения между модулем XBee и платой Arduino.

Наконец, вам понадобится способ связи с этими модулями XBee с вашего компьютера. Я использовал другой модуль XBee (также Series 1, 1 мВт, с антенной), подключенный к плате проводника XBee от Sparkfun, которая в основном представляет собой интерфейсную плату USB, к которой вы можете подключить любой модуль XBee. Я использовал следующие компоненты для модуля, подключенного к компьютеру:

- XBee Explorer USB
- Модуль XBee серии 1

Что касается программного обеспечения, на вашем компьютере должна быть установлена последняя версия Arduino IDE. Для этого проекта использовалась Arduino IDE 1.0.5.

Вы также получите библиотеку REST для Arduino. Вы можете найти эту библиотеку по следующей ссылке:

```
https://github.com/marcoschwartz/aREST
```

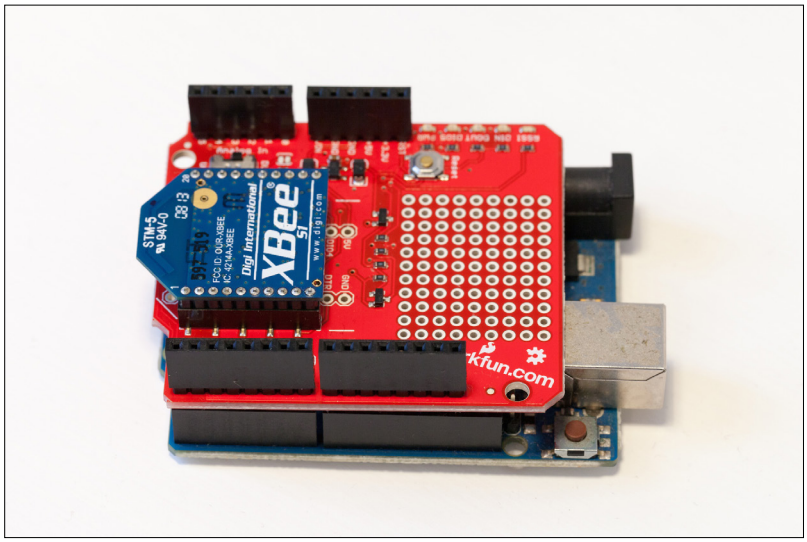
Чтобы установить библиотеку, извлеките все файлы из папки с именем daREST (или создайте эту папку). Затем поместите эту папку в папку / library в вашей основной папке Arduino. Вам также потребуется установить и запустить веб-сервер на вашем компьютере, чтобы вы могли использовать веб-интерфейс, который мы собираемся разработать в конце этой главы.

Чтобы настроить модули XBee, вам также необходимо установить программу XCTU. Вы можете найти его по следующему URL-адресу:

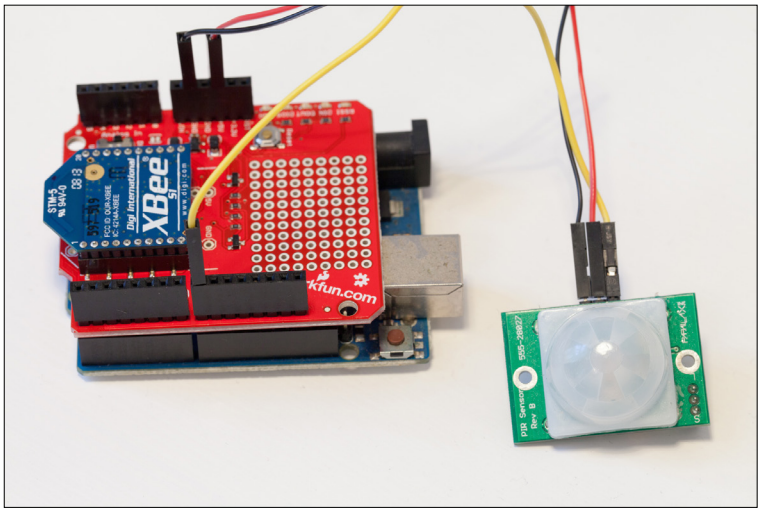
```
http://www.digi.com/products/wireless-wired-embedded-solutions/
zigbee-rf-modules/xctu
```

Аппаратное обеспечение

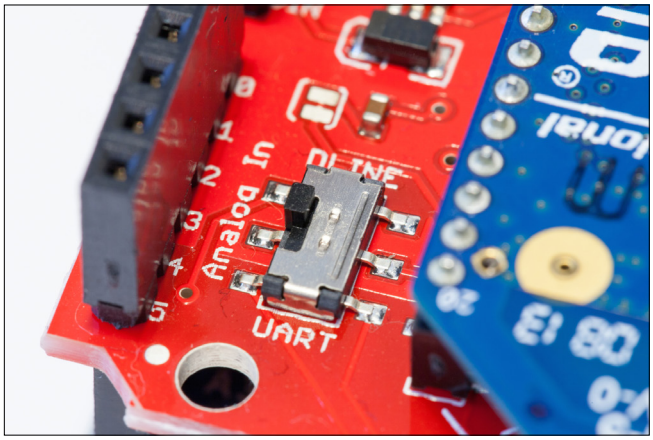
Аппаратная конфигурация этого проекта не очень сложна. Для каждого модуля датчика движения, который вы хотите построить, вам необходимо выполнить следующие шаги. Первый - подключить модуль XBee к плате XBee. Затем вам нужно подключить шилд к плате Arduino, как показано на следующем рисунке:



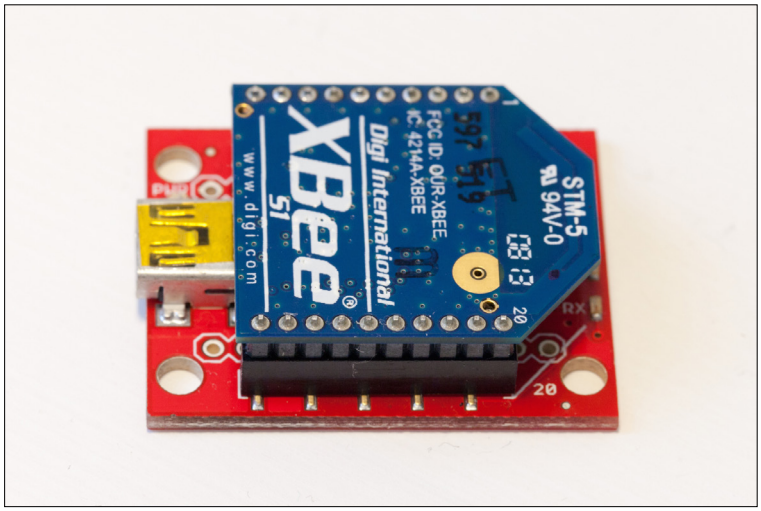
Теперь вы можете подключить датчик движения. Он имеет три контакта: VCC (+ Uпит), GND ("земля") и SIG (выходной сигнал). Подключите VCC к контакту Arduino 5V, GND к Arduino GND и SIG к контакту Arduino номер 8 (в примере кода используется контакт 8, но вы также можете использоватьлюбой цифровой контакт). Вы должны сделать что-то похожее как здесь:



Вам также нужно будет правильно установить перемычку на плате, чтобы мы могли загрузить скетч. На шилде XBee есть маленький переключатель рядом с модулем XBee, чтобы выбрать между подключением модуля XBee напрямую к последовательному интерфейсу платы Arduino (что означает, что вы больше не можете загружать скетчи) или его отключением. Поскольку нам нужно сначала загрузить скетч Arduino, вам нужно установить этот переключатель в положение DLINE, как показано на этом изображении:



Вам также нужно будет подключить плату XBee Explorer к компьютеру на этом этапе. Просто вставьте один модуль XBee в плату, как показано на следующем рисунке:



Теперь, когда это сделано, вы можете включить все, подключив плату Arduino и модуль проводника к компьютеру через USB-кабели.

Если вы хотите использовать несколько датчиков движения XBee, вам нужно будет повторить начало процедуры для каждого из них: собрать одну плату Arduino с шилдом XBee, один модуль XBee и один датчик движения. Однако, если у вас много датчиков, вам понадобится только один USB-модуль XBee, подключенный к вашему компьютеру.

Взаимодействие датчика PIR с Arduino

Во-первых, вы собираетесь оставить XBee в сторону и просто проверить, правильно ли работает датчик движения. Что вы сделаете в первом скетче, так это распечатаете показания датчика движения на последовательном порту. Это полный код для этой части, который вы можете просто скопировать и вставить в Arduino IDE:

```
// простой датчик движения
int sensor_pin = 8;

void setup() {
```

```
  Serial.begin(9600);
}

void loop() {

  // Прочитать данные датчика
  int sensor_state = digitalRead(sensor_pin);

  // Данные для печати
  Serial.print("Motion sensor state: ");
  Serial.println(sensor_state);
  delay(100);
}
```

Скачивание примеров кода и цветных изображений

можно осуществить по следующей ссылке:
https://github.com/marcoschwartz/aREST

Посмотрим, что делает этот код. Он начинается с объявления пина, к которому подключен датчик, в нашем случае 8. В функции setup () скетча мы инициализируем последовательное соединение с компьютером, чтобы мы могли распечатать результаты на последовательном мониторе.

Затем в части скетча, посвященной циклу (), мы считываем состояние датчика движения с помощью простой команды digitalRead () и сохраняем этот результат в переменной. Затем это состояние просто выводится на последовательный порт каждые 100 мс..

Теперь вы можете загрузить скетч на свою плату Arduino и открыть последовательный монитор. Вот что вы должны увидеть:

```
Motion sensor state:0
Motion sensor state:1
Motion sensor state:1
Motion sensor state:1
Motion sensor state:0
Motion sensor state:0
```

Если вы видите, как состояние датчика меняется, когда вы машете перед ним рукой, это означает, что датчик работает правильно и вы можете перейти к остальной части проекта.

Programming an XBee motion detector

Теперь вы собираетесь немного изменить скетч, чтобы он мог передавать состояние датчика на центральный интерфейс, работающий на вашем компьютере. Однако вы хотите передать не только состояние датчика движения, но и идентификатор датчика, который обнаруживает движение. Программирование детектора начинается с импорта необходимых библиотек:

```
// Libraries
#include <aREST.h>
#include <SPI.h>
```

Библиотека REST реализует REST API для Arduino. REST расшифровывается как REpresentational State Transfer и широко используется в веб-приложениях, таких как приложения «Software as a Service»(SaaS) (Программное обеспечение как услуга) . В нашем случае мы будем использовать эту библиотеку для стандартизации связи с центральным интерфейсом, который будет работать на компьютере. В этом проекте команды REST будут отправляться через соединение XBee, которое действует как последовательный порт с точки зрения Arduino.

После импорта библиотек вам необходимо объявить контакт датчика и идентификатор модуля следующим образом:

```
// Вывод и идентификатор датчика движения
int sensor_pin = 8;
String xbee_id = "2";
```

После этого вы можете создать экземпляр библиотеки REST, который будет обрабатывать запросы, поступающие из графического интерфейса:

```
// Create aREST instance
aREST rest = aREST();
```

В функции скетча setup () первым шагом является запуск последовательной связи. Будьте осторожны, так как скорость последовательного объекта должна быть такой же, как и скорость ваших модулей XBee, которая по умолчанию составляет 9600 бод:

```
// Start Serial
   Serial.begin(9600);
```

Вы также можете установить ID модуля:

```
   // Дать имя и идентификатор устройству
   rest.set_id(xbee_id);
```

Обратите внимание: если вы настраиваете более одного датчика, вам необходимо изменить идентификатор каждого настраиваемого датчика. Теперь, благодаря библиотеке REST, часть скетча с петлей () довольно проста. Нам просто нужно обрабатывать входящие запросы от компьютера, которые будут поступать через последовательный интерфейс XBee:

```
void loop() {

  // Обработка вызовов REST
  rest.handle(Serial);

}
```

Теперь скетч готов к использованию.

Весь код доступен в репозитории GitHub проекта:

`https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter1`

Вы можете загрузить скетч на свою плату Arduino, убедившись, что переключатель по-прежнему установлен на DLINE. Как только это будет сделано, вы можете протестировать код локально через последовательный монитор Arduino IDE. Откройте монитор последовательного порта, убедитесь, что скорость последовательного порта установлена на 9600, и введите следующее:

```
/digital/8/r
```

Это команда REST для считывания цифрового значения с пина номер 8 и возврата значения, чего мы и хотим достичь. В зависимости от текущего состояния датчика вы должны увидеть следующий возврат данных:

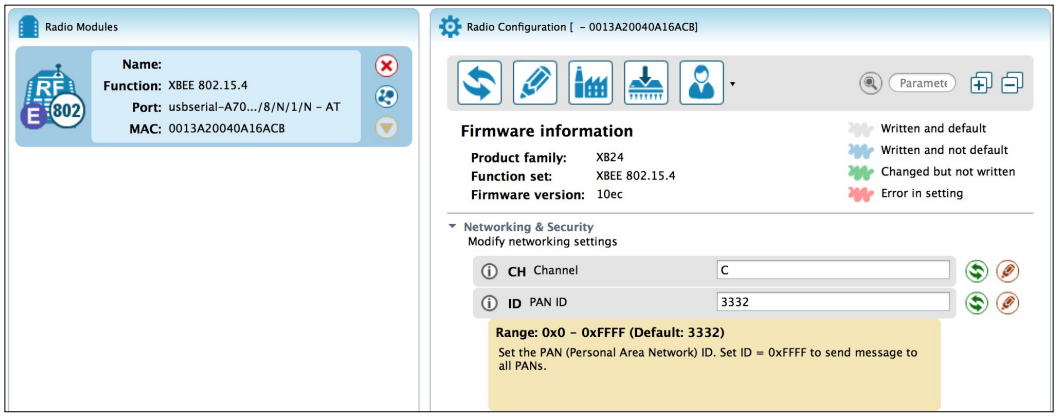
```
{"return_value": 0, "id": "2", "name": "", "connected": true}
```

Что нас интересует, так это поле return_value, которое содержит результат функции digitalRead (). Попробуйте помахать рукой перед датчиком, чтобы увидеть, изменится ли возвращаемое значение соответственно. Обратите внимание, что возвращаемые данные имеют формат JSON, что будет очень важно позже, когда мы будем обрабатывать эту информацию и отображать ее.

Теперь, когда вы уверены, что код работает, вы можете переключиться на связь XBee. Для этого просто переведите переключатель в положение UART на модуле XBee. Теперь последовательный порт платы Arduino напрямую подключен к модулю XBee.

По умолчанию все проданные модули XBee настроены на одну и ту же **Personal Area Network** (**PAN**) ID - 3332 (Идентификатор персональной сети). Это означает, что все модули будут получать данные от других модулей с одним и тем же идентификатором PAN. Для экспериментов вы можете оставить значение по умолчанию. Однако вы можете изменить это позже, если ваш сосед, например, также использует устройства XBee.

Чтобы продолжить, вставьте модуль XBee, который вы хотите изменить, в проводнике USB XBee и откройте инструмент XCTU XBee. Нажмите верхнюю левую кнопку, чтобы добавить новое устройство, и выберите последовательный порт USB-проводника. Вы должны получить следующий экран:



После этого вы сможете изменить PAN ID вашего устройства. Чтобы настроить все модули в вашей сети, просто повторите процедуру для каждого модуля XBee.

Создание графического интерфейса для ваших детекторов движения XBee

Теперь, когда оборудование полностью настроено, вы можете создать код на стороне сервера для чтения данных со всех ваших модулей датчиков движения. Для этого проекта я использовал только два модуля, но вы можете использовать больше модулей для проекта.

Как и для многих веб-приложений, наш центр управления детекторами движения XBee будет построен с использованием четырех разных языков: HTML, CSS, JavaScript и PHP. Мы собираемся использовать HTML и CSS для разработки самого интерфейса, JavaScript для обработки различных элементов интерфейса и установления связи с PHP и, наконец, использовать PHP для непосредственного взаимодействия с модулями XBee.

Первый шаг - это разработка интерфейса. У нас в основном будет несколько блоков на нашей странице, один блок соответствует одному модулю XBee. Например, это HTML-код для первого блока:

```
<div class="sensorBlock"><span class="sensorTitle">Sensor 1</span>
     <span class="display" id="display_1"></span>
</div>
```

Теперь вы можете взглянуть на код JavaScript, который будет обрабатывать различные элементы интерфейса. Обратите внимание, что весь этот код включен в специальный файл JavaScript. Здесь мы собираемся регулярно проверять значение датчика с помощью команды digitalRead (). Для этого мы должны вызвать PHP-файл с правильной командой в качестве аргумента:

```
$.get( "xbee.php", {command: "/digital/8/r"}, function( data ) {
```

Результатом использования файла PHP являются некоторые строковые данные, отформатированные в формате JSON. Чтобы фактически создать пригодный для использования объект из этой строки, мы можем использовать функцию parseJSON () jQuery:

```
json_data = jQuery.parseJSON(data);
```

Теперь у нас есть объект JavaScript, который имеет те же свойства, что и поля данных внутри контейнера JSON. Например, чтобы получить идентификатор сенсорного модуля, вы можете просто ввести:

```
var sensorID = json_data.id;
```

Теперь, когда мы знаем, какой датчик вернул некоторые данные, мы можем прочитать поле return_value внутри объекта JSON, чтобы узнать состояние датчика движения в этом модуле. Если это значение равно 0, мы устанавливаем серый цвет фона блока дисплея этого датчика. Если он равен 1, значит, было обнаружено какое-то движение. В этом случае мы меняем цвет фона на оранжевый следующим образом:

```
if (json_data.return_value == 0){
  $("#display_" + sensorID).css("background-color","gray");
}
else {
  $("#display_" + sensorID).css("background-color","orange");
}
```

Давайте теперь посмотрим подробности этого PHP-файла, который вызывается каждый раз. В коде JavaScript мы видели, что файл PHP также получает аргумент, называемый командой. Эта переменная извлекается внутри файла PHP:

```
$command = $_GET['command'];
```

Внутри этого файла нам также необходимо включить php_serialclass:

```
include "php_serial.class.php";
```

Нам также необходимо указать правильный последовательный порт для вашего устройства XBee Explorer:

```
$serial_port = "/dev/cu.usbserial-A702LF8B";
```

Вам придется изменить это в соответствии с вашими настройками. Чтобы узнать, какой последовательный порт используется платой проводника XBee, просто найдите устройства, начинающиеся с cu, в папке / dev, если вы используете Linux или OS X. Если вы используете Windows, вы увидите список последовательных портов на своей панели конфигурации.

Теперь вам нужно настроить различные параметры последовательного соединения, которое мы собираемся открыть с платой:

```
$serial = new phpSerial;
$serial->deviceSet($serial_port);
$serial->confBaudRate(9600);
$serial->confParity("none");
$serial->confCharacterLength(8);
$serial->confStopBits(1);
```

Теперь вы можете открыть соединение:

```
$serial->deviceOpen();
```

Теперь мы можем отправить команду и прочитать ответ:

```
$serial->sendMessage($command . "\r");
$answer = $serial->readPort();
```

Теперь мы закрываем последовательное соединение:

```
$serial->deviceClose();
```

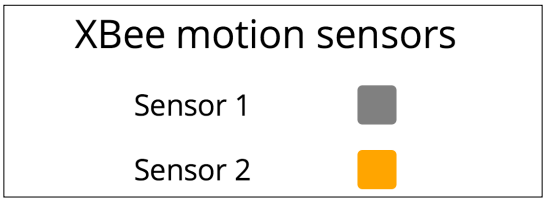
Наконец, мы отправляем обратно прочитанные данные:

```
echo $answer;
```

Теперь мы готовы протестировать наш проект. Конечно, вы можете найти весь код для этой части в репозитории проекта GitHub на следующем веб-сайте:

`https://github.com/openhomeautomation/arduino-home-автоматики/tree/master/chapter1`

Убедитесь, что ваш веб-сервер запущен, и что все файлы проекта помещены в папку в основной папке вашего веб-сервера. Теперь вы можете перейти в эту папку, обычно доступную при вводе локального хоста в вашем любимом веб-браузере. Откройте файл HTML, и вы должны увидеть что-то вроде следующего снимка экрана:



Обратите внимание, что на этом примере снимка экрана второй датчик в моей системе уже определил движение. Теперь попробуйте со своими датчиками, помахав рукой перед ними; вы должны сразу увидеть, что состояние датчиков меняется на интерфейсе.

Конечно, изменяя код HTML и JavaScript, вы можете легко изменить этот интерфейс, чтобы адаптировать его для большего количества датчиков.

Кроме того, имейте в виду, что эти модули не нужно напрямую подключать к вашему компьютеру после загрузки на них кода. Вы можете просто отсоединить их от компьютера и питать от простого батарейного блока (например, плата Arduino Uno может питаться до 12 В).

Обратите внимание, что вы также можете получить доступ к этому интерфейсу с мобильного телефона или планшета, просто открыв локальную папку хоста на вашем компьютере. Конечно, для доступа к интерфейсу ваш компьютер должен быть включен и запущен веб-сервер.

Резюме

Давайте теперь подведем итоги того, что мы сделали в этом первом проекте книги. Мы создали датчики движения XBee, основанные на Arduino, и узнали, что вы можете разместить их в любом месте своего дома. Они взаимодействуют с центральным интерфейсом, где вы можете видеть состояние каждого датчика в реальном времени.

Конечно, мы могли использовать для этого проекта другие устройства беспроводной связи, например Wi-Fi. Тем не менее, модули XBee очень энергоэффективны по сравнению с модулями Wi-Fi, поэтому в этом проекте есть смысл использовать XBee, поскольку вы, возможно, захотите запитать эти датчики движения от батарей.

Давайте посмотрим, каковы были основные выводы этой главы. Сначала мы выбрали компоненты для нашего проекта и построили наш первый датчик движения на основе XBee, используя Arduino и датчик движения PIR. Мы также подключили модуль XBee к нашему компьютеру через USB, чтобы он может общаться с другими модулями.

Затем мы протестировали наш модуль датчика движения, сначала проверив ИК-датчик, подключенный к плате Arduino Uno. На этом этапе мы также протестировали скетч, который использовали для доступа к плате через XBee.
Наконец, мы создали веб-интерфейс для визуализации информации, поступающей от различных датчиков движения XBee, в реальном времени. Этот интерфейс был протестирован с двумя модулями, но он легко расширяется до многих других модулей XBee.

В следующей главе мы займемся еще одной огромной областью домашних автоматов: управлением освещением. И для этого проекта мы не будем использовать XBee, но мы собираемся взаимодействовать с Arduino с чипом Wi-Fi для управления освещением с любого мобильного устройства.

2

Управляйте светом с телефона или планшета

В этой главе мы рассмотрим еще один очень распространенный проект домашней автоматики: управление освещением. Есть много устройств, которые вы можете купить и которые дадут вам возможность управлять устройствами в вашем доме, но эти устройства обычно имеют высокую цену. И даже если они беспроводные, они обычно идут с дистанционным управлением.

В этом проекте мы собираемся применить другой подход и управлять светом прямо с вашего компьютера, телефона или планшета, и все это через Wi-Fi. Следующие основные выводы из этой главы:

- Во-первых, мы собираемся создать аппаратную часть нашего проекта, подключив чип Wi-Fi к Arduino, а устройство, которым мы хотим управлять (здесь, простую лампу), к релейному модулю. На самом деле мы будем использовать в этом проекте два релейных модуля, но их, конечно, можно расширить до любого количества, которое вы захотите.
- Затем мы собираемся написать базовый код для тестирования различных частей проекта. Мы проверим, что модуль реле работает правильно и что чип Wi-Fi действительно может быть подключен к сети Wi-Fi вашего дома.
- Наконец, мы создадим базовый графический интерфейс, чтобы вы могли управлять своими устройствами не только с компьютера, но и с мобильного устройства, например смартфона или планшета.

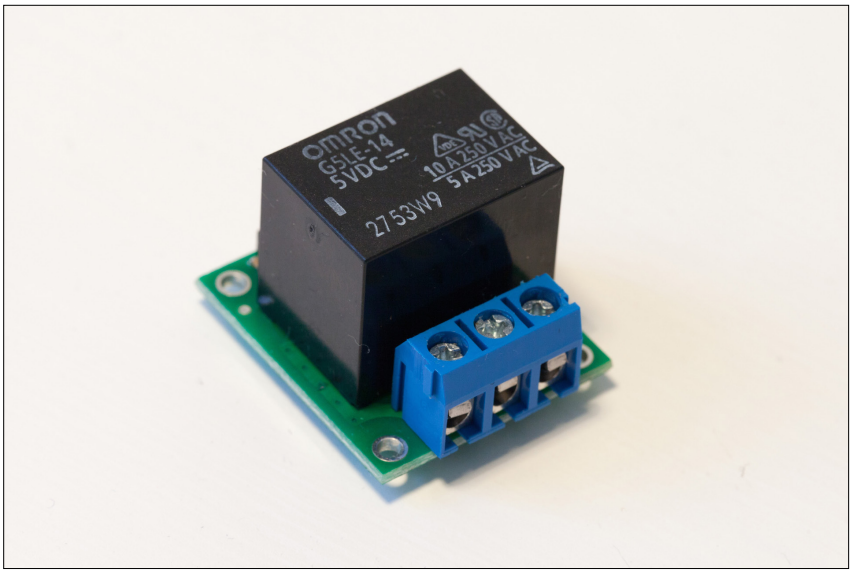
Компоненты и программы

Давайте сначала посмотрим, что вам понадобится для создания этого проекта. Подобно многим другим проектам этой книги, этот проект основан на платформе Arduino, и мы снова будем использовать плату Arduino Uno. Обратите внимание, что плата Arduino Mega также будет работать.

Затем вам также понадобятся несколько релейных модулей, чтобы вы могли управлять лампами. Реле - это в основном электромеханический переключатель. Обычно они используются, когда вы хотите управлять устройствами на 220В. Для получения дополнительной информации о реле вы можете посетите следующую веб-страницу:

```
http://en.wikipedia.org/wiki/Relay
```

В этом проекте я использовал релейный модуль на 5 В, который может коммутировать устройства до 250 В (таким образом, поддерживая устройства на 110 и 230 В) и максимум 10 А. На следующем изображении показан модуль реле:

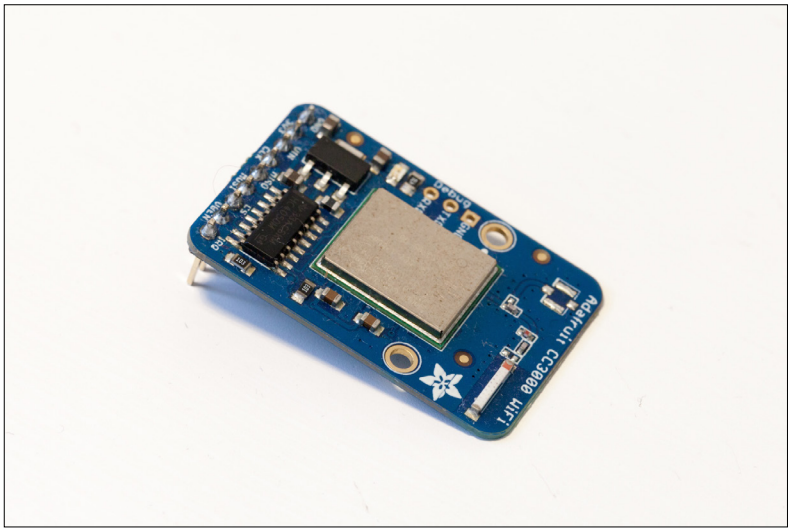


Я использовал два из них, так как хочу управлять двумя лампами со своей платы Arduino. Конечно, проект будет работать, если вы захотите добавить больше реле или просто использовать одно.

Вам не обязательно покупать этот специальный релейный модуль, вы можете приобрести его от любого из ваших любимых брендов. Он просто должен иметь аналогичные характеристики: быть совместимым с 5 В для командной части (так чтобы он работал с Arduino) и иметь возможность обрабатывать напряжение и ток устройства, которым вы хотите управлять. Например, модуль, который вы увидите в этом проекте, может работать с напряжением до 230 В и 10 А, что означает, что модуль может работать с устройством, потребляющим до 2300 Вт энергии.

Поскольку мы будем использовать устройства на 110 или 230 В с этими реле, вам необходимо принять особые меры предосторожности. Никогда не прикасайтесь к реле, когда оно подключено к розетке с помощью кабеля. По возможности спрячьте реле, где к нему не смогут получить доступ посторонние люди, например, в пластиковый корпус.

Теперь вам нужен модуль Wi-Fi. Код этого проекта специфичен для микросхемы Wi-Fi CC3000. Этот чип широко используется в сообществе Arduino, и вы найдете множество руководств и ресурсов в Интернете, в которых используется этот конкретный чип. Я использовал коммутационную плату от Adafruit, содержащую этот чип. Коммутационная плата в основном представляет собой печатную плату (PCB), которая объединяет интересующий вас чип, а также все дополнительные компоненты, необходимые для правильной работы чипа. Это изображение платы, которую я использовал:



Опять же, у вас есть довольно много вариантов при выборе этой платы. Вы можете выбрать один из ваших любимых производителей. Ему просто нужно интегрировать чип CC3000 Wi-Fi и быть совместимым с уровнями напряжения 5 В (что характерно для большинства плат CC3000). Вы также можете использовать шилд Arduino, который объединяет чип CC3000, например, от Adafruit.

Чтобы получить дополнительную информацию об этой плате и о том, как чип CC3000 Wi-Fi работает в целом, вы можете ознакомиться с документацией и примером библиотеки Adafruit CC3000 по следующей ссылке:

```
https://github.com/adafruit/Adafruit_CC3000_Library
```

Вам также понадобится макетная плата и несколько перемычек для выполнения всех необходимых аппаратных соединений.
Ниже приводится список компонентов, которые использовались в этой главе:

- Arduino Uno R3
- 5V relay
- Модуль Wi-Fi CC3000
- Макетная плата
- Перемычки

Что касается программного обеспечения, на вашем компьютере должна быть установлена последняя версия Arduino IDE. Вам также потребуется установить библиотеку REST для Arduino, которую вы можете найти по следующей ссылке:

```
https://github.com/marcoschwartz/aREST
```

Использование библиотеки REST также требует наличия библиотеки микросхем CC3000, которую мы будем использовать позже в этой книге. Вы можете найти ее по следующему адресу в Интернете:

```
https://github.com/adafruit/Adafruit_CC3000_Library
```

Вы можете найти библиотеку mDNS CC3000 на следующем веб-сайте:

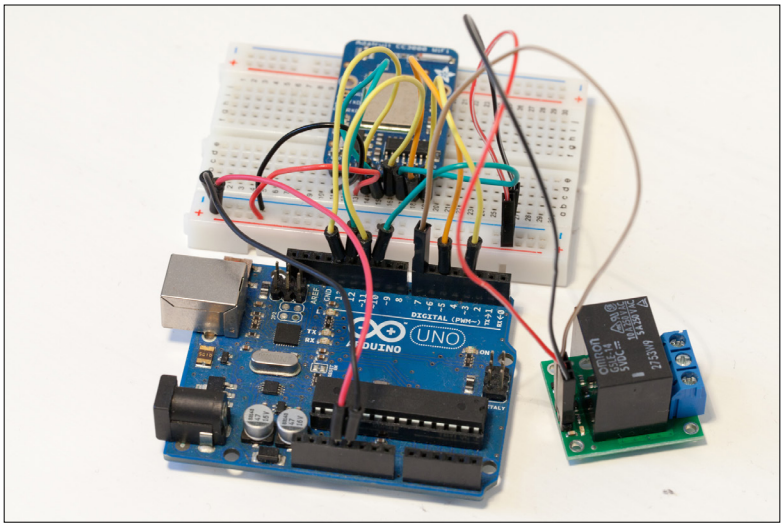
```
https://github.com/adafruit/CC3000_MDNS
```

Чтобы установить данную библиотеку, распакуйте все файлы в папку с именем REST и поместите ее в папку / libraries внутри основной папки Arduino (или создайте эту папку).

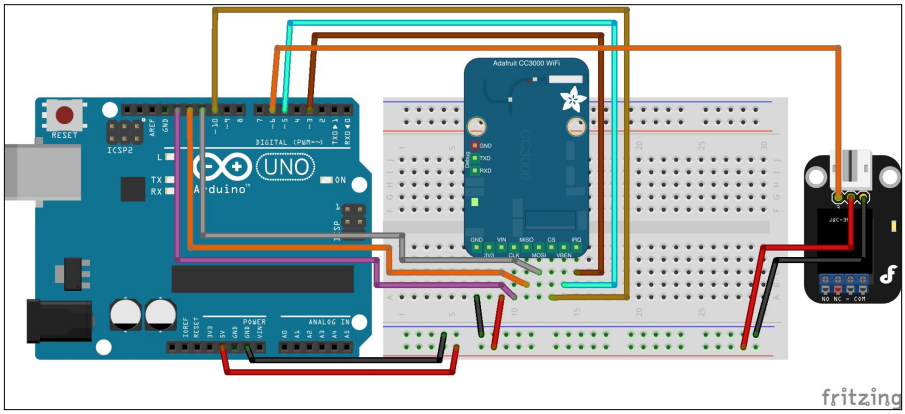
Вам также потребуется установить и запустить веб-сервер на вашем компьютере, чтобы вы могли использовать веб-интерфейс, который мы собираемся разработать в конце этой статьи.

Конфигурация оборудования

Давайте теперь посмотрим, как соединить разные части проекта. Сначала мы позаботимся о подключении всех компонентов к плате Arduino, в основном релейного модуля и микросхемы Wi-Fi CC3000. Чтобы дать вам представление, вот что у вас должно получиться:



Чтобы точно знать, какие провода и контакты вы должны подключить, на следующем изображении описаны все соединения проекта:

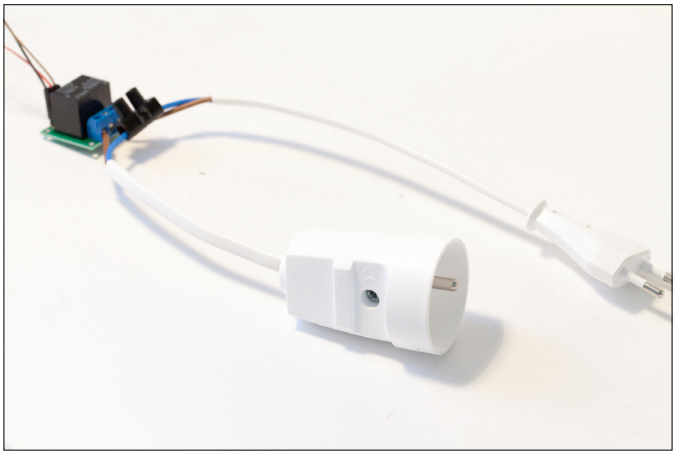


Обратите внимание, что на этой схеме показаны соединения только для одного модуля реле. Давайте сначала посмотрим, как подключить одиночный релейный модуль. Чтобы приступить к подключению оборудования, сначала разместите различные компоненты рядом друг с другом и подключите модуль CC3000 к макетной плате. В первую очередь займемся питанием. Подключите контакт заземления (GND) платы Arduino к контакту GND платы CC3000. Затем подключите вывод 5V платы Arduino Uno к выводу Vin платы CC3000.

Теперь посмотрим на микросхему CC3000. К модулю нужно подключить несколько контактов, поэтому внимательно следуйте инструкциям, иначе модуль может работать некорректно. Подключите вывод IRQ платы CC3000 к контакту номер 3 платы Arduino, VBAT к контакту 5 и CS к контакту 10. Затем вам необходимо подключить контакты SPI к плате Arduino: MOSI, MISO и CLK подключить к контактам 11. , 12, и 13 соответственно.

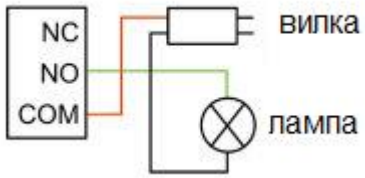
Затем мы закончим с релейным модулем. Модуль реле имеет три входных контакта: VCC, GND и SIG, который является сигнальным контактом. Просто подключите контакт номер 6 платы Arduino к контакту SIG модуля реле. Наконец, подключите GND к Arduino GND, а вывод VCC к Arduino 5V.

Теперь мы подключим лампу к релейному модулю. Для этого у вас должно быть два разных кабеля: вилка с вилкой и розетка. Следующее изображение иллюстрирует конечный результат, который вы должны получить:



Релейный модуль в основном имеет три выходных контакта: COM (для общего контакта), NC (для нормально замкнутого) и NO (для нормально разомкнутого). Мы хотим, чтобы контакт COM был подключен непосредственно к одному контакту вилки питания, NC не был подключен, а NO был подключен к другому контакту вилки питания.

На следующей схеме показаны соединения между модулем реле и лампой:



Конечно, вам нужно повторить эту операцию для каждого реле в вашем проекте. Вы можете просто подключить источник питания каждого нового модуля реле к линиям + 5V и GND на макетной плате, а вывод SIG нового реле - к свободному цифровому выводу на плате Arduino.

Тестирование реле и Wi-Fi соединения

Теперь, когда настройка оборудования завершена, мы можем начать писать код для тестирования нашего проекта. Сначала мы напишем простой скетч для проверки данного реле. Перейдите в свою Arduino IDE, и вы можете начать писать код. Наиболее важные части кода будут подробно описаны ниже, а полный код вы можете найти на GitHub репозиторий проекта по адресу https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter2

Первый шаг - объявить, к какому выводу подключено тестируемое реле:

```
const int relay_pin = 6;
```

Здесь мы используем переменную const int, которая похожа на #define, но мы применим именно этот тип, поскольку мы уверены, что константа имеет правильный тип. В функции скетча setup () нам нужно указать, что этот вывод является цифровым выходом с функцией pinMode ():

```
pinMode(relay_pin,OUTPUT);
```

Затем внутри функции loop () мы в основном собираемся активировать реле, подождать 5 секунд, снова выключить его, снова подождать 5 секунд и повторить процесс. Это делается с помощью следующего фрагмента кода:

```
// Активировать реле
digitalWrite(relay_pin, HIGH);

// ждем 5 секунд
delay(5000);

// Отключить реле
digitalWrite(relay_pin, LOW);

// ждем 5 секунды
delay(5000);
```

Теперь вы можете загрузить код на плату Arduino. Если вы правильно подключили компоненты, теперь вы должны видеть, как ваша лампа включается и выключается каждые 5 секунд. Если это не так, вам необходимо проверить все соединения , особенно между модулем реле и Arduino. Вы должны слышать щелчок реле всякий раз, когда состояние реле изменяется.

Убедитесь, что всякий раз, когда вы прикасаетесь к какой-либо части проекта, вилка кабеля питания отключена от розетки. В вашем проекте может быть оголенный кабель, и это может стать очень опасным, если вы будете работать над своим проектом, пока он работает.

Конечно, весь код для этого первого тестового эскиза доступен на репозиторий проекта на GitHub по адресу https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter2, в папке therelay_testfolder.

Теперь мы можем работать над следующим шагом, который будет последним шагом нашего проекта. На этом этапе мы просто собираемся проверить, работает ли чип CC3000 Wi-Fi, и проверить, можем ли мы управлять реле по беспроводной сети.

Начнем с объявления библиотек для нашего проекта:

```
#include
<Adafruit_CC3000.h>
#include <SPI.h>
#include <CC3000_MDNS.h>
#include <aREST.h>
```

Нам также необходимо указать, к каким пинам подключен чип CC3000. Если вы следовали разделу настройки оборудования, вам здесь нечего менять:

```
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
```

Затем мы должны объявить экземпляр чипа CC3000 из библиотеки CC3000 вместе с правильными пинами, которые мы только что определили:

```
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
  ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);
```

Затем есть некоторые параметры, которые вам действительно нужно изменить, чтобы чип знал, к какой сети Wi-Fi подключаться. Вы должны ввести свой SSID сети Wi-Fi и пароль:

```
#define WLAN_SSID       "yourSSID"
#define WLAN_PASS       "yourPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Обратите внимание, что вам также придется изменить настройки безопасности, если вы не используете WPA2. Чтобы получить список всех возможных настроек безопасности, вы можете взглянуть на примеры эскизов внутри библиотеки CC3000:

```
// Security can be WLAN_SEC_UNSEC, WLAN_SEC_WEP, WLAN_SEC_WPA
  or WLAN_SEC_WPA2
```

Вам также необходимо определить порт, на котором мы будем прослушивать входящие соединения. Я использовал 80, что довольно удобно, потому что вы можете напрямую ввести команды в своем браузере для доступа к плате:

```
#define LISTEN_PORT           80
```

Нам также необходимо определить сервер, который будет работать на плате Arduino и принимать входящие соединения:

```
Adafruit_CC3000_Server restServer(LISTEN_PORT);
```

Нам также необходимо создать пример службы mDNS, который будет весьма полезен для доступа к плате без наличия IP-адреса платы:

```
MDNSResponder mdns;
```

Наконец, вам нужно создать пример библиотеки aREST, чтобы мы могли обрабатывать запросы, поступающие из внешнего мира:

```
aREST rest = aREST();
```

Внутри функции setup () после инициализации платы и подключения к локальной сети Wi-Fi вы можете запустить службу mDNS. Здесь вы фактически даете имя своей плате Arduino. Здесь мы просто называем ее arduino, что означает, что она будет доступна позже, используя имя arduino.localin в вашем браузере:

```
if (!mdns.begin("arduino", cc3000)) {
    while(1); }
```

Однако может случиться так, что сервер mDNS не работает на вашем компьютере. Я приглашаю вас посетить страницу библиотеки mDNS CC3000 для получения более подробной информации. Если ваш компьютер несовместим, вам необходимо знать точный IP-адрес платы CC3000. Это делается с помощью следующей функции, которая будет печатать IP-адрес при запуске платы:

```
while (! displayConnectionDetails()) {
    delay(1000); }
```

Перед окончанием настройки нам еще нужно запустить наш сервер, работающий на плате Arduino. Это делается с помощью следующего фрагмента кода:

```
restServer.begin();
Serial.println(F("Listening for connections..."));
```

Теперь в функции скетча loop () мы сначала должны обновить службу mDNS:

```
  mdns.update();
```

И, наконец, проверьте, хочет ли какой-либо клиент подключиться к веб-серверу Arduino, и обработайте этого клиента с помощью библиотеки REST, если это так:

```
Adafruit_CC3000_ClientRef client = restServer.available();
rest.handle(client);
```

А благодаря библиотеке REST это все, что нам нужно сделать! Теперь вы можете загрузить код на плату (убедитесь, что вы изменили код на свой собственный SSID и пароль). Когда это будет сделано, откройте монитор последовательного порта, подождите несколько секунд, и в конце вы должны увидеть следующий вывод:

```
Listening for connections...
```

Если вы видите это сообщение, это означает, что ваша плата подключена к сети Wi-Fi. Вы также должны увидеть IP-адрес печатаемой платы. Запишите где-нибудь на бумаге, может скоро понадобиться.

Теперь вы можете закрыть монитор последовательного порта и открыть свой любимый веб-браузер. Сначала мы установим контакты реле как выходы, начиная с контакта номер 6. Просто введите следующую команду в поле URL-адреса вашего браузера:

```
http://arduino.local/mode/6/o
```

Если все работает нормально, в вашем браузере должно быть напечатано подтверждение:

```
Setting pin D6 to output
```

Если это не так, не паникуйте. Возможно, библиотека mDNS несовместима с вашим компьютером. Попробуйте выполнить ту же команду еще раз с IP-адресом вашей платы CC3000 вместо arduino.localalias. Например, введите следующую команду:

```
http://192.168.0.103/mode/6/o
```

Вы не можете сделать то же самое для вывода № 7, куда я подключил второе реле при сборке проекта. Теперь попробуем активировать первое реле через Wi-Fi. Все еще присутствуя в вашем браузере, введите:

```
http://arduino.local/digital/6/1
```

Вы должны сразу услышать переключение реле и включение лампы, если она все еще подключена к вашему проекту Arduino. Вы также получите немедленное подтверждение в вашем браузере:

```
Pin D6 set to 1
```

Вы также можете сделать то же самое для контакта номер 7, просто чтобы убедиться, что другое реле также может управляться через Wi-Fi. Конечно, если вы используете больше реле или подключили их по-другому, вы можете проверить их все на этом этапе, просто повторив процедуру, которую мы провели в этом разделе.

Конечно, весь код этого раздела доступен в GitHubrepository проекта на следующей веб-странице:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter2

Создание графического интерфейса для управления реле

Хотя для тестирования проекта удобно использовать команды непосредственно в браузере, на самом деле использовать проект для повседневного использования не так удобно. Вот почему мы собираемся создать специальный интерфейс, чтобы вы могли управлять своими реле со своего компьютера, просто нажимая на некоторые кнопки.

Поскольку для этого мы будем использовать веб-технологии, этот интерфейс также будет доступен с вашего мобильного телефона или планшета, если вы подключены к той же сети Wi-Fi.

Как было показано в первой главе этой книги, мы собираемся использовать смесь HTML, CSS, JavaScript и PHP для создания интерфейса. Давайте сначала посмотрим на важную часть файла HTML. В основном он состоит из нескольких блоков, каждый из которых соответствует реле.
Для каждого реле есть две кнопки: одна для включения реле, а другая для его выключения. Ниже приведен код, отвечающий за эту часть:

```
<div class="relayBlock"><span class="relayTitle">Relay 1</span>
    <button class="btn btn-block btn-lg btn-primary" type="button"
      id="1" onClick="relayClick(this.id)">On</button>
    <button class="btn btn-block btn-lg btn-danger" type="button"
      id="2" onClick="relayClick(this.id)">Off</button>
</div>
```

Чтобы сделать эти кнопки действительно полезными, нам нужно связать их с некоторым кодом JavaScript. Это делается внутри файла JavaScript. В основном мы получаем идентификатор кнопки, по которой была нажата кнопка для отправки правильной команды в систему. Например, кнопка с ID1 будет связана с реле, подключенным к контакту 6, и включит это реле:

```
if (clicked_id == "1"){
     $.get( "update_relay.php", {
  command: "/digital/6/1"} );
}
```

То же самое сделано для всех реле системы. Конечно, в зависимости от количества реле и конфигурации вашего оборудования вам потребуется изменить код в этом файле.

Давайте теперь посмотрим на этот PHP-файл, который вызывается каждый раз, когда мы нажимаем кнопку. Мы видели, что переменная с именем command передается при каждом нажатии кнопки. Внутри файла PHP нам нужно получить значение этой переменной:

```
$command = $_GET['command'];
```

Теперь, когда у нас есть команда, которую мы хотим отправить, мы можем построить строку, которая будет содержать URL-адрес, который мы должны вызвать. Это точно то же самое, что мы сделали в браузере при тестировании скетча. Обратите внимание: если служба mDNS у вас не работает, вам нужно будет ввести здесь IP-адрес вашей платы CC3000. Вот как мы создаем переменную, содержащую URL-адрес и команду для вызова:

```
$service_url = 'http://arduino.local'. $command;
```

Теперь мы можем инициализировать службу cURL, которую мы уже использовали в первом разделе книги. Это делается с помощью следующего фрагмента кода:

```
$curl = curl_init($service_url);
```

После этого мы выполняем команду, используя следующий оператор:

```
$curl_response = curl_exec($curl);
```

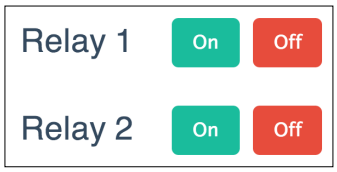
Затем закройте вызов cURL с помощью следующего кода:

```
$curl_close($curl);
```

Обратите внимание, что здесь нас не особо интересует ответ от платы Arduino, поэтому мы не будем использовать его вне файла PHP.

Тестирование графического интерфейса

Пришло время протестировать интерфейс. Убедитесь, что на вашем компьютере запущен веб-сервер. Также убедитесь, что все файлы интерфейса помещены в папку в корне вашего веб-сервера. Вы можете перейти в эту папку в своем браузере (набрав на локальном хосте) и открыть HTML-файл. Вот что вы должны увидеть:



Чтобы проверить интерфейс, просто попробуйте нажать на эти разные кнопки. Вы увидите, что свет переключается мгновенно, когда вы нажимаете правую кнопку. Если на этом этапе это не сработает, вы можете проверить несколько вещей.

Прежде всего, убедитесь, что ваш веб-сервер работает правильно. Также убедитесь, что вы ввели правильное имя для своей платы Arduino (или правильный IP-адрес) внутри файла PHP. Наконец, убедитесь, что ваш скетч Arduino работает правильно, набрав команды напрямую в браузере. Также обратите внимание, что на этом этапе вы также должны были установить контакты реле как выходы, как мы видели в разделе, где мы тестировали наш скетч Arduino.

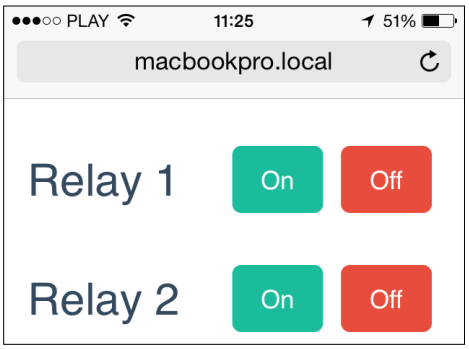
Конечно, весь код интерфейса доступен в репозитории GitHub проекта на следующей веб-странице:

`https://github.com/openhomeautomation/arduino-home-автоматики/tree/master/chapter2`

Теперь вы также можете перейти на свое мобильное устройство, смартфон или планшет и использовать этот интерфейс. Для этого просто узнайте имя вашего компьютера в сети.

Это действительно зависит от вашей операционной системы, но обычно оно находится в меню, аналогичном тому, которое называется Сетевые настройки в настройках вашей системы. Например, в OS X мой компьютер был доступен по имени macbookpro.local. В Windows вы можете найти его, перейдя в Панель управления | Сеть и Интернет | Просмотр состояния сети и параметров общего доступа. Если вы используете Linux, вы можете просто ввести имя хоста в терминале.

Введя имя своего компьютера на мобильном устройстве, вы сможете напрямую перейти в папку, где хранится интерфейс. Теперь вы можете использовать его так же, как на своем компьютере. Вот, например, так это выглядело на моем смартфоне:



Резюме

Подведем итог тому, что мы сделали в этой главе. В этом проекте мы создали систему беспроводных командных реле с вашего компьютера или мобильного устройства. Проект работает так же, как коммерческое устройство, которое вы можете купить в любом магазине оборудования, но вы получаете то преимущество, что можете настроить систему в соответствии со своими потребностями. И у вас также есть преимущество управления этими устройствами прямо со своего смартфона, а не с другого пульта дистанционного управления.

Давайте посмотрим, каковы были основные выводы из этой главы. Во-первых, мы выбрали компоненты для построения нашей системы, включая модули реле для подключения к управляемым устройствам. Также мы выбрали чип Wi-Fi, чтобы проект можно было подключить к вашей локальной сети Wi-Fi.

Затем мы собрали аппаратную часть проекта. Мы также подключили к проекту настольную лампу, чтобы мы могли управлять ею через Wi-Fi.

Затем мы написали код Arduino для тестирования различных компонентов проекта. Мы протестировали модули реле по отдельности, а затем проверили, можно ли подключить проект к вашей локальной сети Wi-Fi.

Наконец, мы создали веб-интерфейс, чтобы упростить управление различными устройствами, подключенными к системе. Поскольку этот интерфейс был построен с использованием веб-стандартов, таких как HTML и JavaScript, он доступен из любого браузера или мобильного устройства.

Конечно, можно добавить в проект больше плат Arduino, например, если вы хотите иметь реле в двух разных комнатах. Для этого сначала нужно дать каждой плате уникальное имя, например, arduino1.local, arduino2.local и так далее. Затем вам нужно будет соответствующим образом изменить код на стороне сервера.

3

Измерение температуры с помощью Bluetooth

В этой третьей главе книги мы собираемся использовать еще одну технологию для целей домашней автоматики. Мы будем использовать очень часто используемую технологию для датчиков, а именно Bluetooth.
Мы собираемся связать модуль Bluetooth с Arduino, провести некоторые измерения с помощью Arduino и передать эти данные обратно на ваш компьютер через Bluetooth. Для получения и отображения данных на вашем компьютере мы будем использовать язык программирования Python.

Ниже приведены основные моменты, которые мы увидим в этой главе:

- Сначала мы выберем различные аппаратные компоненты для проекта, включая плату Arduino, модуль Bluetooth и датчик температуры и влажности. На этом этапе мы также установим различные программные компоненты, необходимые для проекта.

- Затем мы фактически создадим аппаратную часть проекта. Для этого мы соединим различные компоненты вместе с помощью макетной платы и перемычек.

- Сразу после сборки оборудования мы собираемся написать скетч Arduino для нашего проекта. На этом этапе мы сможем протестировать ваше устройство, чтобы убедиться, что все подключения были выполнены правильно.

- Наконец, мы собираемся создать графический интерфейс на основе Python, который будет автоматически измерять данные и отображать их на вашем компьютере, и все это через интерфейс Bluetooth.

Компоненты и программы

Давайте сначала посмотрим, что нам нужно для создания этого проекта. Проект снова основан на плате Arduino Uno, но он также будет работать с другими платами Arduino, такими как Arduino Mega. Из-за различий в архитектуре, особенно в поведении последовательного порта, некоторые более новые платы, такие как Arduino Due или Arduino Leonardo, могут некорректно работать с кодом, который мы собираемся построить вэтом проекте.

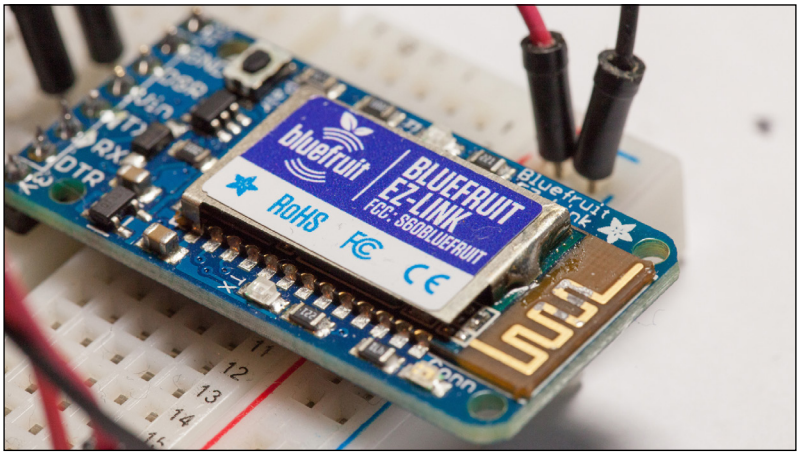
Затем другой важный компонент проекта - это модуль Bluetooth. Для этого проекта я использовал Bluetooth-модуль Bluefruit EZ-Link от Adafruit. Он объединяет чип Bluetooth и всю необходимую электронику для работы модуля с Arduino. Он действительно прост в использовании и взаимодействует напрямую с последовательным портом вашей платы Arduino.

Вы можете найти документацию по этой плате `Bluetooth` по следующему URL-адресу:

`https://learn.adafruit.com/introduction-bluefruit-ez-link`

Конечно, в этом проекте можно использовать и другие модули Bluetooth. Просто убедитесь, что у них есть прямой последовательный выход / вход (обычно обозначаемый как TX / RX). Если это так, они могут быть подключены к последовательному порту Arduino, и вы можете следовать инструкциям этого проекта.

На следующем изображении показан модуль, который я использовал для этого проекта, который уже интегрирован в завершенный проект, который мы собираемся построить в этой главе:



Вам также понадобится датчик для измерения температуры и влажности. В этом проекте я использовал датчик DHT, который представляет собой цифровой датчик температуры и влажности. Внутри датчика есть две части: термистор для измерения температуры и датчик влажности. Микросхема внутри датчика выполняет все аналого-цифровое преобразование. Его действительно удобно использовать с Arduino, и он поставляется с хорошей библиотекой Arduino.

На самом деле существует две версии этого датчика: DHT11 и DHT22, причем вторая более точная. Вы можете использовать ту версию, которую хотите; вам просто нужно будет изменить одну строку кода. Наряду с датчиком DHT вам понадобится резистор 4,7 кОм для правильной работы датчика. Это необходимо для создания подтяжки на линии данных. Мы увидим, как вставить этот резистор в следующем разделе. Вы можете узнать больше об этих датчиках по следующему URL-адресу:

```
https://learn.adafruit.com/dht
```

Вы также можете использовать датчик по вашему выбору или даже два отдельных датчика. Вы можете использовать, например, аналоговый датчик температуры и цифровой датчик влажности; это полностью зависит от вас. Вам просто нужно убедиться, что эти измерения отображаются в соответствующих переменных в скетче Arduino.

Для соединения между различными компонентами вам также понадобится макетная плата и несколько перемычек. Вам также понадобится конденсатор 10 мкФ для программирования платы Arduino через Bluetooth. Действительно, одна из замечательных особенностей этого типа модуля Bluetooth заключается в том, что он действует как последовательный порт, как и USB, поэтому вы можете программировать свою плату Arduino по беспроводной сети. Для этого необходим конденсатор, так как он будет использоваться между модулем Bluetooth и платой Arduino для управления выводом сброса и установки Arduino в режим программирования.

Ниже приводится список всех компонентов, которые использовались для этого проекта:

- Плата Arduino R3
- Bluefruit EZ-Link модуль Bluetooth
- Датчик DHT11 и резистор
- Конденсатор 10uF
- Макетная плата
- Перемычки

Благодаря этому последнему пункту вам на самом деле не нужно подключать плату через USB. Вы можете просто использовать внешний источник питания, например аккумулятор, для питания платы Arduino.

Ниже приводится список требований к программам:

- Для этого проекта вам понадобится библиотека DHT, которую можно найти по адресу `https://github.com/adafruit/DHT-sensor-library`.

  Устанавливаем библиотеку в папку / library вашей основной папки Arduino.

- Наконец, для этого проекта вам понадобится рабочая установка Python 2.7. Он уже установлен во многих операционных системах, но если это не так, вы можете найти его на `https://www.python.org/downloads/`
- Вам также потребуется последовательная библиотека Python, которая доступна по адресу `http://pyserial.sourceforge.net/`.

  Инструкции по установке этой библиотеки можно найти по адресу http://pyserial.sourceforge.net/pyserial.html#installation.

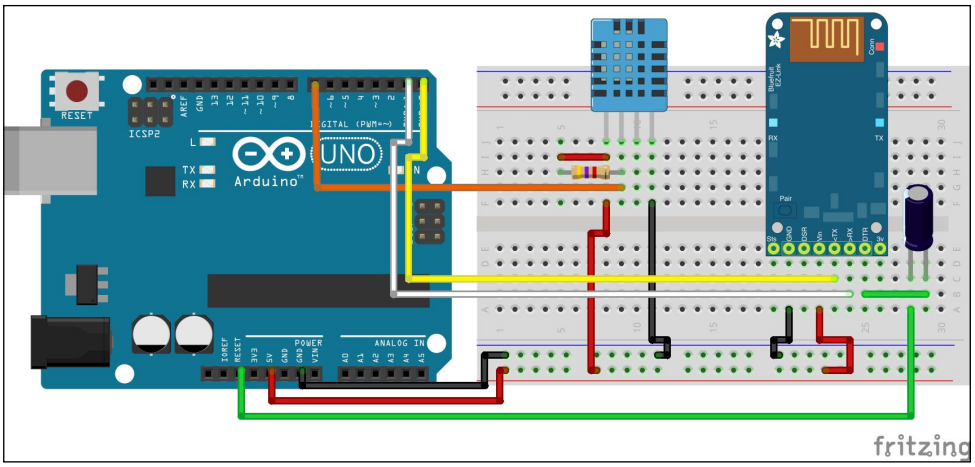
- Вы также можете установить дистрибутив Python, который уже предварительно установлен вместе с pySerial. Вы можете получить это на следующей веб-странице: `https://store.continuum.io/cshop/anaconda/`

Полный код этого проекта доступен в репозитории GitHub, соответствующем этой главе; он находится на следующей веб-странице: `https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter3`

Конфигурация оборудования

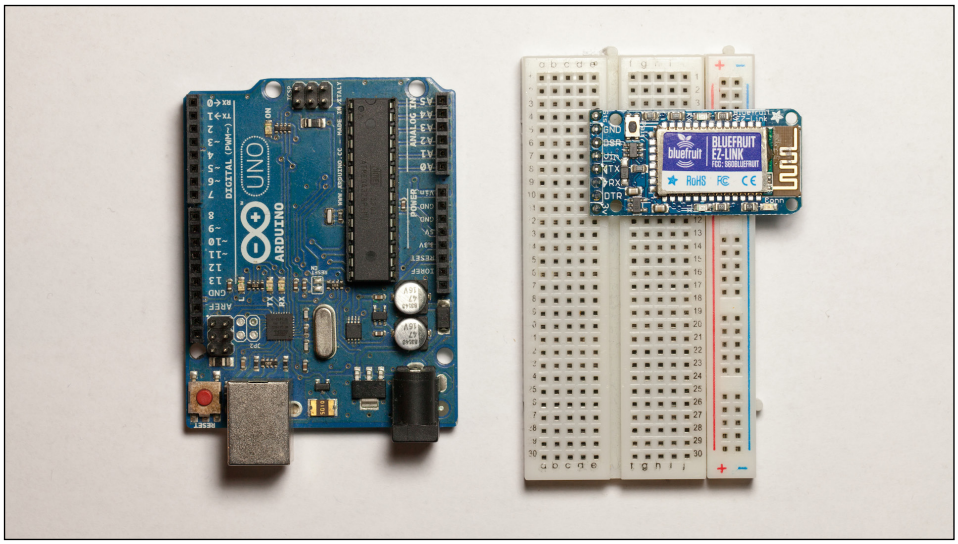
Теперь создадим наш проект. Нам нужно будет соединить различные модули и датчики с Arduino, подключить более мелкие компоненты (такие как резистор и конденсатор) и соединить все вместе с помощью перемычек.

Чтобы помочь вам в визуализации различных соединений, которые необходимо установить, вы можете взглянуть на следующее изображение:

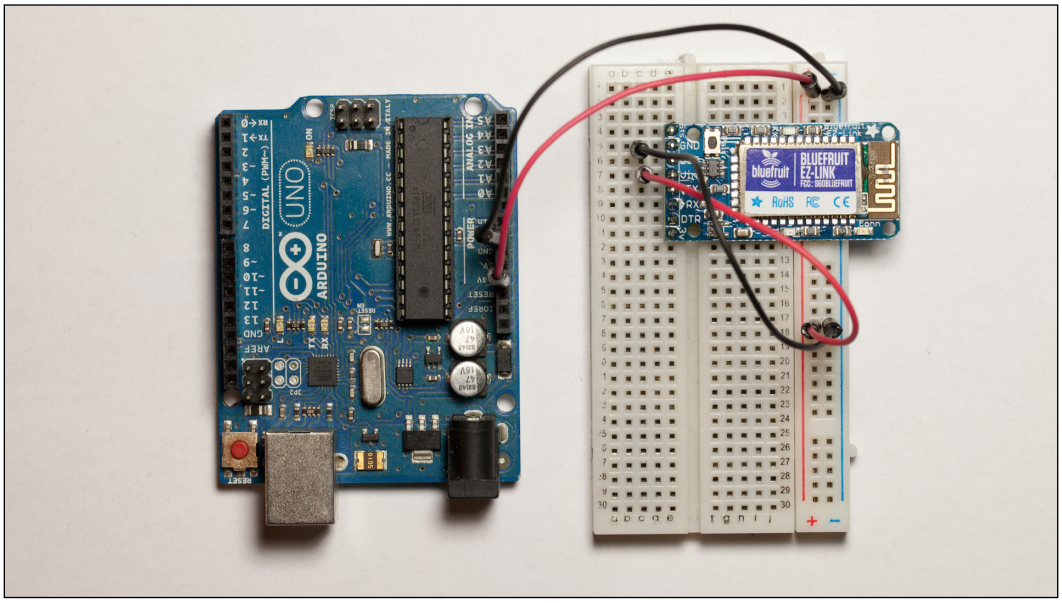


Чтобы собрать наш проект, выполним следующие шаги:

1. Первый шаг - разместить плату Arduino рядом с макетной платой, чтобы вы могли легко выполнять различные соединения. Вы также можете разместить здесь модуль Bluetooth. Когда вы закончите с размещением компонентов, у вас должно получиться что-то похожее на следующее изображение:

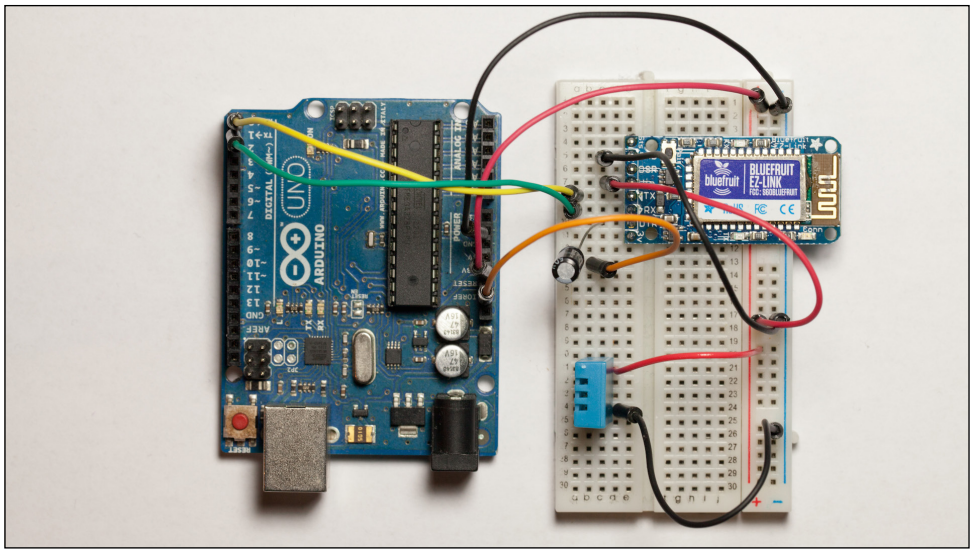


2. Теперь займемся подключением модуля Bluetooth, начиная с питания. Вы должны подключить контакт GND платы Arduino к одной из синих шин питания макетной платы, которая теперь будет отрицательной. Вы также можете подключить 5 В платы Arduino к шине питания Redpower, которая будет положительной шиной питания. На этом этапе вы также можете подключить модуль Bluetooth к источнику питания. Контакт Vin модуля Bluetooth должен быть подключен к красной шине питания, а контакт GND должен быть подключен к синей шине питания:



3. Теперь вы можете разместить конденсатор 10 мкФ между выводом DTR модуля Bluetooth и выводом RESET на плате Arduino. Убедитесь, что минус конденсатора подключен к выводу Arduino RESET.

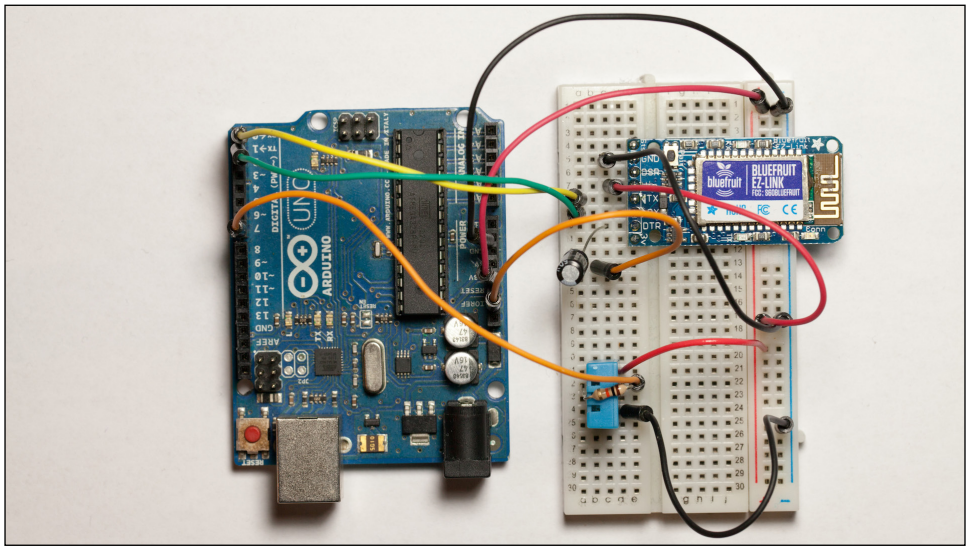
4. Затем вы можете поместить датчик DHT на плату. Сначала мы подключим датчик DHT к источнику питания. Контакт VCC (контакт 1) датчика DHT должен бытьподключен к красной шине питания, а контакт GND (контакт 4) должен быть быть подключенным к синей шине питания макета. Следующие изображения показывают, как это должно выглядеть в этот момент:



5. Наконец, вы должны подключить вывод SIG (вывод 2) датчика DHT к выводу 7 Arduino. Также необходимо подключить резистор 4,7 кОм между выводом VCC датчика (вывод 1) и выводом SIG. .

6. Для модуля Bluetooth вам необходимо подключить контакт TX к контакту RX Arduino,а контакт RX - к контакту Arduino TX.

Следующее изображение даст вам представление о том, что у вас должно получиться:



Создание скетча Arduino

Теперь давайте напишем простой скетч для измерения температуры и влажности от датчика DHT, когда заданная команда получена через последовательный порт.

Мы начинаем с включения библиотеки DHT, чтобы мы могли использовать датчик DHT11:

```
#include "DHT.h"
```

Подключим датчика пину 7 и введем:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

Обратите внимание, что если вы используете другой датчик DHT, вам нужно будет изменить код здесь. Например, если вы используете датчик DHT22, просто введите следующее:

```
#define DHTTYPE DHT22
```

Вам также нужно будет описать параметры датчика DHT и отправить тип датчика и контакт датчика в качестве аргумента:

```
DHT dht(DHTPIN, DHTTYPE);
```

В функции setup () скетча вам нужно будет инициализировать экземпляр датчика DHT и запустить последовательное соединение:

```
dht.begin();
Serial.begin(115200);
```

Здесь нам нужно использовать скорость 115 200 бод вместо стандартных 9600 бод, как это рекомендовано в спецификациях модуля Bluetooth.

Затем в функции loop () мы собираемся постоянно проверять состояние пробного порта, чтобы узнать, была ли получена команда с компьютера:

```
if (Serial.available()) {
```

Если это так, мы зачитаем содержание этой команды. Это просто делается с помощью следующей строки кода:

```
byte c = Serial.read ();
```

Если мы получили команду измерения, обозначенную просто asm, мы продолжим работу по скетчу:

```
if (c == 'm'){
```

Первое, что мы делаем после получения команды измерения - это измеряем данные с датчика DHT:

```
float h = dht.readHumidity();
float t = dht.readTemperature();
```

Обратите внимание, что если вы используете другой датчик или два разных датчика, именно здесь вам придется выполнять свои собственные измерения.

Затем мы хотим вернуть некоторые данные в последовательный порт. Для этого проекта мы будем использовать очень простой протокол и возвращать данные в формате температура, влажность. Это делается путем печати этих значений на последовательном порту:

```
Serial.print(t);
Serial.print(",");
Serial.println(h);
```

Это означает, что если Arduino получит соответствующий символ, измерения будут возвращены на компьютер через модуль Bluetooth.

Полный скетч `Arduino` доступен в репозитории GitHub проекта на следующей веб-странице:
`https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter3`

Тестирование датчика температуры и влажности

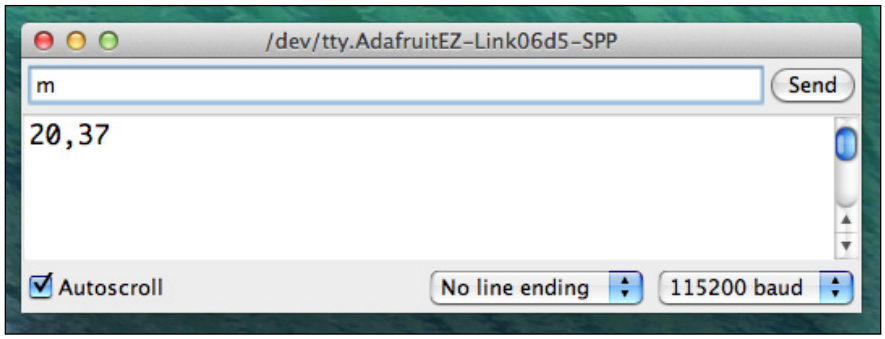
Пришло время скомпилировать эскиз и загрузить его. Для этого мы будем использовать соединение Bluetooth. Выполним следующие шаги для проверки датчика температуры и влажности:

1. После включения вашего проекта (с помощью кабеля USB или внешнего источника питания) вы можете перейти к настройкам Bluetooth в вашей операционной системе. Вы должны увидеть, что модуль Bluetooth был обнаружен. Обратите внимание, что имя этого модуля Bluetooth будет меняться в зависимости от вашего собственного модуля, как показано на следующем снимке экрана:

2. Просто нажмите кнопку сопряжения (или аналогичную кнопку в зависимости от вашей операционной системы), и модуль Bluetooth должен быть подключен к вашему компьютеру.
3. Теперь вернемся к IDE Arduino. Теперь у вас должна быть возможность выбрать модуль Bluetooth в списке последовательных портов. На этом этапе вы можете выбрать тот, который начинается с tty или cu; это не имеет значения для целей тестирования. Обратите внимание, что в зависимости от вашей настройки вы увидите разные имена, но всегда с EZ-Link на них:

4. Теперь вы можете загрузить эскиз на плату Arduino. Просто нажмите «Загрузить» как обычно, и скетч должен быть загружен на плату. Обратите внимание, что по сравнению с обычной передачей скетчей через USB соединение Bluetooth намного медленнее.

5. Теперь вы можете открыть последовательный монитор, чтобы проверить скетч. Опять же, вы заметите некоторую задержку по сравнению с USB-соединением, и для открытия пробного монитора может потребоваться несколько секунд. Теперь просто введите m, и вы должны получить:



Если вы видите, как измерения распечатываются на последовательном мониторе, это означает, что датчик работает правильно. Это также означает, что проект отвечает ожидаемым образом, когда получает сигнал для измерения данных.

Если вы не можете увидеть измерения на этом этапе, первым делом проверьте проводку. Также убедитесь, что модуль Bluetooth правильно сопряжен с вашим компьютером. Обычно вы можете легко увидеть эту информацию, снова проверив панель настроек Bluetooth в вашей операционной системе. Если последовательный интерфейс открывается, но вы не видите, что данные распечатываются после отправки команды измерения, это означает, что проблема могла быть вызвана датчиком DHT. В этом случае убедитесь, что вы правильно подключили датчик и подключили резистор между сигнальным выводом и выводом VCC.

Удаленное измерение температуры и влажности

Теперь, когда основы проекта работают, мы можем двигаться дальше. Мы явно не хотим, чтобы монитор последовательного порта из IDE Arduino постоянно открывался и требовал ввод команды вручную. Вот почему нам нужно создать графический интерфейс на вашем компьютере.

В этом разделе мы собираемся создать простой графический интерфейс, используя Python и прилагаемую к нему графическую библиотеку Tkinter. Весь код будет просто содержаться в файле Python, который можно будет запустить позже, чтобы открыть интерфейс.

Первым шагом кода Python является импорт соответствующих модулей Python. Модули Python в основном похожи на библиотеки Arduino; они добавляют к Python дополнительные функции. Нам нужен модуль времени, последовательный модуль,который мы установили ранее, и все остальные компоненты модуля Tkinter:

```
import time
import serial
from Tkinter import *
```

Если вам нужна дополнительная информация о модуле Tkinter, вы можете посетить страницу официальной документации по адресу https://wiki.python.org/moin/TkInter.
Мы должны настроить последовательное соединение так же, как мы это делали в последовательном мониторе Arduino IDE. Первым шагом является установка последовательной скорости и последовательного порта. Конечно, вам придется изменить последовательный порт в зависимости от того, как он отображается на вашем компьютере:

```
serial_speed = 115200
serial_port = '/dev/cu.AdafruitEZ-Link06d5-SPP'
```

Теперь мы можем создать последовательный порт со следующими параметрами:

```
ser = serial.Serial(serial_port, serial_speed, timeout=1)
```

Следующим шагом будет создание класса, который будет содержать наш графический интерфейс:

```
class Application(Frame):
```

Внутри этого класса будет несколько функций для создания различных элементов интерфейса. Мы не будем вдаваться в подробности каждой отдельной функции, но самая важная из них - это функция отправки порядка измерения на плату:

```
def measure(self):
```

Внутри этой функции нам просто нужно отправить им команду на последовательный порт, чтобы запустить измерение на плате Arduino. Затем мы сразу же считываем данные, которые будут отражены в последовательном порту:

```
ser.write("m")
data = ser.readline()
```

После этого мы должны проверить, верны ли полученные данные. Просто проверяем, что он не пустой:

```
if (data != ""):
```

Если это не так, значит, мы получили сообщение с платы. Помните, что мы будем получать данные в виде температуры, влажности. Вот почему нам нужно использовать функцию Python split () для разделения этих данных и сохранения их в массиве:

```
processed_data = data.split(",")
```

Это означает, что теперь у вас есть массив, в котором измерения температуры хранятся в первой ячейке, а влажность - во второй. Теперь мы можем соответствующим образом обновить интерфейс. Например, поле температуры обновляется с помощью следующего фрагмента кода:

```
self.temp_data.set("Temperature: " + str(processed_data[0]))
self.temperature.pack()
```

То же самое сделано и для отображения влажности. Мы также хотим обновить интерфейс автоматически. Вот почему эта функция повторяется каждую секунду. Это делается с помощью следующего фрагмента кода:

```
self.after(1000,self.measure)
```

Наконец, мы должны запустить приложение в конце файла Python. Это делается с помощью следующего фрагмента кода:

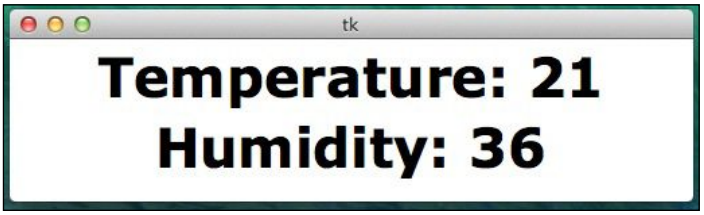
```
root = Tk()
app = Application(master=root)
app.mainloop()
```

Полный код доступен в репозитории GitHub проекта на следующей веб-странице:
https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter3

Теперь вы готовы протестировать проект. Убедитесь, что скетч **Arduino**, который мы разработали ранее, загружен на плату, что плата запитана (через USB или внешний источник питания) и что модуль **Bluetooth** сопряжен с вашим компьютером. Теперь вы можете просто перейти в окно терминала, перейти в папку, в которой находится файл **Python**, и ввести следующее:

```
python interface.py
```

Это должно открыть интерфейс и немедленно провести некоторые измерения с платы Arduino. В зависимости от операционной системы вы должны увидеть на экране следующее:



Если вы не можете увидеть интерфейс на этом этапе, вы можете проверить несколько вещей. Во-первых, убедитесь, что скетч Arduino работает так же, как мы тестировали его в предыдущем разделе. Также убедитесь, что ваш установленый Python работает правильно и что скетч Python не вызывает ошибок. Если вы видите некоторые ошибки, отображаемые внутри вашего терминала, первым делом следует использовать файл Python, который находится в репозитории GitHub, который соответствует книге. Также убедитесь, что установлены все необходимые библиотеки, например, последовательная библиотека Python.

Резюме

В этом проекте мы создали датчик температуры и влажности на базе Bluetooth и Arduino. Используя этот датчик, вы можете измерять данные, поступающие с вашей платы Arduino через Bluetooth, и отображать эти данные на вашем компьютере.

Давайте посмотрим на основные выводы этой главы. В первой части главы мы выбрали компоненты этого проекта, в том числе модуль Bluetooth и датчик температуры и влажности. Также мы скачали и установили необходимые программы проекта.

Затем мы построили аппаратную часть проекта и сделали необходимые подключения на макетной плате. Мы подключили модуль Bluetooth и датчик температуры и влажности к плате Arduino.

Затем мы потратили некоторое время на создание основного скетча проекта. В то же время мы протестировали проект, проверив, действительно ли работает Bluetooth-соединение. Также мы проверили, правильно ли работает датчик температуры и влажности.

Наконец, мы использовали Python для создания простого интерфейса на вашем компьютере. Используя этот интерфейс, мы установили соединение с платой Arduino через Bluetooth и отобразили данные измерений на вашем компьютере.

Также есть несколько способов улучшить проект. Вы можете легко связать Python и базу данных, например MySQL, для автоматического локального протоколирования измерений на вашем компьютере. Вы также можете связать этот проект, все еще использующий Python, с одним из модулей построения графиков Python, чтобы постоянно отображать измеренные значения на графике.

В следующей главе мы снова будем использовать Wi-Fi, но уже совсем для другой цели. На этот раз мы не собираемся измерять и хранить данные локально, а автоматически загружать данные в облачную службу, чтобы они там хранились и отображались.

4

Метеостанция в облаке с Xively

До сих пор мы строили только системы домашней автоматики, к которым можно получить доступ локально. Мы создали датчики движения XBee, управляли освещением через Wi-Fi, а также измеряли данные о температуре и влажности с помощью Bluetooth. В этой главе мы собираемся применить другой подход к построению систем домашней автоматики.

Мы собираемся провести некоторые измерения на вашей плате Arduino, но отправлять эти данные в облачный сервис под названием Xively вместо того, чтобы отправлять эти измерения на локальный сервер. Xively - это онлайн-платформа, на которой вы можете отправлять данные с различных устройств, таких как Arduino. Мы хотим отправлять данные в Xively, чтобы их можно было записать на их серверах, а после этого можно было бы получить к ним доступ из любой точки мира через веб-браузер.

Это будет первый проект в этой книге: подключение домашних автоматов к «Интернету вещей», что на данный момент является основной тенденцией. Интернет вещей - это идея подключения к сети всех устройств, таких как не только компьютеры и смартфоны, но и холодильники, кофемашины и т. д., Чтобы все они могли общаться друг с другом.

Вот основные выводы из этого

- Во-первых, мы собираемся подключить несколько датчиков к вашей плате Arduino, включая датчики температуры и влажности, а также датчик уровня освещенности. Мы также собираемся снова использовать чип CC3000 для подключения к Интернету.
- Затем мы собираемся протестировать различные датчики локально, чтобы убедиться, что оборудование работает правильно, прежде чем делать что-либо, связанное с облаком.

- Следующим шагом является создание учетной записи Xively и ее настройка. Таким образом, вам будут предоставлены разные учетные данные, чтобы плата Arduino знала, куда отправлять данные. Мы также собираемся создать панель инструментов, на которой вы сможете визуализировать записанные данные.

- Наконец, мы собираемся написать окончательный скетч вашей платы Arduino и подключиться к Интернету. Скетч будет автоматически загружать данные в Xivelyat через определенные промежутки времени. Мы также собираемся визуализировать эти данные прямо на веб-сайте Xively и видеть данные, поступающие в режиме реального времени.

Компоненты и программы

Давайте сначала посмотрим, какие компоненты нам понадобятся для этого проекта. Вам понадобится обычная плата Arduino Uno. Другие платы, такие как Arduino Mega, также будут работать, но этот проект не тестировался с более поздними платами, такими как Arduino Leonardo или Arduino Due.

Вам также понадобится плата, на которой установлен чип CC3000 Wi-Fi. Я снова использовал плату Wi-Fi Adafruit CC3000, но вы также можете использовать шилд Arduino, на котором размещается этот чип Wi-Fi, например, Adafruit CC3000.

Другие платы или шилды с таким же чипом Wi-Fi тоже будут работать, но нет никаких гарантий, и вам, возможно, придется немного изменить код проекта.

Для измерения температуры и влажности мы будем использовать цифровой датчик DHT11. Обратите внимание, что вы также можете использовать датчик DHT22, который является более точным и работает с той же библиотекой.

Вы также можете использовать свой собственный датчик для этих измерений или даже два датчика, но тогда вам придется соответственно изменить код. Для работы проекта вам также понадобится резистор 4,7 кОм.

Для измерения уровня освещенности вам понадобится фотоэлемент (который представляет собой просто резистор, который изменяет свое сопротивление в зависимости от окружающего освещения) и резистор 10 кОм. Конечно, вам понадобятся обычные макетные платы и перемычки для соединения между различными составными частями.

Это список компонентов, которые используются в этой главе:

- Arduino Uno R3
- CC3000 Wi-Fi модуль)
- Датчик DHT11 и резистор

- Фотоэлемент
- Макетная плата)
- Перемычки

Что касается программного обеспечения, на вашем компьютере должна быть установлена последняя версия Arduino IDE, а также библиотека DHT для датчика DHT11, который будет использоваться в этом проекте. Вы можете найти эту библиотеку по адресу https://github.com/adafruit/DHT-sensor-library.

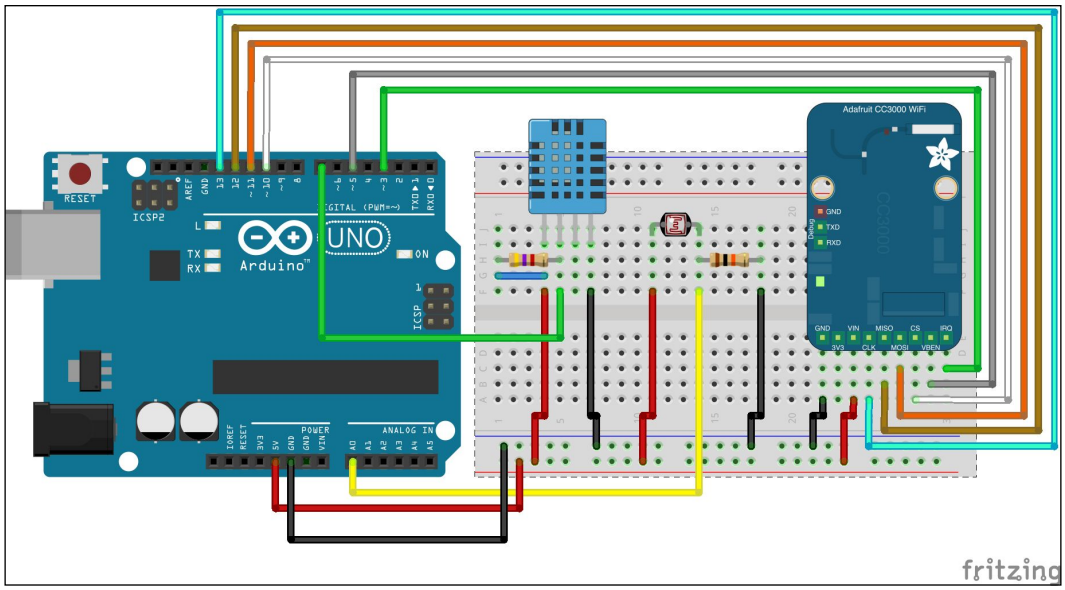
Для этого проекта также требуется библиотека микросхем CC3000. Вы можете найти его на https://github.com/adafruit/Adafruit_CC3000_Library.

Чтобы установить данную библиотеку, распакуйте все файлы в папку с именем REST. Затем поместите эту папку в папку / libraries внутри основной папки Arduino (или создайте эту папку, если ее нет).

Подключение компонентов

В этом проекте вам нужно будет подключить в основном три вещи: чип CC3000Wi-Fi, датчик температуры и влажности и датчик освещенности.

Чтобы точно знать, какие провода и контакты вам нужно соединить, взгляните на следующее изображение, на котором описаны все соединения проекта:



Давайте подключим компоненты, выполнив следующие шаги:

1. Чтобы приступить к подключению устройства, сначала разместите компоненты рядом друг с другом и подключите модуль CC3000 и датчики к макетной плате. Чтобы уменьшить количество необходимых проводов, подключите один вывод фотоэлемента к одному выводу резистора 10 кОм. Кроме того, подключите резистор 4,7 кОм между контактами 1 и 2 датчика DHT11, как показано на предыдущем изображении.

2. Поскольку у нас достаточно много устройств для питания, я предлагаю вам сначала подключить красную шину питания к контакту Arduino 5V, а синюю шину питания - к контакту Arduino GND.

3. Сначала займемся датчиками. Датчик DHT11 имеет три вывода, которые необходимо подключить: два для источника питания и один для сигнала. Подключите вывод № 1 (VCC) к красной шине питания, а вывод № 4 (GND) к синей шине питания. Наконец, подключите вывод 2 (сигнальный контакт) к выводу 7 Arduino.

4. Что касается фотоэлемента, вам необходимо подключить оставшиеся выводы фотоэлемента к красной шине питания, а оставшиеся выводы резистора - к синей шине питания. Наконец, подключите общий контакт к контакту A0 Arduino.

5. Теперь пора поработать с чипом CC3000. Для этого модуля необходимо подключить несколько пинов, поэтому внимательно следуйте инструкциям, иначе модуль может работать некорректно. Подключите пин IRQ платы CC3000 к пину № 3 платы Arduino, VBAT к пину 5 и CS к пину 10. Затем вам нужно подключить пины SPI к плате Arduino: MOSI, MISO и CLK подключить к пинам 11, 12 и 13 соответственно. P.S. Пины - это то же самое что и контакты.

6. Теперь вы можете подключить плату к компьютеру через USB, и вы готовы написать свой первый скетч для этого проекта.

Тестирование датчиков

Перед отправкой данных с наших датчиков в облако мы сначала убедимся, что эти датчики работают правильно. Для этого мы сделаем обычный тестовый скетч, который попытается прочитать данные с датчиков и распечатать эти данные на последовательном мониторе.

Теперь перейдем к IDE Arduino. Скетч Arduino начинается с импорта соответствующей библиотеки для датчика DHT. Библиотека выглядит следующим образом:

```
#include "DHT.h"
```

Мы также должны объявить переменные для хранения измерений:

```
int lightLevel;
float humidity;
float temperature;
```

И чтобы определить, к какому выводу подключен датчик DHT, используйте следующий код:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

Мы также должны создать пример для датчика DHT:

```
DHT dht(DHTPIN, DHTTYPE);
```

In the `setup()` function of the sketch, we need to initialize the DHT sensor:

```
dht.begin();
```

Затем запустите последовательное соединение:

```
Serial.begin(115200);
```

В функции скетча loop () мы производим измерения:

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int lightLevel = analogRead(A0);
```

И распечатаем эти измерения в последовательном порту:

```
Serial.print("Temperature: ");
Serial.println(temperature);
Serial.print("Humidity: ");
Serial.println(humidity);
Serial.print("Light level: ");
Serial.println(lightLevel);
Serial.println("");
```

Это повторяется каждые 500 миллисекунд:

```
delay(500);
```

Полный код можно найти в репозитории проекта GitHub по следующему URL-адресу:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter4

Теперь вы можете загрузить код на плату Arduino и открыть монитор последовательного порта. Убедитесь, что скорость последовательного порта установлена на 115200 бод. Вы должны видеть, что данные измерений печатаются каждые 500 миллисекунд:

```
Temperature: 23.00
Humidity: 40.00
Light level: 728
```

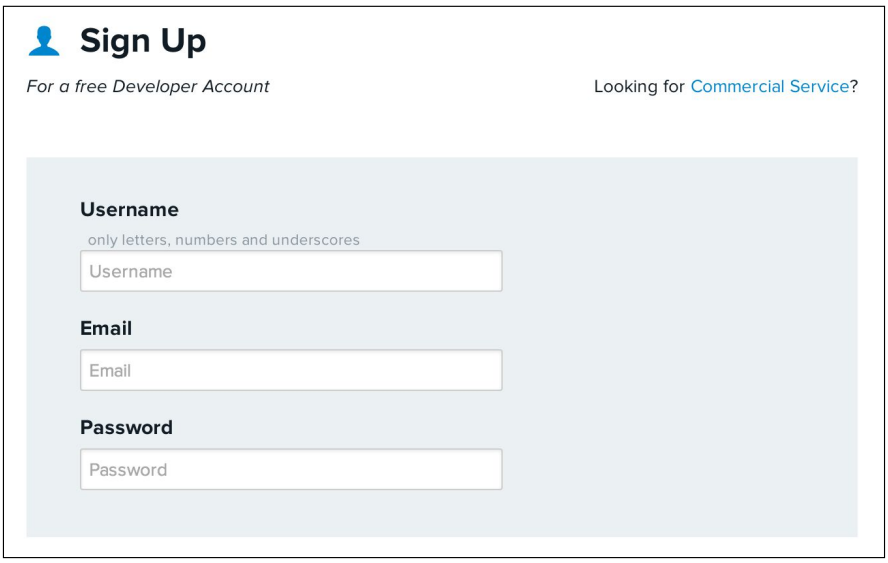
Если это не так, вам нужно будет снова проверить аппаратные соединения. Снова выполните все соединения и убедитесь, что они похожи на соединения в схемах.

Настройка учетной записи Xively

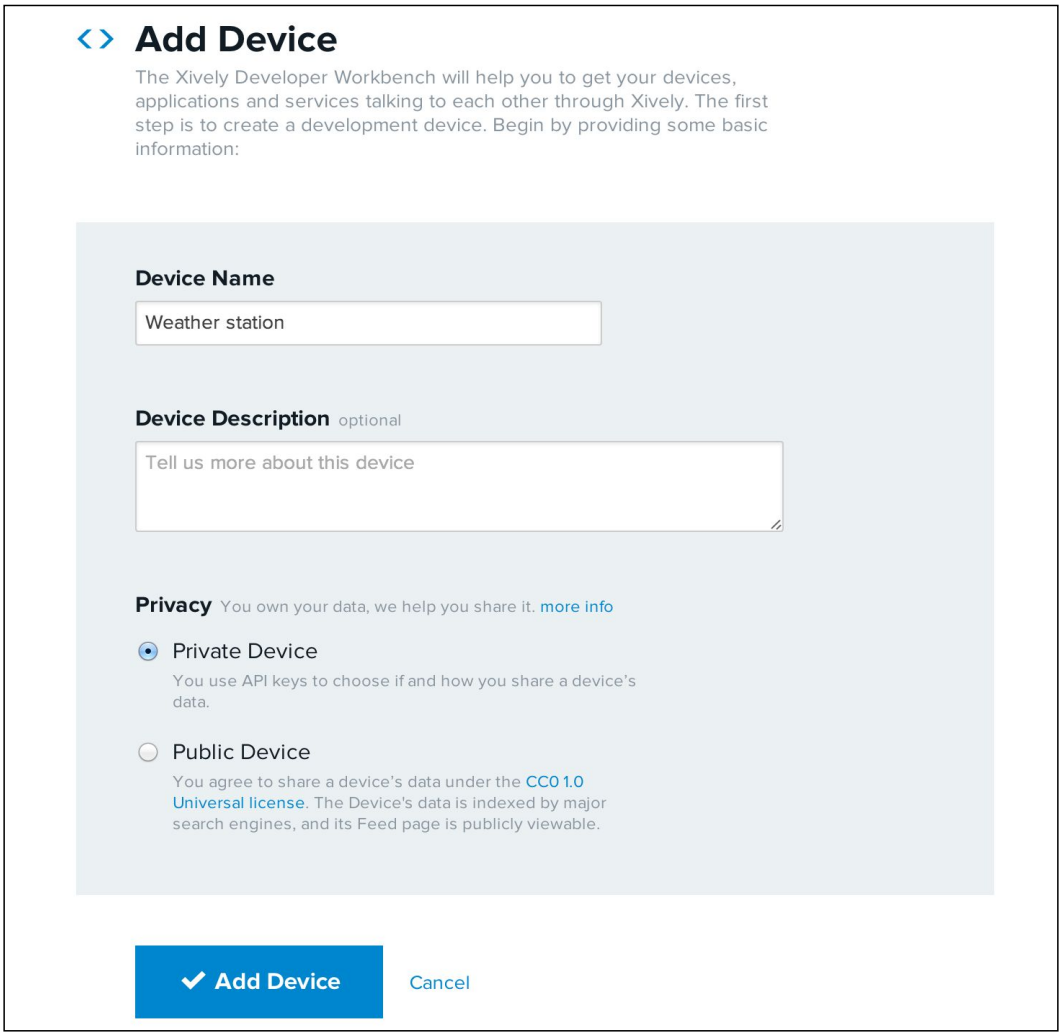
Давайте теперь настроим вашу учетную запись Xively. Мы собираемся создать учетную запись, а затем настроить ее так, чтобы вы могли загружать данные в Xively. В конце концов, у вас будет ключ API и идентификатор фида, который вам нужно будет ввести в Arduino позже. Эти два параметра важны для бесперебойной работы проекта. Ключ API используется для аутентификации на сервере Xively, когда мы делаем запрос на сохранение данных с платы Arduino. Что касается идентификатора фида, он будет использоваться Xively, чтобы знать, на каком устройстве должны храниться данные.

Начнем с создания учетной записи. Перейдите на главную страницу регистрации Xively по адресу https://xively.com/signup/.

Введите свои данные на странице и завершите процесс регистрации:

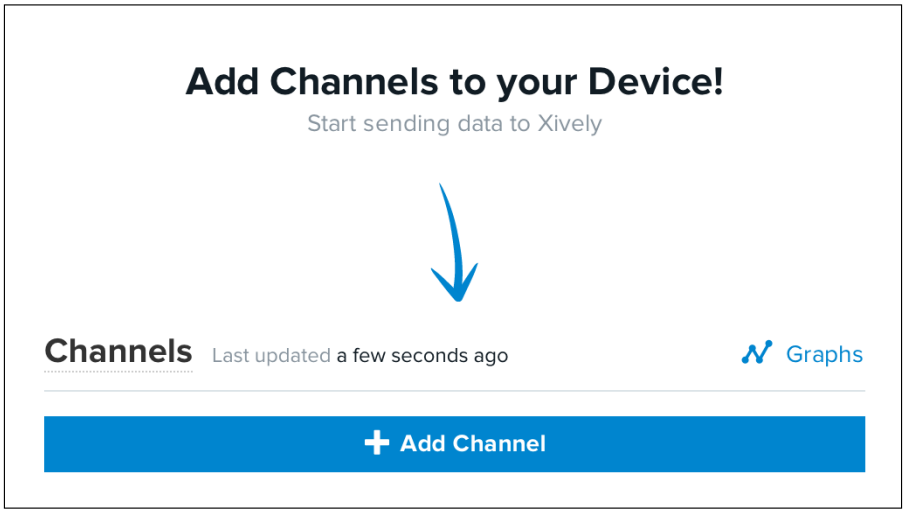


Затем, как только вы войдете в систему, вам нужно будет создать устройство. Для этого перейдите на вкладку «Develop»(Разработка) и нажмите « Add Device» (Добавить устройство). Вы попадете на следующий экран:

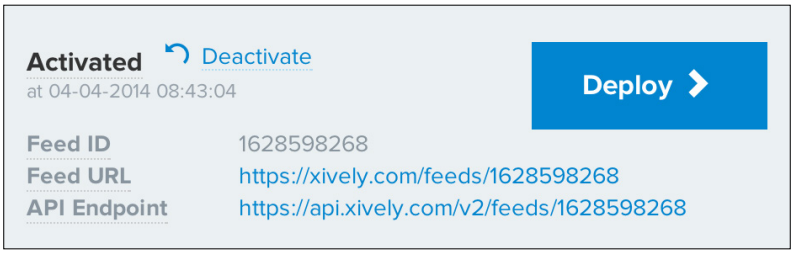


На этой странице вставьте имя своего устройства, краткое описание и установите устройство как частное, чтобы только вы могли получить к нему доступ. Завершите, нажав на Add Device (Добавить устройство)

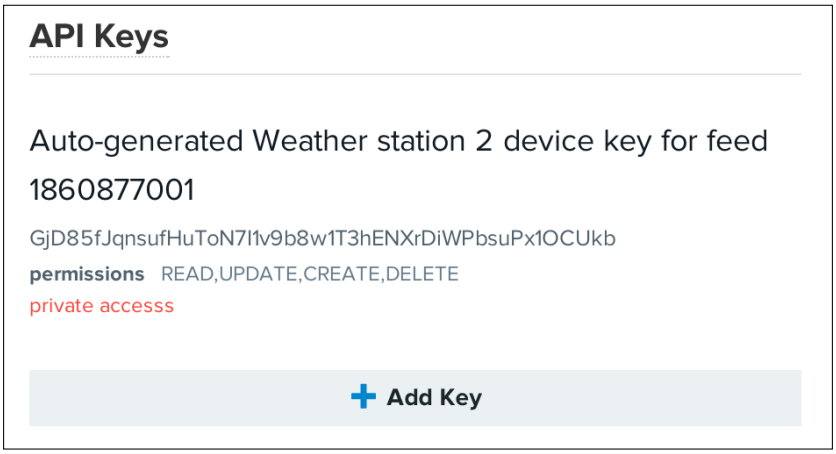
Затем вы сможете выбрать его из списка всех устройств, которые есть у вас в учетной записи. На странице устройства нам теперь нужно создать каналы для нашего устройства, которые будут получать данные от нашего датчика. Давайте создадим первый, щелкнув Add channe (Добавить канал):



Затем вы можете ввести данные о своем канале. Самый важный аспект - это название канала. Например, в этом проекте я назвал каналы Temperature(Температура), Humidity (Влажность) и Light(Свет). Просто введите имя каждого канала и укажите, нужна ли вам единица записанных данных (но это необязательно).
Теперь вам также понадобится некоторая информация о вашем устройстве и учетная запись для скетча Arduino. Сначала вам нужно получить идентификатор фида вверху страницы:



Как только это будет сделано, мы можем перейти к функции setup () скетча. Начнем с инициализации микросхемы CC3000:



Также запишите этот ключ API где-нибудь на бумаге; он вам понадобится позже.

Создание скетча Arduino

На данный момент у нас есть работающие датчики на нашей плате Arduino, а также у нас есть учетная запись на Xively, которая готова принимать некоторые данные. Теперь мы собираемся написать скетч, который будет подключаться к вашей сети Wi-Fi и автоматически загружать данные в вашу учетную запись Xively.

Теперь мы собираемся записать код в IDE Arduino. Начнем с включения соответствующих библиотек, как показано:

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "DHT.h"
```

Затем, как и в главе 2 «Управляйте светом с телефона или планшета», нам нужно определить, к каким пинам подключена плата CC3000:

```
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
```

И определите пин и тип датчика DHT:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

Теперь мы можем описать параметры чипа CC3000:

```
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
  ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);
```

Теперь вам нужно будет изменить несколько вещей в скетче. Первая пара вещей - это SSID и пароль вашей сети Wi-Fi:

```
#define WLAN_SSID       "yourSSID"
#define WLAN_PASS       "yourPassword"
```

Теперь вам нужно ввести некоторую информацию о Xively, которую мы получили ранее в этой главе. Измените эти строки, чтобы использовать собственный ключ API и идентификатор ID фида:

```
#define WEBSITE  "api.xively.com"
#define API_key
  "GFSM8c7f83E0MMta3FkGAwzVktkzXNN6kXhoRBV8QN1WmkiZ"
#define feedID  "1628598268"
```

Как только это будет сделано, мы можем перейти к функции setup () скетча. Начнем с инициализации микросхемы CC3000:

```
Serial.println(F("\nInitializing..."));
if (!cc3000.begin())
{
  Serial.println(F("Couldn't begin()! Check your wiring?"));
  while(1);
}
```

Затем в функции скетча loop () мы начинаем с подключения к вашей локальной сети Wi-Fi:

```
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
```

Мы также определяем IP-адрес веб-сайта Xively, чтобы чип CC3000 знал, куда он должен подключиться:

```
uint32_t ip = cc3000.IP2U32(216,52,233,120);
```

Пришло время провести измерения. Это похоже на то, что мы видели в первом скетче, где мы тестировали различные датчики:

```
float temperature = dht.readTemperature();
float humidity = dht.readHumidity();
int lightLevel = analogRead(A0);
```

А теперь самое сложное. Чтобы загружать данные в Xively, нам нужно следовать заданному формату, чтобы серверы Xively понимали, что мы хотим отправить. Все это определено в документации Xively API, которую вы можете найти на следующем веб-сайте Xively:

```
https://xively.com/dev/docs/api/
```

В нашем случае нам нужно отформатировать данные в формате JSON, включив все измерения в поле, называемое потоками данных. Это результат внутри скетча Arduino, где мы помещаем все в строковую переменную с именем data:

```
String data = "";
data = data + "\n" + "{\"version\":\"1.0.0\",\"datastreams\" :
  [ {\"id\" : \"Temperature\",\"current_value\" : \"" +
  String((int)temperature) + "\"}," + "{\"id\" :
  \"Light\",\"current_value\" : \"" + String(lightLevel) + "\"},"
 + "{\"id\" : \"Humidity\",\"current_value\" : \"" +
  String((int)humidity) + "\"}]}";
length = data.length();
```

Из предыдущего кода мы также получаем длину строки данных, которая будет использоваться при передаче данных на серверы Xively. Затем мы можем подключиться к серверу Xively:

```
Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
```

Когда скетч подключен к Xively, мы можем отправлять данные. Это будет сделано путем выполнения нескольких функций print () с использованием данных клиента. Для этого мы будем использовать запрос PUT для ID фида, который мы определили ранее, чтобы сервер знал, что мы хотим отправить данные JSON. Вот результат с точки зрения кода внутри скетча Arduino:

```
client.println("PUT /v2/feeds/" + String(feedID) + ".json HTTP/1.1");
client.println("Host: api.xively.com");
client.println("X-ApiKey: " + String(API_key));
client.println("Content-Length: " + String(length));
client.print("Connection: close");
client.println();
client.print(data);
client.println();
```

После отправки данных нам нужно прочитать ответ, исходящий от сервера Xively, прежде чем мы сможем продолжить:

```
while (client.available()) {
  char c = client.read();
  Serial.print(c);
}
```

Наконец, в целях экономии энергии мы каждый раз отключаемся от локальной сети Wi-Fi:

```
cc3000.disconnect();
```

И повторяем процесс каждые 10 секунд:

```
delay(10000);
```

Полный код можно найти в репозитории проекта GitHub по следующему URL-адресу:

`https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter4`

Авторизуйтесь и отображайте данные на Xively

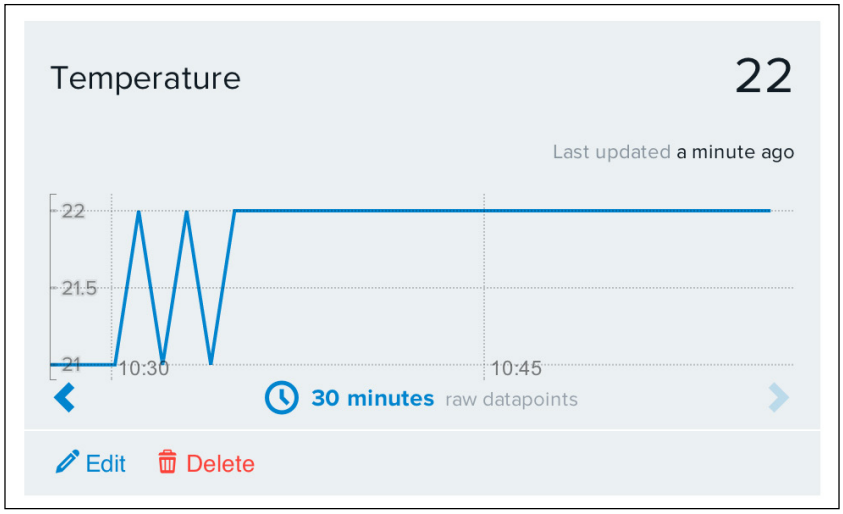
Пришло время протестировать проект. Вы можете загрузить код на плату Arduino и открыть последовательный монитор. Вы должны увидеть, что скетч подключен к Интернету и загружает данные в Xively. Вы также должны увидеть ответ сервера на последовательном мониторе, как показано ниже:

```
HTTP/1.1 200 OK
Date: Wed, 07 May 2014 08:48:05 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 0
Connection: close
X-Request-Id: aedcb75d10ea1556813941846bdaded812904bb4
Cache-Control: max-age=0
Vary: Accept-Encoding
```

Если вы видите этот код HTTP / 1.1 200 OK, это означает, что передача прошла успешно. Если это не так, убедитесь, что у вас действительно тот же скетч, что и в репозитории GitHub, и что вы правильно ввели свою информацию (ключи Wi-Fi и Xively). Также убедитесь, что чип CC3000 правильно подключен.

Давайте теперь проверим страницу устройства на Xively. Снова перейдите на страницу устройства, и вы должны увидеть следующее:

Нажав на кнопку «Graphs» (Графики), вы также сможете увидеть графики измеряемых данных. Например, для температуры вы видите графики данных, показанные на следующем снимке экрана:



Приятно то, что данные постоянно измеряются, отправляются и сохраняются в сервисе Xively. Это не зависит от того, включен или выключен ваш собственный компьютер, есть дополнительный уровень безопасности и удобства по сравнению с тем, что ваш собственный сервер записывает данные у вас дома.

Резюме

Подведем итог тому, что мы сделали в этом проекте. Мы построили простую метеостанцию на базе Arduino, которая может автоматически загружать данные в Интернет с помощью веб-сервиса Xively. По сравнению с подходом, который мы использовали в предыдущих главах, этот подход позволяет создавать автономные системы, которые автоматически отправляют данные в удаленную службу, которая хранит данные в облаке. Это означает, что ваши данные находятся в безопасности в облаке и могут быть доступны из любой точки мира; вам просто нужно подключение к Интернету, а также имя и пароль вашей учетной записи Xively.

Посмотрим основные выводы из этой главы. Сначала мы построили наш проект вокруг датчика температуры и влажности, датчика уровня освещенности и чипа CC3000 Wi-Fi. Мы собрали все вокруг платы Arduino и макета для соединений. Затем мы написали простой скетч Arduino, чтобы убедиться, что мы правильно подключили оборудование.

.После этого мы зашли в наш веб-браузер и создали учетную запись в веб-сервисе Xively. Внутри учетной записи мы создаем новое устройство для размещения измерений, поступающих из нашего проекта, а также набор каналов, которые будут получать отдельные измерения, например, температуру или уровень освещенности. У нас также есть идентификатор канала и ключ API, которые необходимы нашей плате Arduino, чтобы знать, куда отправлять данные.

Наконец, мы написали финальный скетч Arduino для загрузки данных в Xively. Мы протестировали скетч, чтобы убедиться, что Xively правильно получает данные, а также проверили на панели управления устройства на Xively, что данные были получены правильно. В следующей главе книги мы собираемся взять те же принципы и применить их к другой области домашней автоматики: мониторингу энергии. Мы собираемся создать датчик тока и мощности, который будет автоматически отправлять данные о потреблении энергии у нас в доме в Интернет, чтобы вы могли отслеживать потребление энергии в Интернете.

5

Следите за потреблением энергии в облаке

В этой главе мы будем опираться на то, что мы узнали в главе 4, Метеостанция в облаке с Xively, и на этот раз мы отправим данные о потреблении энергии в облако. Мы собираемся подключить датчик тока к Arduino и подключить устройство, подобное лампе, к проекту, чтобы вы могли непрерывно измерять ток и мощность, производимые этим устройством. Затем, используя чип CC3000 Wi-Fi, мы собираемся отправить эти данные в Xively, чтобы их можно было отслеживать в Интернете.

Вот основные моменты, которые мы увидим в этой главе:

- Сначала мы подключим различные компоненты к плате Arduino. Это будет состоять из подключения датчика тока к плате Arduino, а также устройства, с которого мы хотим измерить ток. Мы также подключим к проекту CC3000 Wi-Fi, чтобы он мог отправлять данные в Xively.
- Затем мы напишем простой эскиз для локального измерения силы тока. Это подтвердит, что аппаратные соединения проекта были выполнены правильно.
- Мы также настроим учетную запись Xively так, чтобы она могла принимать измерения, полученные в результате этого проекта.
- Наконец, мы напишем окончательный скетч, который отправит данные в Xively. Кроме того, мы проверим сервер Xively, чтобы убедиться, что мы действительно можем отслеживать потребление энергии устройством в реальном времени.

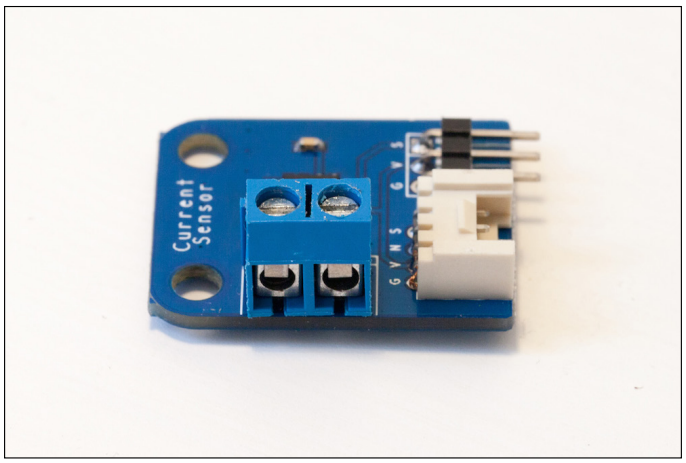
Компоненты и программы

Во-первых, давайте посмотрим на компоненты, которые нам понадобятся в этом проекте. Как и в предыдущей главе, вам понадобится плата Arduino Uno и плата с CC3000 Wi-Fichip. Подробнее о том, как выбрать эти две платы, читайте в предыдущей главе.

Тогда вам понадобится датчик тока. Я использовал коммутационную плату ITead Studio, на которой размещен датчик тока AC712. Этот датчик представляет собой аналоговый датчик, который возвращает сигнал, пропорциональный измеренному току. Позже в этой главе мы узнаем, как рассчитать измеренный ток по выходному напряжению. Вы можете найти более подробную информацию о том, как этот датчик работает в техническом описании датчика на

https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf.

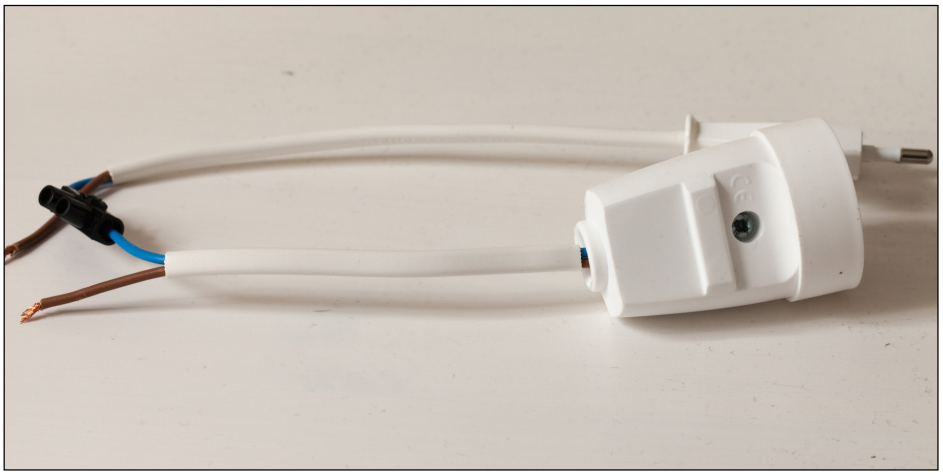
Ниже показано изображение платы, которую я использовал:



Вы можете видеть на предыдущем изображении, что на плате есть три контакта: G для заземления, V для VCC и S для сигнала, которые выдают выходное напряжение, пропорциональное измеренному току.

Вы должны иметь возможность использовать других поставщиков этого датчика, например Adafruit, SparkFun или SeeedStudio. Просто убедитесь, что используемая вами сенсорная плата совместима с уровнем напряжения 5 В на плате Arduino.

Вам также понадобятся два кабеля питания со стандартными вилками (один мужской и один женский), чтобы подключить устройство к вашему проекту и к розетке.
Конечно, вы должны использовать вилку питания в соответствии со стандартами страны, в которой вы живете. Для этого проекта я использовал стандартные европейские вилки на 230 В.
Кабель питания на следующем изображении - это тот же кабель, который я использовал в главе 2, Управляйте светом с телефона или планшета:



Для создания тестов с помощью проекта вам также понадобится устройство для тестирования. Я провел тест с обычной настольной лампой мощностью 30 Вт, но вы можете выбрать любое устройство переменного тока по своему усмотрению.

Датчик тока предназначен для измерения больших токов, поэтому не берите маломощное устройство переменного тока, иначе показания тока могут быть неверными.
Однако на некоторых платах, на которых установлен датчик тока ACS712,у вас есть возможность изменить чувствительность измерения тока. Таким образом, вы можете измерить более низкие токи и мощность в зависимости от используемой вами платы.

Ниже приведены требования к компонентам:

- Arduino Uno
- Коммутационная плата Adafruit CC3000
- Модуль ACS712
- Макетная плата
- Перемычки

Что касается программного обеспечения, на вашем компьютере должна быть установлена последняя версия `Arduino IDE`, а также библиотека микросхем `CC3000`:

```
https://github.com/adafruit/Adafruit_CC3000_Library
```

Чтобы установить данную библиотеку, распакуйте все файлы в папку с именем REST. Затем поместите эту папку в папку / libraries внутри основной папки Arduino (или создайте эту папку, если ее нет).
Вам также понадобится учетная запись на Xively. Пожалуйста, обратитесь к предыдущей главе для получения информации о том, как создать там учетную запись.

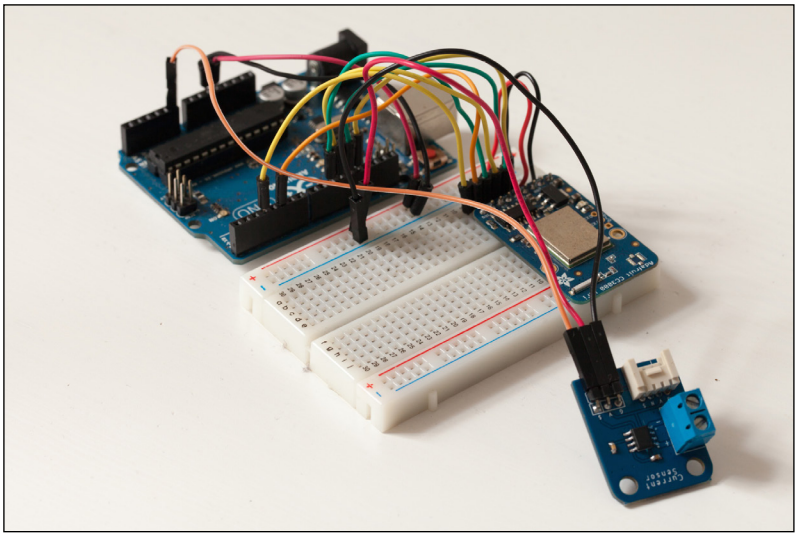
Выполнение аппаратных подключений

Пришло время установить аппаратные соединения проекта. Вам необходимо подключить три вещи: чип CC3000 Wi-Fi к плате Arduino, датчик тока к плате Arduino и кабели питания к датчику тока.

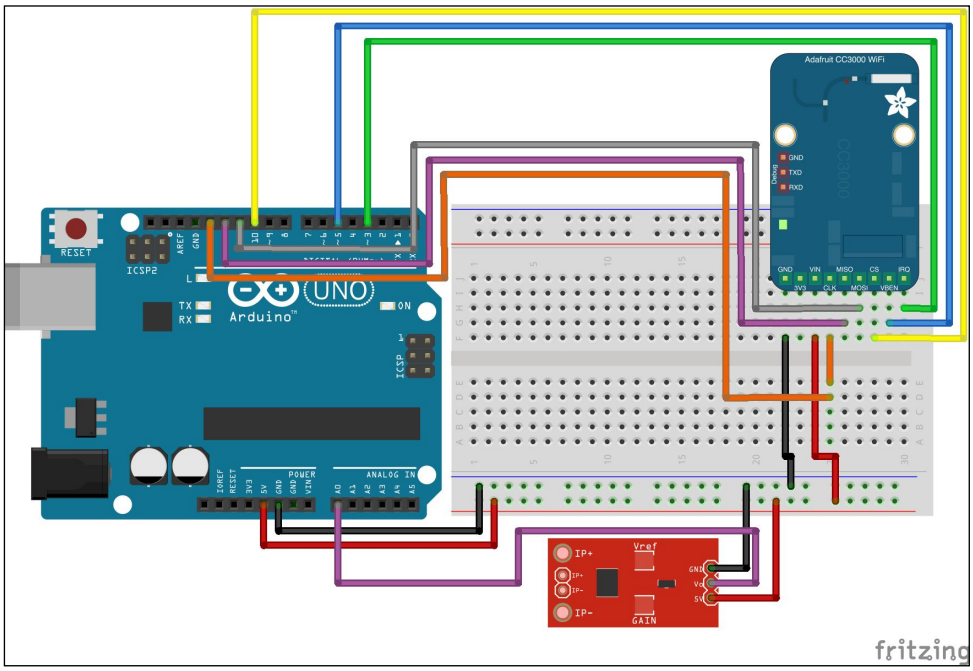
Поскольку у нас есть несколько устройств для питания, я предлагаю вам сначала подключить красную шину питания к контакту Arduino 5V, а синюю шину питания - к контакту Arduino GND.

Для CC3000 достаточно нескольких контактов; поэтому внимательно следуйте инструкциям, чтобы модуль мог работать правильно. Сначала подключите контакт IRQ платы CC3000 к контакту № 3 платы Arduino, VBAT - к контакту 5 Arduino, а CS - к контакту 10 Arduino. После этого вам необходимо подключить оставшиеся контакты SPI к плате Arduino. Контакты MOSI, MISO и CLK идут к контактам 11, 12 и 13 Arduino именно в этом порядке.

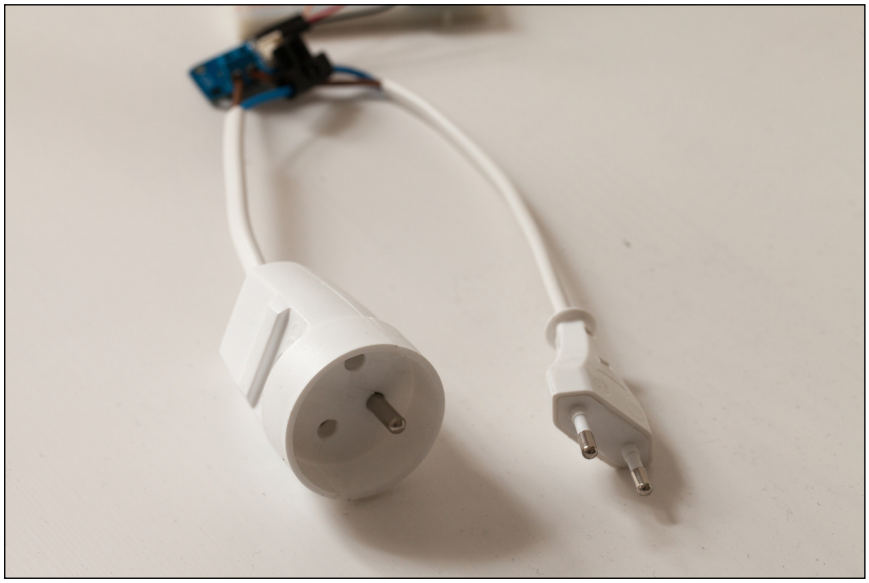
Датчик тока подключить довольно просто. Подключите контакт V к красной шине питания, контакт G к синей шине питания, а контакт S к аналоговому контакту A0. На следующем изображении показано, как система должна выглядеть в этот момент:



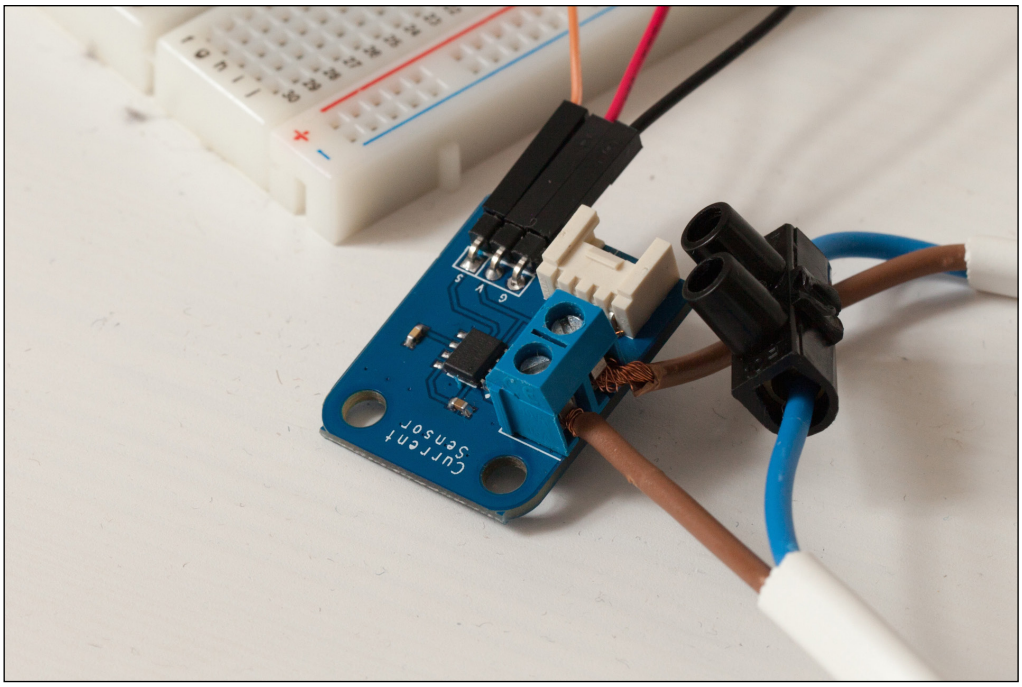
Следующий рисунок показывает порядок соединения компонентов:



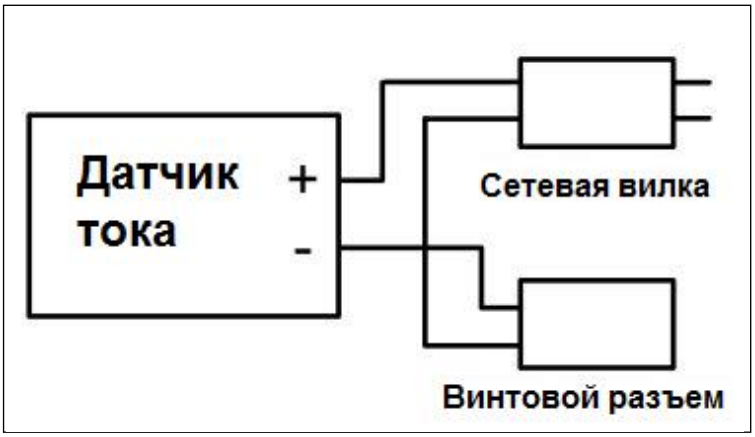
Теперь вы можете подключить силовые кабели к датчику тока, чтобы подключить устройство к системе:



Порядок соединении силовых элементов



Идея состоит в том, чтобы два провода силовых кабелей были соединены вместе (через черный соединитель, как показано на предыдущем рисунке), а два других провода были подключены к плате датчика тока. Таким образом, датчик тока будет определять ток, идущий по проводам. На следующей схеме показаны различные соединения в этой части:



Тестирование проекта

Теперь мы можем написать простой тест для проекта, просто чтобы проверить правильность подключения оборудования. Мы собираемся написать код внутри Arduino IDE. Обратите внимание, что в этом разделе объясняется код для тестирования проекта, и вы можете найти полный код в репозитории проекта на GitHub.

Код начинается с определения вывода датчика тока:

```
#define CURRENT_SENSOR A0
```

Затем мы определяем различные переменные, которые необходимы для теста. Здесь мы собираемся измерить ток и мощность. Обратите внимание, что мы выставляем сетевое напряжение устройства в зависимости от вашего региона:

```
float amplitude_current;
float effective_value;
float effective_voltage = 230; // Установите напряжение 230 В (Европа) или
110 В (США).
float effective_power;
float zero_sensor;
```

Затем в функции скетча setup () нам нужно получить нулевое значение датчика. Поскольку у нас есть аналоговый датчик, нам нужно знать, что датчик считывает при отсутствии тока:

```
zero_sensor = getSensorValue();
Serial.print("Zero point sensor: ");
Serial.println(zero_sensor);
Serial.println("");
```

Давайте теперь подробно рассмотрим эту функцию, которая измеряет значение с датчика. Поскольку это аналоговый датчик, нам нужно усреднить показания по нескольким измерениям, чтобы получить стабильное измерение. Для этого проекта значение усредняется по 100 измерениям. Это делается с помощью следующей функции:

```
float getSensorValue()
{
  int sensorValue;
  float avgSensor = 0;
  int nb_measurements = 100;
  for (int i = 0; i < nb_measurements; i++) {
    sensorValue = analogRead(CURRENT_SENSOR);
    avgSensor = avgSensor + float(sensorValue);
  }
  avgSensor = avgSensor/float(nb_measurements);
  return avgSensor;
}
```

Внутри функции loop () мы также измеряем значение, поступающее от датчика, используя ту же функцию:

```
float sensor_value = getSensorValue();
Serial.print("Sensor value: ");
Serial.println(sensor_value);
```

Затем мы можем вычислить значение действующего тока, которое указано в таблице данных датчика тока ACS712:

```
amplitude_current=(float)(
  sensor_value-zero_sensor)/1024*5/185*1000000;
effective_value=amplitude_current/1.414;
```

После этого печатаем значение тока, действующий ток, а также вычисляем эффективную мощность:

```
Serial.println("Current amplitude (in mA): ");
Serial.println(amplitude_current,1);
Serial.println("Current effective value (in mA)");
Serial.println(effective_value,1);
Serial.println("Effective power (in W): ");
Serial.println(abs(effective_value*effective_voltage/1000),1);
Serial.println("");
```

Мы повторяем это каждые 500 мс:

```
delay(500);
```

Код для этой части можно найти в репозитории GitHub книги на следующей веб-странице:
https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter 5

Теперь вы можете протестировать эскиз. Убедитесь, что устройство, подключенное к проекту, полностью выключено, чтобы начальное измерение нулевого тока могло быть правильно выполнено платой Arduino. Загрузите код на плату Arduino и откройте монитор последовательного порта. Вот что вы должны увидеть:

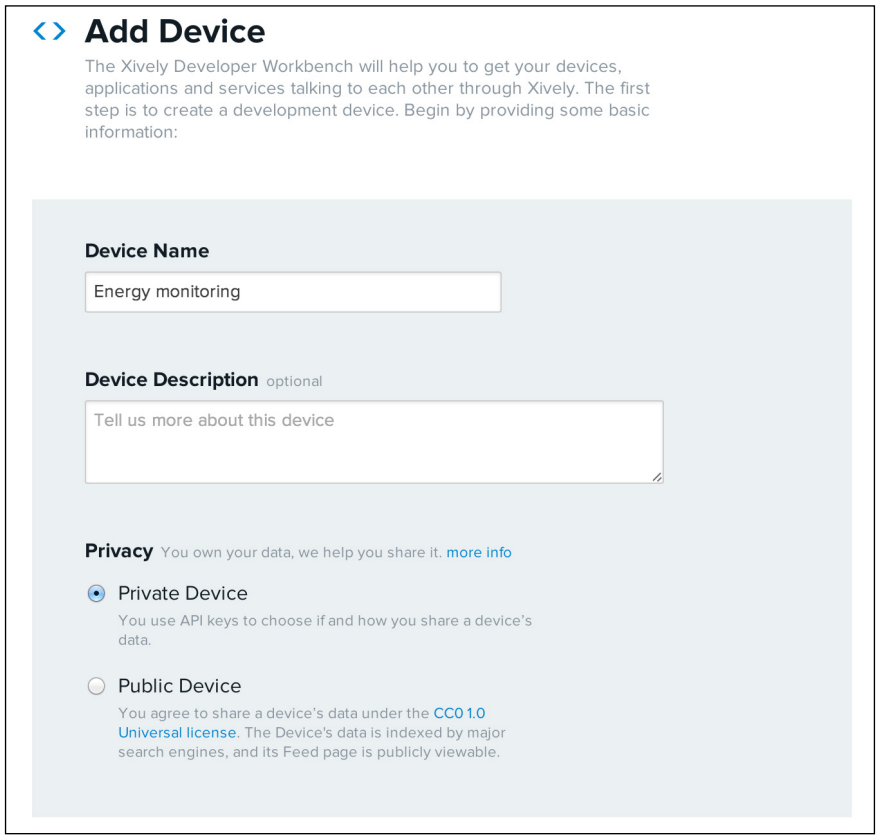
```
Current amplitude (in mA):
0.8
Current effectivate value (in mA)
0.6
Effective power (in W):
0.1
```

Вы также можете попробовать подключить устройство к вашему проекту сейчас, включить его и снова запустить скетч. Вы должны увидеть мгновенное изменение измерения мощности:

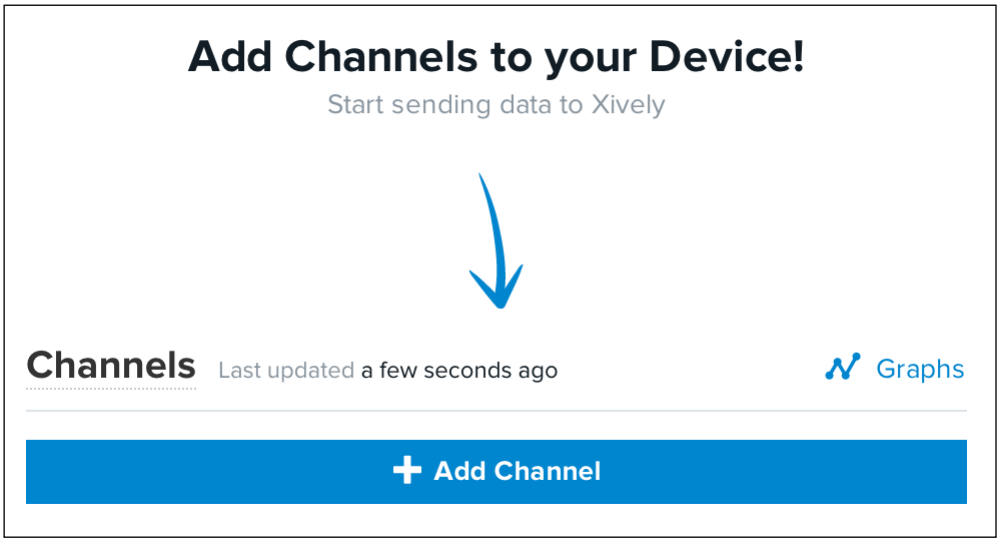
```
Current amplitude (in mA):
17.8
Current effectivate value (in mA)
12.6
Effective power (in W):
28.9
```

Настройка учетной записи Xively

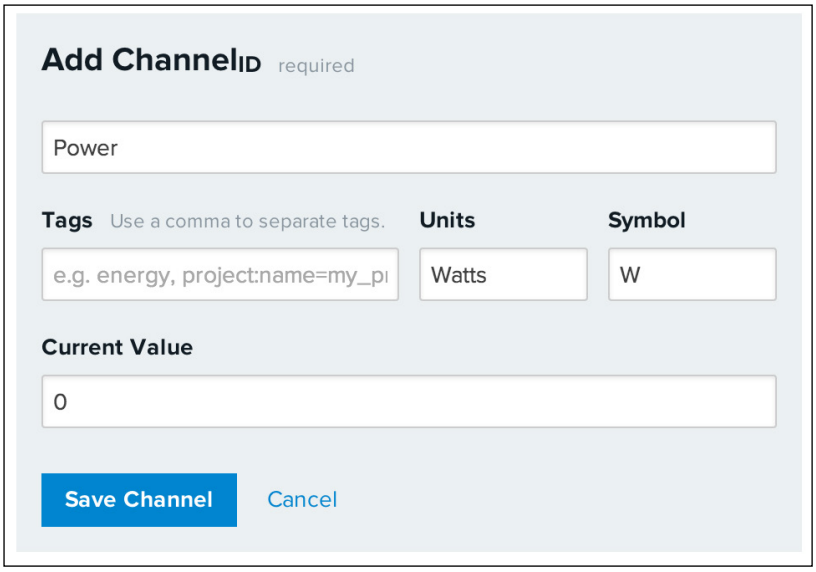
Теперь давайте настроим вашу учетную запись Xively для этого проекта. На этом этапе у вас уже должна быть учетная запись. Если нет, создайте его, следуя процедуре, описанной в предыдущей главе. Он начинается с создания нового устройства, как показано на следующем снимке экрана:



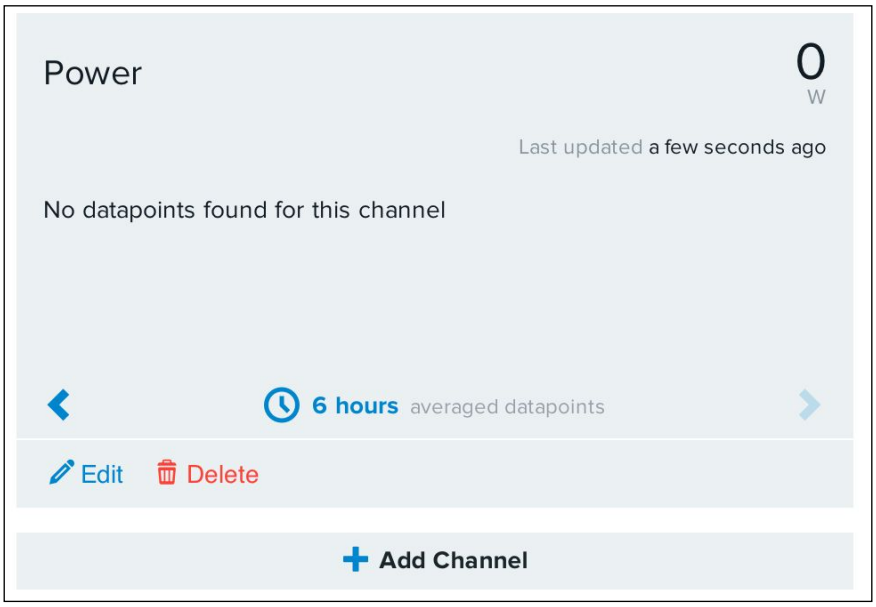
Затем внутри этого нового устройства вам нужно будет создать два новых канала: один для тока и один для питания. Я просто называю мощность и ток:



Например, для канала мощности вы даже можете установить правильную единицу измерения (ватты), как показано на следующем снимке экрана:



Теперь вы должны увидеть канал питания, отображаемый на странице:



Сделайте то же самое для текущего канала, и все готово. Просто запишите идентификатор устройства и ключ API устройства в блокноте на своем компьютере или где-нибудь на бумаге; они вам скоро понадобятся.

Отправка данных о потреблении энергии в Xively

Давайте теперь напишем скетч, который загрузит данные измерения мощности в Xively. Здесь речь идет о тестовом скетче, поэтому я подробно расскажу только о новых частях. Теперь мы рассмотрим наиболее важные части кода в этом разделе. Чтобы получить полный код, обратитесь к репозиторию GitHub этой главы.

Он начинается с импорта соответствующих библиотек:

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
```

Затем нам нужно описать выводы микросхемы CC3000:

```
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10
```

После этого нам нужно объявить CC3000:

```
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS,
  ADAFRUIT_CC3000_IRQ, ADAFRUIT_CC3000_VBAT, SPI_CLOCK_DIV2);
```

Теперь это та часть, где вам нужно ввести свою информацию. Сначала вы вводите SSID и пароль своей локальной сети Wi-Fi:

```
#define WLAN_SSID       "yourSSID"
#define WLAN_PASS       "yourPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Теперь нам нужно ввести ключ API и идентификатор канала вашего устройства Xively:

```
#define API_key  "yourAPIKey"
#define feedID  "yourFeedID"
```

В части скетча setup () нам нужно запустить чип CC3000:

```
if (!cc3000.begin())
{
  Serial.println(F("Couldn't begin()! Check your wiring?"));
  while(1);
}
```

В функции loop () скетча нам нужно подключить чип CC3000 к вашей сети:

```
cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
```

Затем мы определяем IP-адрес сервера Xively:

```
uint32_t ip = cc3000.IP2U32(216,52,233,120);
```

Затем нам нужно отформатировать данные. Все это определено в документации Xively; при этом вы должны форматировать данные внутри контейнера JSON:

```
int length = 0;
String data = "";
data = data + "\n" + "{\"version\":\"1.0.0\",\"datastreams\" : [
  {\"id\" : \"Current\",\"current_value\" : \"" +
  String((int)effective_value) + "\"}," + "{\"id\" :
  \"Power\",\"current_value\" : \"" + String((int)effective_power)
  + "\"}]}";
Serial.println(data);
length = data.length();
```

Как только у нас есть данные, отформатированные в строку, мы можем отправить их на сервер Xively вместе с ключом API:

```
Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
if (client.connected()) {
  Serial.println("Connected!");
  client.println("PUT /v2/feeds/" + String(feedID) + ".json
    HTTP/1.1");
  client.println("Host: api.xively.com");
  client.println("X-ApiKey: " + String(API_key));
  client.println("Content-Length: " + String(length));
  client.print("Connection: close");
  client.println();
  client.print(data);
  client.println();
}
```

Как только у нас есть данные, мы можем отправить на сервер Xively с ключом API:

```
while (client.connected()) {
  while (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
}
```

Затем отключаем чип от локальной сети:

```
cc3000.disconnect();
```

Повторяем процедуру каждые 10 секунд:

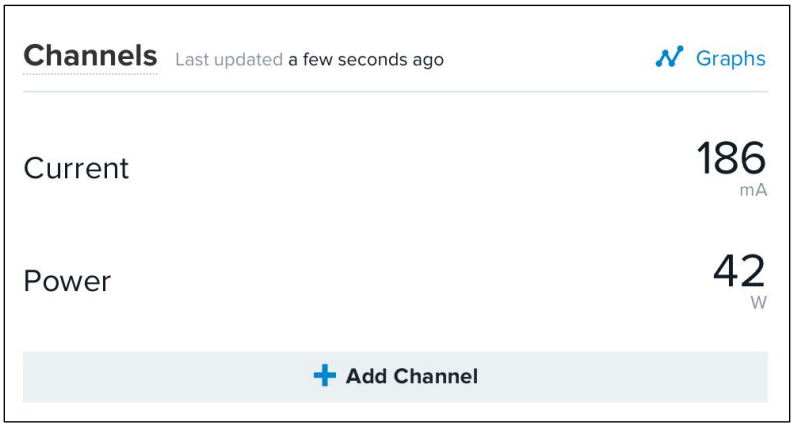
```
delay(10000);
```

Пришло время протестировать скетч

Код этой части можно найти в репозитории GitHub книги на следующей веб-странице:
https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter5

Вы можете загрузить скетч на плату Arduino и открыть последовательный монитор. Вы должны увидеть, что скетч подключается к сети Wi-Fi, отправляет данные в Xively и получает подтверждающее сообщение с сервера.

Вы также можете проверить правильность приема данных на странице устройства на Xively, как показано на следующем снимке экрана:



Резюме

В этой Глава мы построили устройство для мониторинга электроэнергии с подключением к облаку на базе Arduino. Мы подключили датчик тока и чип Wi-Fi к Arduino и написали скетч, который позволяет отправлять данные непосредственно на платформу Xively. Таким образом, данные о потреблении энергии постоянно хранятся на серверах Xively и могут контролироваться в режиме реального времени.

До сих пор мы использовали готовые платы и модули Arduino. Однако в следующей главе мы изучим основы создания собственных устройств домашней автоматики на базе Arduino.

6

Взлом коммерческого устройства домашней автоматики

В предыдущих главах мы использовали платы Arduino и другие компоненты для создания систем домашней автоматики с нуля. В этой главе мы собираемся применить другой подход к созданию систем домашней автоматики.
Мы возьмем простой выключатель питания, который можно легко купить в магазинах, и взломаем его, вставив в него плату Arduino. Таким образом, мы собираемся создать выключатель питания, управляемый USB, на базе Arduino.

Следующие основные выводы из этой главы:

- Сначала мы посмотрим, как выбрать устройство для взлома и какие компоненты вам понадобятся для взлома. Мы также позаботимся о различных необходимых программных модулях.
- Затем мы откроем устройство для взлома и заменим большую часть того, что находится внутри, на плату Arduino и модуль реле.
- Затем мы загрузим код на плату Arduino, чтобы проверить, правильно ли работает устройство. На этом этапе мы уже собираемся управлять устройством удаленно из Arduino IDE.
- Наконец, мы собираемся создать веб-интерфейс для управления устройством прямо из вашего веб-браузера.

Компоненты и программы

Этот проект основан на взломе коммерческого устройства для управления светом прямо с вашего компьютера через USB. Однако на самом деле цель состоит в том, чтобы научить вас, как взломать собственное устройство, поэтому устройство, которое вы собираетесь взломать, не имеет большого значения.
Следовательно, упоминание конкретного бренда или продукта не имеет значения.

Поскольку эта статья посвящена взлому устройства, которое используетопасное напряжение, такое как 110 В или 230 В, его следует рассматриватьтолько как источник информации и не использовать как таковое в вашем домеиз соображений безопасности. В самом деле, нет никакого способа бытьуверенным, что устройство будет продолжать работать должным образом вдолгосрочной перспективе после установки в вашем доме.
Кроме того, никогда не прикасайтесь к внутренним частям устройства, когда оновключено.

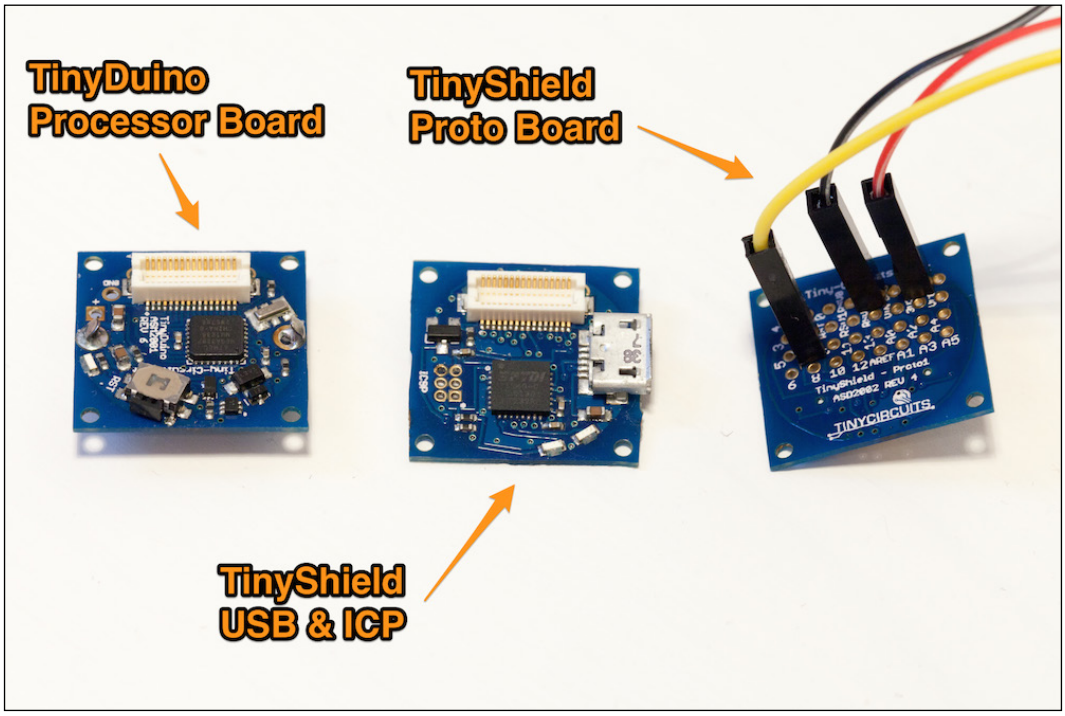
В первую очередь вам нужно устройство, которым можно управлять удаленно. Например, я выбрал набор из четырех идентичных устройств, которыми можно управлять с помощью небольшого пульта дистанционного управления. Каждое из этих устройств представляет собой переключатель включения / выключения, которым можно управлять с помощью реле, расположенного внутри устройства.
Эти устройства были очень дешевыми (9,90 доллара за четыре устройства) и идеально подходили для взлома, поскольку их легко было открыть. На следующем изображении показано такое устройство:

Вы можете взять любое подобное устройство, главное, чтобы им можно было так или иначе управлять с помощью электронной системы, например, пульта дистанционного управления. Это важно, так как позже мы вставим реле внутрь корпуса для управления переключателем. Механически управляемые переключатели (например, с одной кнопкой включения / выключения и без электроники внутри) не подойдут для этого проекта.

Затем вам нужно что-то вставить в устройство, которое мы собираемся взломать. Сначала нам понадобился «мозг» системы, которым будет плата Arduino. Проблема заключалась в том, что устройство, которое я выбрал для этого проекта, было слишком маленьким, чтобы поместить в него плату Arduino Uno.

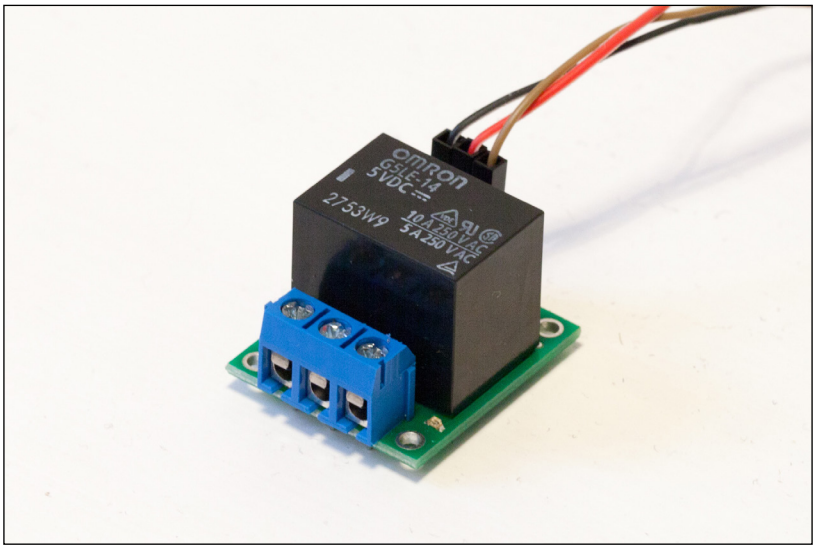
Поэтому мне пришлось использовать меньшую плату Arduino. Я решил использовать платы TinyCircuits, которые являются одними из самых маленьких доступных плат Arduino. Я использовал микроконтроллер, USB и прототип платы, который вы можете найти на http://tiny-circuits.com/.

Ниже приводится изображение плат Arduino, которые я использовал для взлома устройства:



Конечно, вы можете использовать любые другие небольшие платы Arduino, такие как Arduino Micro, или Arduino Mini для этого проекта. Подойдет любая базовая плата Arduino, которая подходит к устройству, которое вы хотите взломать.

Затем вам понадобится модуль реле для управления вилкой. Я использовал релейный модуль Polulu на 5 В, тот же, что и в главе 2, Управляйте светом с телефона или планшета:



Это список частей, использованных в этой главе:

- Плата процессора TinyDuino
- Tiny шилд USB и ICP
- Прототип платы TinyShied
- Релейный модуль 5V фирмы Pololu

Что касается программного обеспечения, на вашем компьютере должна быть установлена последняя версия Arduino IDE, а также библиотека REST для Arduino, которую вы можете найти по следующей ссылке:

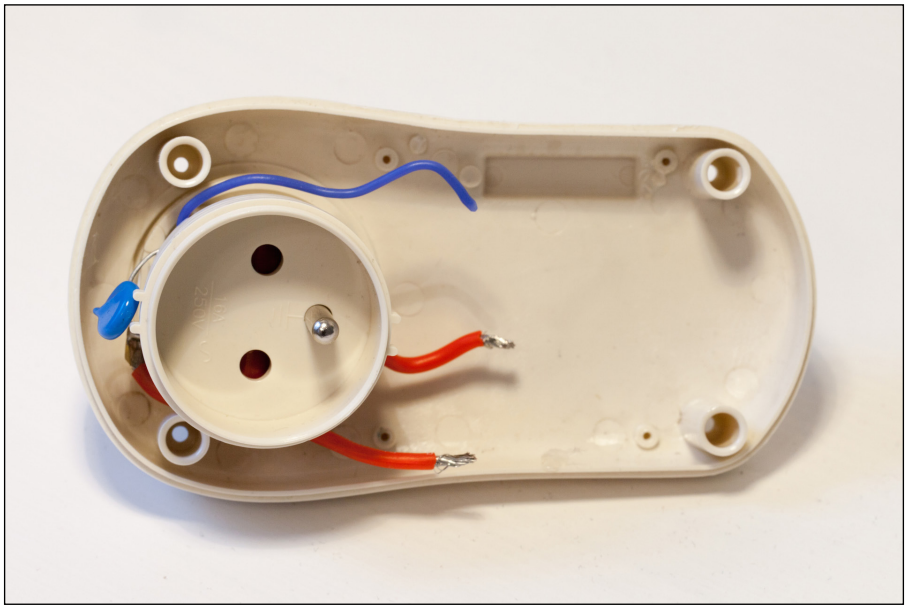
```
https://github.com/marcoschwartz/aREST
```

Вы можете установить библиотеку, просто распаковав папку библиотеки в папку Arduino / libraries (вам нужно будет создать эту папку, если ее еще нет).

Вам также потребуется установить и запустить веб-сервер на вашем компьютере, чтобы мы могли использовать веб-интерфейс, который мы собираемся разработать в конце главы.

Конфигурация оборудования

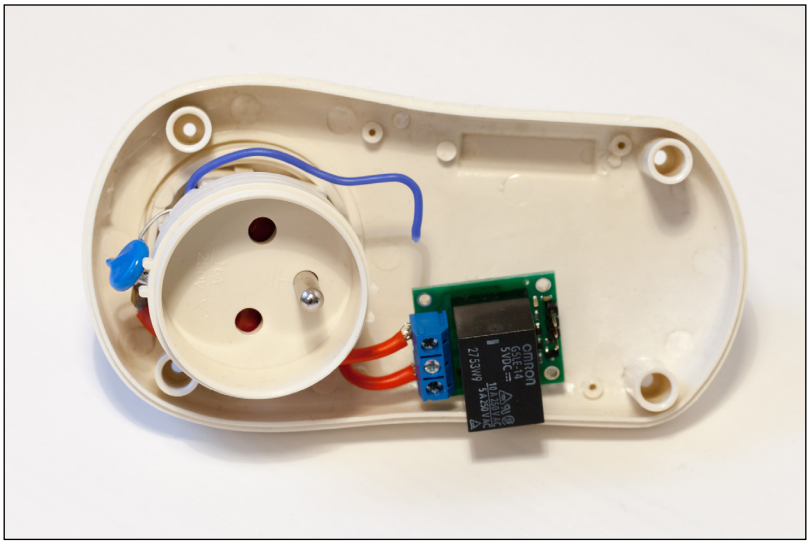
Первый шаг - открыть устройство, которое вы хотите взломать. В устройстве, которое я использовал, было много электроники, которую я просто удалил (нам больше не понадобятся эти компоненты), оставив только важные детали: вилку питания с обеих сторон устройства и два кабеля, идущих к обеим частям вилки:



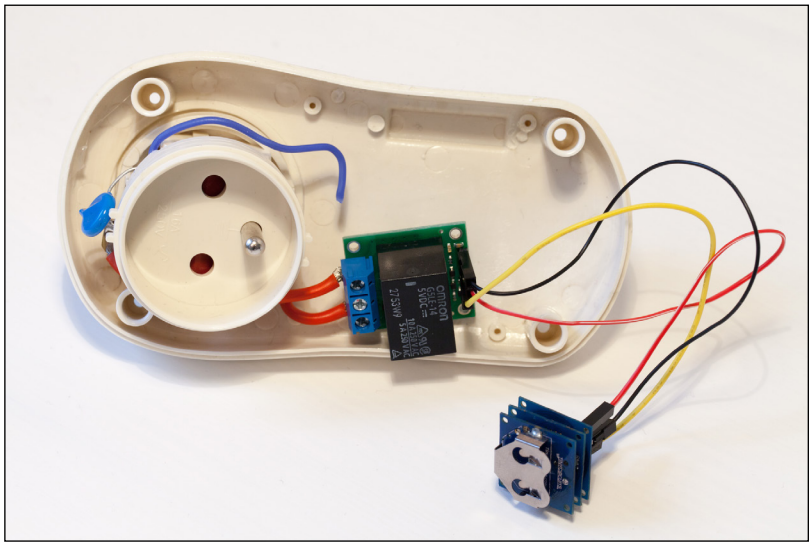
Эти два кабеля должны быть соединены вместе, если мы хотим, чтобы ток протекал в устройство, подключенное к взломанному устройству. Чтобы сделать это и управлять разъемом с платы Arduino, вам нужно вставить модуль реле.

По сути, реле имеет три выходных контакта: общий контакт с именем COM, нормально замкнутый контакт (NC) и другой нормально разомкнутый контакт (NO). Нормально замкнутый означает, что этот вывод подключен к
контакт COM реле, когда на вход реле нет напряжения. С другой стороны, нормально разомкнутый означает, что этот вывод отключен, когда на реле не подается напряжение.

Вам необходимо подключить один из красных кабелей, идущих к вилке, к контакту COM реле, а другой - к контакту NO реле, чтобы цепь замыкалась, когда реле срабатывает. На следующем изображении показано устройство на этом этапе:



Затем вам нужно вставить плату Arduino в устройство и подключить к ней релейный модуль. Подключения к плате Arduino действительно просты: подключите вывод VCC модуля реле к выводу Arduino + 5V, вывод GND - к выводу Arduino GND, а вывод SIG - к выводу 7. Это показано на следующем снимке экрана:



Теперь вы можете подключить USB-кабель к плате Arduino, чтобы управлять им со своего компьютера. Для этого было проделано в корпусе отверстие.

С устройством, которое я использовал, это выглядело в конце так:

Теперь вы готовы написать код Arduino и протестировать устройство, которое вы только что взломали!

Управление устройством с вашего компьютера

Теперь мы собираемся написать скетч Arduino, который мы будем использовать для дистанционного управления реле,который будет основан на библиотеке REST. Он поможет нам обработать запрос, отправленный с вашего компьютера.

Скетч начинается с включения необходимых библиотек:

```
#include <SPI.h>
#include <aREST.h>
```

Создадим функцию aREST

```
// Create aREST instance
aREST rest = aREST();
```

Затем в функции скетча setup () мы просто запускаем последовательное соединение, чтобы позже получить доступ к плате:

```
void setup(void)
{
  // Start Serial
  Serial.begin(9600);

}
```

Затем мы можем обрабатывать запросы, поступающие на последовательный порт, используя следующий код:

```
void loop() {

  // Handle REST calls
  rest.handle(Serial);

}
```

Теперь мы можем протестировать взломанное вами устройство. Обратите внимание, что код для этой части доступен в репозитории GitHub проекта по следующему адресу: https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter6

Включите устройство в сеть и подключите какую-нибудь нагрузку, чтобы проверить его. Например, для тестирования своего проекта я использовал простую настольную лампу мощностью 30 Вт.

Загрузите код для платы Arduino и откройте монитор последовательного порта внутри Arduino IDE. Убедитесь, что скорость последовательного порта установлена на 9600, и введите:

/mode/7/o

Установить контакт номер 7 как выход. Вас должно приветствовать следующее сообщение:

Setting pin D7 to output

Теперь вы можете переключать реле с помощью:

/digital/7/1

Вы должны услышать щелчок реле и увидеть, как включается устройство, которым вы пытаетесь управлять. У вас также должно быть следующее подтверждение на мониторе:

```
Pin D7 set to 1
```

Если это не сработает, в первую очередь проверьте соединения внутри устройства, которое вы взломали. Сначала отсоедините его от сетевой розетки и снова откройте. Убедитесь, что все подключено правильно в соответствии с предыдущим разделом.

Вы также можете протестировать реле независимо, подключив вывод SIG к выводу 5V, чтобы убедиться, что оно работает. Наконец, также убедитесь, что последовательная скорость совпадает со скоростью, определенной в скетче Arduino.

Создание графического интерфейса

Однако управлять взломанным устройством с последовательного монитора неудобно, и сейчас мы собираемся создать простой интерфейс для управления этим устройством. Интерфейс будет очень простым и будет включать две кнопки: одна для включения , а другая для выключения реле.

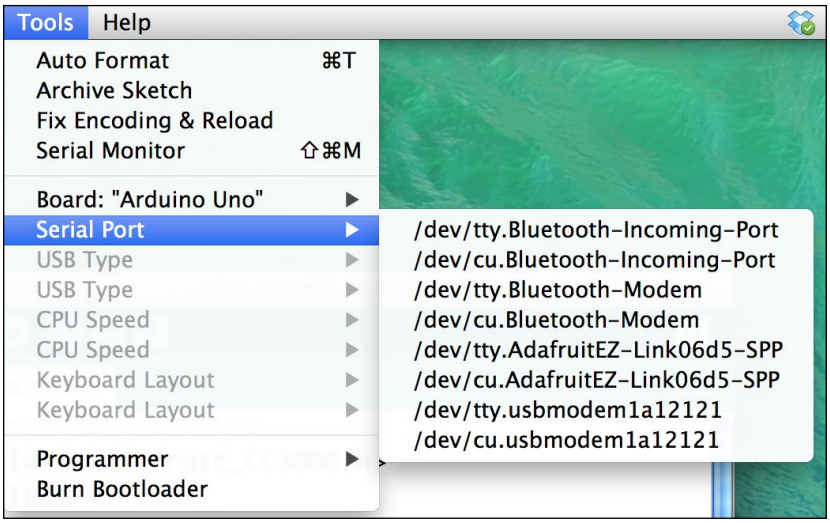
Интерфейс основан на HTML для самого интерфейса, JavaScript для обработки команд и PHP для взаимодействия с платой Arduino. Этот интерфейс очень похож на тот, который мы разработали в главе 2, Управляйте светом с телефона или планшета.

Следующий код представляет собой пошаговое руководство по различным частям интерфейса. Все файлы должны находиться в одной папке в корне папки вашего веб-сервера. Вы можете найти полный код этой части в репозитории книги на GitHub.

Давайте сначала посмотрим на HTML-файл, который содержит сам интерфейс. Внутри HTML-файла следующий код для кнопки:

```
<button class="btn btn-block btn-lg btn-primary" type="button"
  id="1" onClick="buttonClick(this.id)">On</button>
```

Внутри файла JavaScript мы сначала определяем тип связи и последовательный порт. Здесь вам нужно изменить значение цели, чтобы установить ее на свой собственный последовательный порт. Чтобы узнать, какой порт вам нужно выбрать, просто перейдите в меню «Инструменты» в вашей Arduino IDE:



Внутри этого файла JavaScript находится код, который вам необходимо изменить:

```
var target = '/dev/cu.usbserial-A102JJ6G';
var type = 'serial';
```

Когда окно загружается в первую очередь, нам нужно сначала установить контакт реле как выход. Это делается с помощью команды get внутри файла JavaScript:

```
window.onload = function() {
  $.get( "command.php", {target: target, type: type,
  command: "/mode/7/o"} );
}
```

Кнопки вызывают кнопку вызова функции. Щелкните и передайте их идентификатор в качестве аргумента этой функции. Например, это код кнопки ON:

```
if (clicked_id == "1"){
  $.get( "command.php", {target: target, type: type,
  command: "/digital/7/1"} );
}
```

Вы можете видеть, что все эти команды вызывают файл PHP. Этот файл отвечает за взаимодействие с платой Arduino. Файл начинается с получения различных аргументов, исходящих от интерфейса:

```
$type = $_GET['type'];
$target = $_GET['target'];
$command = $_GET['command'];
```

Поскольку в этом проекте мы всегда будем использовать последовательную связь через порт USB, я предоставлю подробную информацию только о той части, которая отвечает за последовательную связь. Мы назначаем цель переменной serial_port:

```
$serial_port = $target;
```

Теперь мы можем определить различные параметры последовательного соединения с платой Arduino:

```
$serial = new phpSerial;
$serial->deviceSet($serial_port);
$serial->confBaudRate(9600);
$serial->confParity("none");
$serial->confCharacterLength(8);
$serial->confStopBits(1);
```

Теперь откройте соединение:

```
$serial->deviceOpen();
```

Затем мы можем отправить следующую команду:

```
$serial->sendMessage($command . "\r");
  $answer = $serial->readPort();
```

И прочтите ответ (который мы не будем использовать в этом проекте):

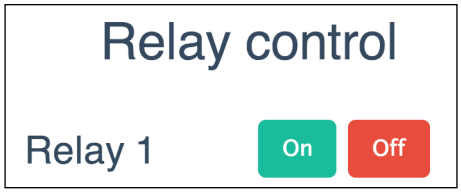
```
if ($answer == "") {echo "{\"connected\": false}";}
     else {echo $answer;}
```

Код для этой части доступен в репозитории GitHub проекта по следующему адресу:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter6

Пришло время протестировать интерфейс. Убедитесь, что все файлы интерфейса помещены в папку вашего веб-сервера и что веб-сервер запущен.

Теперь вы можете открыть файл HTML внутри папки интерфейса (обычно, набрав http: // localhost в вашем браузере), и вот что вы должны увидеть:



Теперь попробуйте нажать кнопку On. Вы должны сразу услышать переключение реле и увидеть, как загорится лампа, подключенная к взломанному устройству.

Если на данном этапе это не сработает, нужно проверить несколько вещей.
Прежде всего необходимо убедиться, что на вашем компьютере работает веб-сервер. Также убедитесь, что вы обращаетесь к файлу HTML через локальную папку хоста, а не напрямую, щелкая файл в проводнике файлов. Наконец, убедитесь, что оборудование работает правильно, попытавшись отправить команды прямо в Arduino IDE.
Вы также можете получить некоторую помощь по этому проекту, разместив свой вопросна официальном Arduinoforum по адресу http://forum.arduino.cc/.
Вы также можете найти похожие проекты и получить помощь по этой главе в блоге Open Home Automation по адресу http://www.openhomeautomation.net/blog.

Резюме

В этой главе мы взломали коммерческое устройство для управления светом прямо с вашего компьютера. Мы заменили внутреннюю электронику устройства, вставив плату Arduino и реле. Это дает нам гибкость системы на базе Arduino с красивым корпусом, поставляемым вместе с коммерческим устройством. После взлома устройства им можно будет управлять прямо из веб-браузера вашего компьютера.

Посмотрим, какие были основные выводы из этой главы. В первой части главы мы выбрали различные компоненты, необходимые для проекта, включая устройство, которое нужно взломать. Мы также выбрали небольшую плату Arduino и модуль реле.

Затем мы открыли устройство, которое нужно взломать, и вставили выбранные нами компоненты в первую часть проекта.

Затем мы написали скетч Arduino для управления устройством путем отправки команд через последовательный порт. Мы протестировали этот скетч, пытаясь включить и выключить реле.

Наконец, мы создали веб-интерфейс для управления взломанным устройством из вашего веб-браузера.

В следующей главе мы пойдем еще дальше и посмотрим, как создавать собственные устройства с нуля. Мы собираемся создать систему Arduino с низким энергопотреблением, которая может работать годами от простой батареи!

7

Создайте свою домашнюю автоматику

До сих пор в этой книге мы использовали только уже собранные платы Arduino и сопрягали их с различными датчиками и модулями для создания проектов домашней автоматики. В предыдущей главе мы немного изменили ситуацию и взломали коммерческое устройство с помощью Arduino.

Однако в этой последней главе книги мы собираемся пойти дальше и построить нашу собственную систему для домашней автоматики. В качестве примера я возьму проектирование и изготовление очень простой маломощной платы Arduino. Затем эту плату можно использовать для проектов домашней автоматики, таких как датчики движения с батарейным питанием.

Обратите внимание, что цель этой последней главы книги - дать вам пошаговый метод построения вашей собственной домашней системы автоматики. Следовательно, вам следует сосредоточиться не на конкретном примере, а на общей методологии разработки ваших собственных систем.

Вот основные выводы из этой главы:

- Сначала мы выберем необходимые компоненты для проекта и установим библиотеку, предназначенную для маломощных приложений.
- Затем мы собираемся построить прототип нашей системы домашней автоматики на макете, чтобы убедиться, что она работает правильно. Мы также собираемся протестировать маломощную библиотеку Arduino, чтобы убедиться, что плата будет энергоэффективной.
- В следующей части главы мы перейдем от этапа прототипирования к собственному созданию плат. Мы увидим, как использовать программу для проектирования печатных плат, чтобы вы могли сделать свою собственную печатную плату, а затем отправить ее производителю для изготовления.
- Наконец, мы увидим, как напечатать на 3D-принтере корпус для вашей домашней системы автоматики и изготовить его.

Компоненты и программы

Давайте сначала посмотрим, какие компоненты необходимы для этого проекта. В качестве примера в этой главе я буду использовать проект, над которым я работал ранее, а именно плату Arduino с низким энергопотреблением. На этой плате имеется минимум компонентов, которые можно использовать. Таким образом, энергия не расходуется на компоненты, которые не нужны при работе системы, такие как светодиоды, интерфейс USB и т. д. Затем эту плату можно использовать в приложениях с очень низким энергопотреблением, таких как датчики движения на базе XBee.

Самым важным компонентом в этом проекте будет сам микроконтроллер. Микроконтроллер лежит в основе всех плат Arduino. Это компонент, который мы программируем каждый раз, когда пишем новый скетч.

Я основывал свой проект на дизайне платы Arduino Uno, поэтому я использовал тот же контроллер Atmel ATmega328, что и плата Arduino Uno. Для этого проекта вам понадобится один из этих микроконтроллеров, которые продаются отдельно. Убедитесь, что вы покупаете его с загрузчиком Arduino, чтобы вы могли напрямую загрузить код Arduino в микроконтроллер.

Для работы микроконтроллера вам также понадобятся несколько более мелких компонентов: один конденсатор 10 мкФ, два конденсатора 22 пФ, один резистор 10 кОм, один резистор 220 Ом, один зеленый светодиод и один кварцевый генератор на 16 МГц.

Вам также понадобится аккумулятор для питания проекта. Я использовал батарейный блок, который может содержать две батарейки AA (1,5 В каждая) для питания проекта. Вы можете использовать батарею по своему выбору, помня, что в официальной документации к этой микросхеме указано, что напряжение должно быть от 1,8 до 5,5 В. Тем не менее, я действительно рекомендую использовать для питания проекта не менее 3 В.

Чтобы запрограммировать микроконтроллер, вам понадобится внешний программатор. Вместо того, чтобы использовать специальный программатор, я просто использовал для этой задачи плату Arduino Uno.

Наконец, вам понадобится макетная плата и несколько перемычек для соединения компонентов между собой.

Это список всех частей, которые использовались в этом проекте:

- ATmega328 microcontroller with Arduino bootloader
- Конденсатор 10 мкФ
- 2 конденсатора по 22 пФ
- Резистор 10 кОм
- Резистор 220 или 330 Ом
- Зеленый светодиод
- Кварц 16 МГц
- 2 батарейки типа AA 1,5 В)
- Макетная плата)
- Перемычки папа / мама

Что касается программного обеспечения, вам необходимо загрузить библиотеку `JeeLib Arduino`, чтобы обеспечить работу вашей системы с низким энергопотреблением:

```
https://github.com/jcw/jeelib
```

Чтобы установить библиотеку `Arduino`, просто поместите папку библиотеки в папку `/ libraries` основной папки `Arduino`.

Вам также необходимо загрузить `EAGLE`, чтобы спроектировать собственную печатную плату `(PCB)` для этого проекта. Вы можете скачать его по следующей ссылке:

```
http://www.cadsoftusa.com/download-eagle/
```

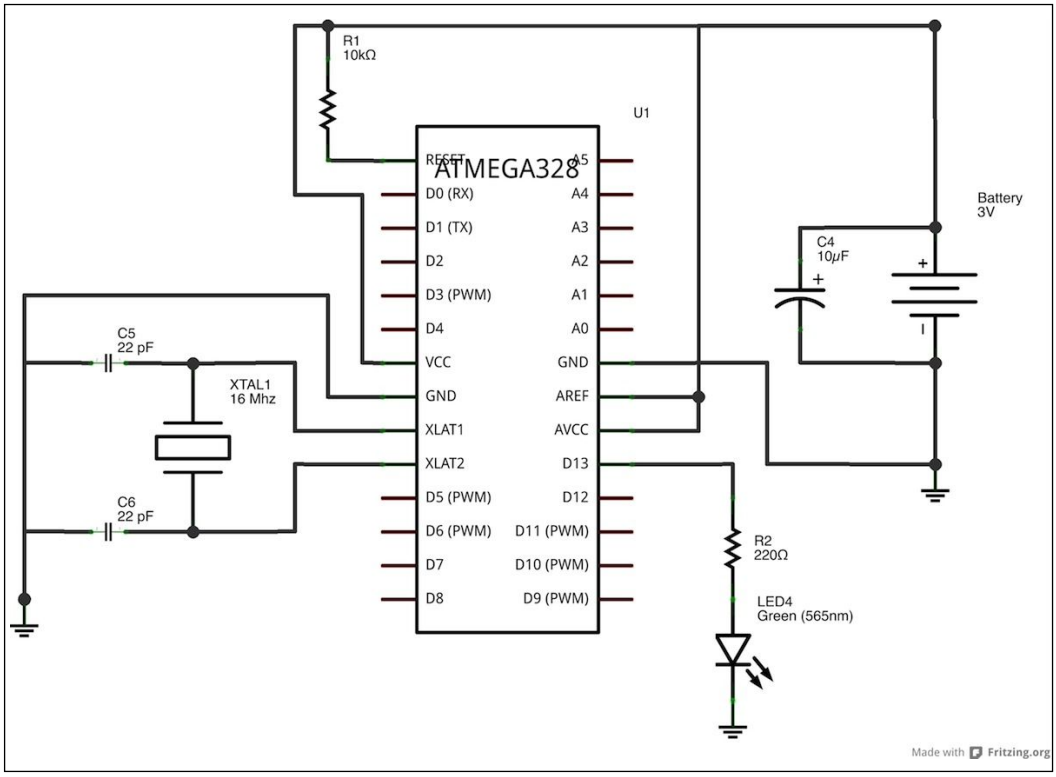
Чтобы спроектировать свой собственный корпус, напечатанный на `3D`-принтере, вы можете использовать `OpenSCAD`, бесплатный инструмент для проектирования. Вы можете найти его по следующему адресу:

```
http://www.openscad.org/
```

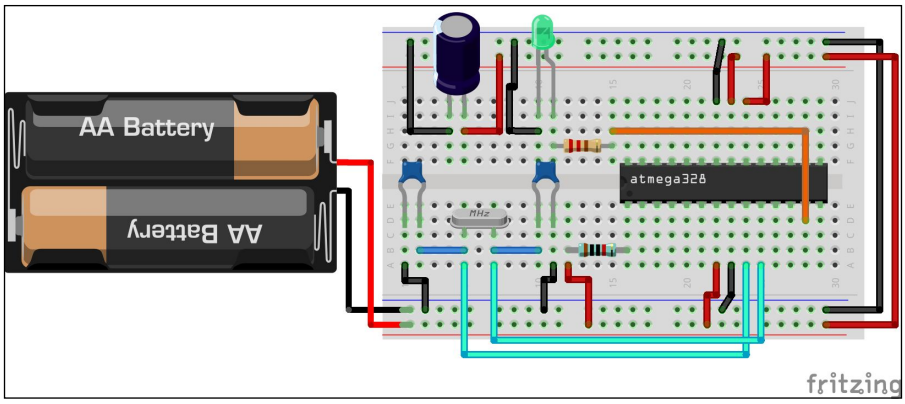
Создание системы Arduino с нуля

Давайте теперь посмотрим, как построить систему Arduino на макете, используя выбранные нами компоненты.

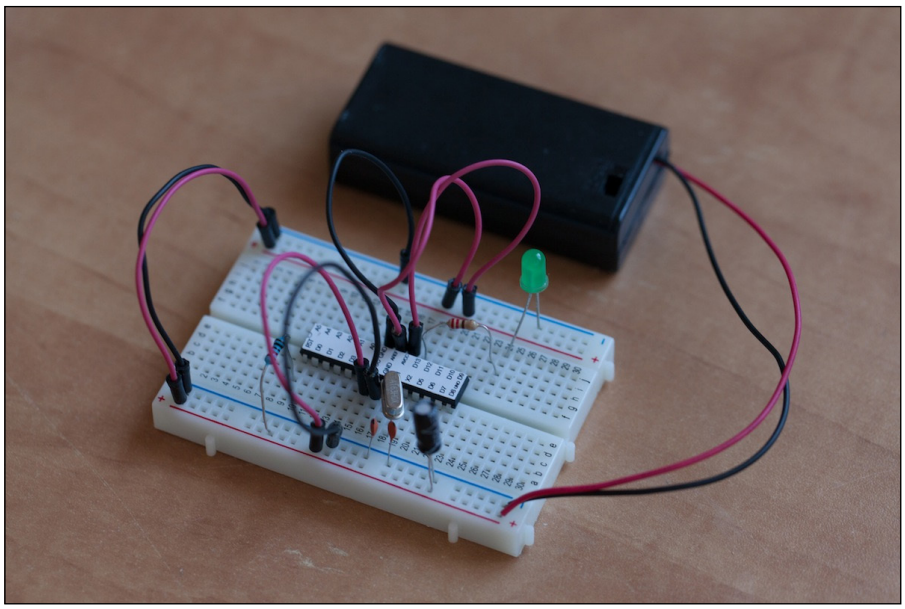
Пользуясь приведеной схемой соберите устройство на макетной плате:



Я рекомендую сначала разместить микроконтроллер Arduino (ATmega328) в середине макета, чтобы вы могли добавлять вокруг него различные компоненты. На этой схеме показано размещение компонентов:



На следующем изображении показана полностью собранная система с батарейками:



Теперь мы можем протестировать проект, чтобы убедиться, что он работает правильно. Для этого мы собираемся заставить светодиод мигать с помощью скетча Blink, который поставляется с Arduino IDE.

Тестирование системы Arduino

Пришло время протестировать систему Arduino. Сделаем это следующим образом:

- Сначала снимите чип и поместите его на плату Arduino Uno, которая должна быть подключена к вашему компьютеру. Чтобы снять микросхему с макетной платы или платы Arduino Uno, используйте экстрактор. Конечно, всякий раз, когда вы снимаете чип с платы, убедитесь, что на него не подается питание.

- Загрузите скетч Blink, поставляемый с IDE Arduino, и загрузите код на плату. Вы должны увидеть, что встроенный светодиод замигал.

- Теперь отключите плату от компьютера и снимите микроконтроллер. Вставьте его обратно в макетную плату и подключите питание к аккумуляторной батарее. Вы должны увидеть, что светодиод на макете мигает.

Если это не сработает, вы можете проверить несколько вещей. Во-первых, убедитесь, что вы правильно соединили разные части проекта согласно схеме. Затем убедитесь, что микроконтроллер был правильно запрограммирован, проверив, что встроенный светодиод платы Arduino Uno мигает.

Теперь мы также можем провести тест с библиотекой JeeLib, чтобы перевести контроллер Arduino в режим глубокого сна, когда он не активен. Действительно, без указания чего-либо в коде микроконтроллер Arduino будет потреблять энергию, даже если он ничего не делает. Нам нужно объявить конкретные команды в коде, чтобы поместить Arduino в спящий режим с низким энергопотреблением.

Теперь посмотрим код для этой части. Мы собираемся увидеть самые важные элементы кода. Вы можете найти полный код в репозитории GitHub этой главы.

Этот скетч начинается с включения библиотеки JeeLib следующим образом:

```
#include <JeeLib.
```

Чтобы убедиться, что микроконтроллер Arduino активен, мы также заставим светодиод мигать. Однако между каждым изменением состояния светодиода мы переводим микроконтроллер в режим глубокого сна. Сначала нам нужно определить вывод светодиода, которым мы хотим управлять:

```
int led_pin = 13;
```

Мы также должны запустить сторожевой таймер, который необходим для перевода Arduino в спящий режим:

```
ISR(WDT_vect) { Sleepy::watchdogEvent(); }
```

В функции скетча loop () мы включаем и выключаем светодиод, но здесь важно то, что мы используем специальную функцию для введения некоторой задержки:

```
Sleepy::loseSomeTime(5000);
```

Разница с функцией delay () Arduino заключается в том, что контроллер Arduino в это время переводится в спящий режим, тогда как с функцией delay () чип просто ждет.

Код для этой части можно найти в репозитории проекта GitHub по следующей ссылке:
https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter7

Я выполнил некоторые измерения, чтобы показать вам разницу с библиотекой и без нее. Без использования библиотеки схема потребляла 6,7 мА, когда светодиод был выключен. При использовании библиотеки она потребляет всего 43 мкА, что в 150 раз меньше, чем в первом случае.

Это означает, что для схемы, которая большую часть времени неактивна (например, датчик температуры или датчик движения), она может работать годами без замены батареи!

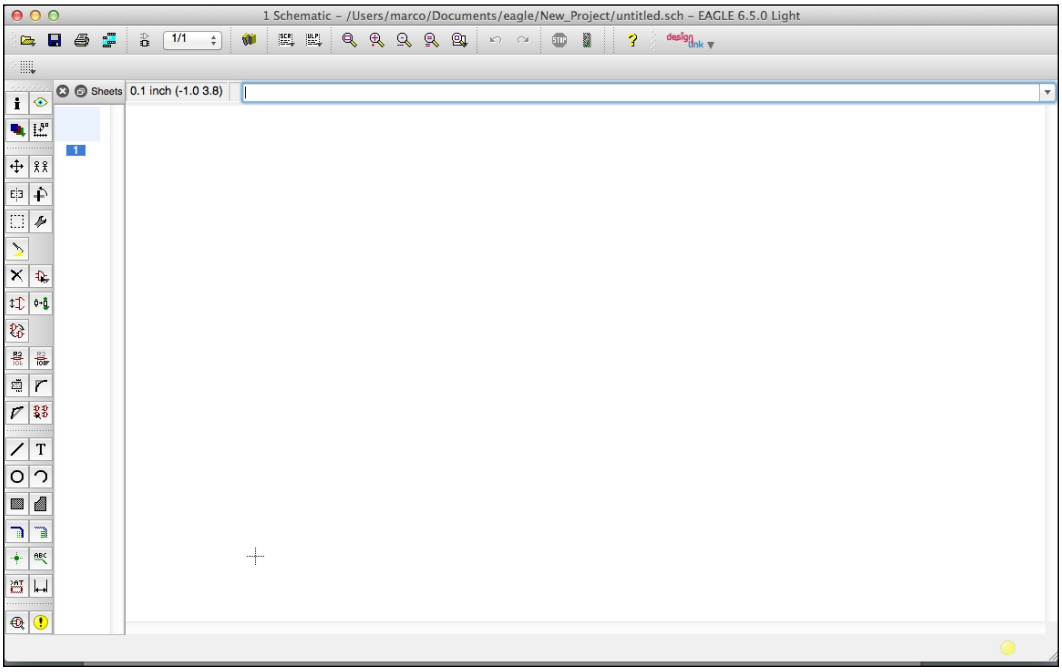
Разработка печатной платы для системы домашней автоматики

Пока что в этой главе мы узнали, как создать прототип системы на базе Arduino на макетной плате. Теперь мы посмотрим, как взять идею из прототипа, который мы построили на макетной плате, и построить из нее нашу собственную плату.

Первый шаг - спроектировать плату на вашем компьютере. Для этого я использовал программу EAGLE, которое широко используется для разработки систем на базе Arduino. Вы можете скачать EAGLE, перейдя на следующую страницу:

http://www.cadsoftusa.com/download-eagle/

Первое, что вам нужно сделать, это открыть программу и создать новую схему. Вы попадете в новое пустое окно, подобное тому, что показано на следующем снимке экрана, где вы создадите свою схему:

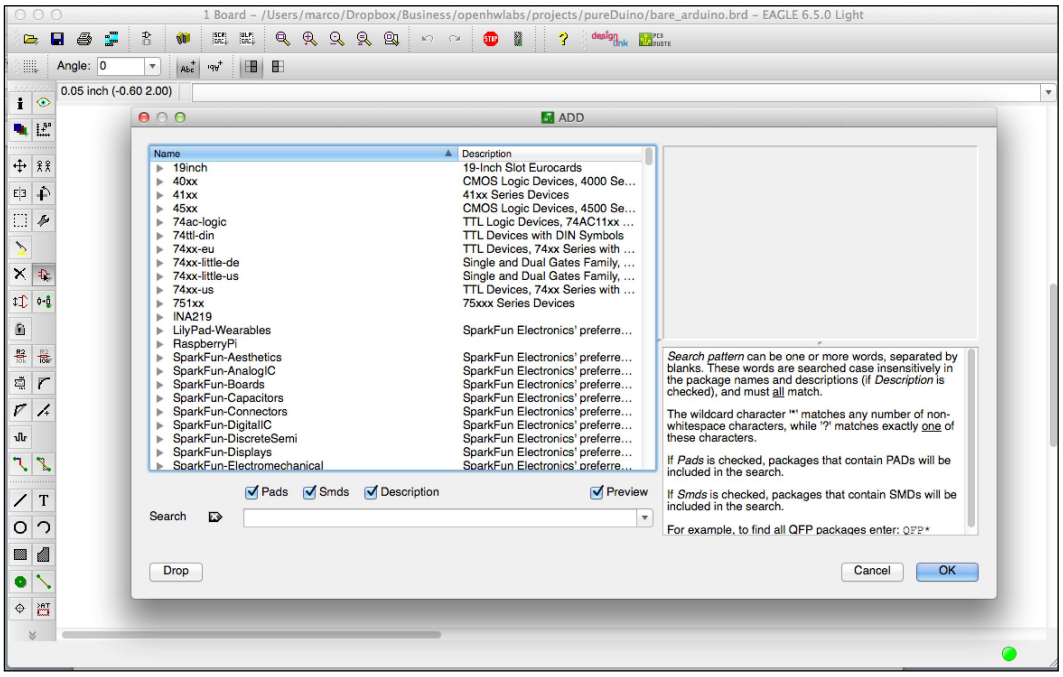


В этом разделе главы я хочу показать вам, как построить свою собственную плату, а не создавать ту же плату, которую я использовал в качестве примера.

Поэтому я дам вам обзор того, как создать плату в EAGLE. На этом этапе вы действительно должны сосредоточиться на том, как использовать наиболее важные команды программы.

Первое, что мы сделаем, - это создадим схеме нашего проекта. Это означает выбор компонентов (называемых частями), которые мы хотим использовать в нашем проекте, размещение их на экране и соединение их проводниками.

Важная команда, которая вам сейчас понадобится, - это команда для размещения детали, которая находится в меню слева. После того, как вы нажмете на этот значок, вы попадете в меню, в котором вы можете выбрать части, которые хотите использовать в своем проекте, как показано на следующем снимке экрана:

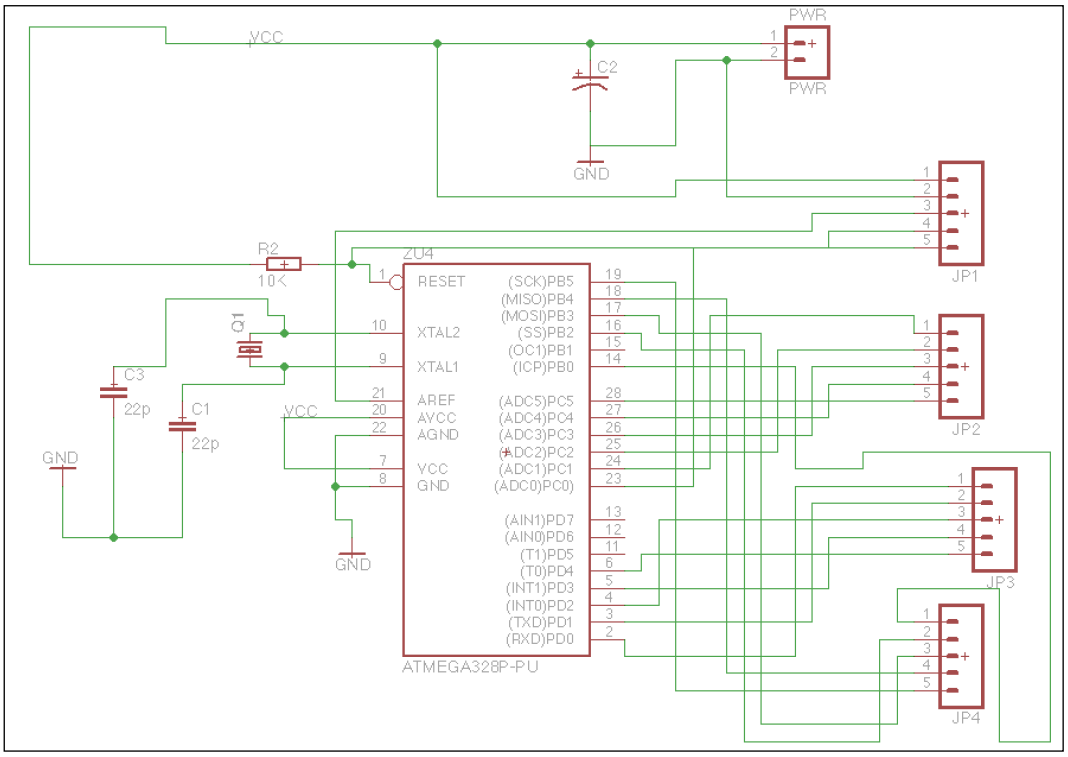


В этом меню вы можете искать детали, которые хотите использовать в своем проекте. Каждый раз, когда вы нажимаете на деталь и нажимаете ОК, вы сможете разместить ее на главном экране вашего проекта.

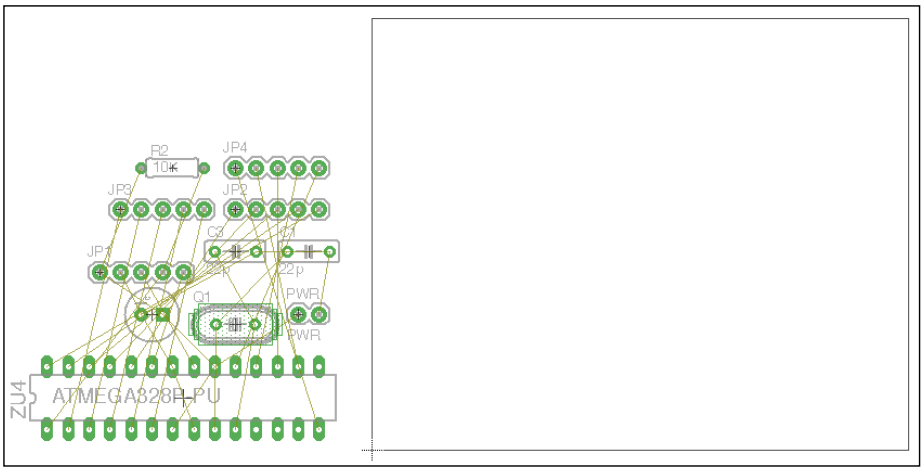
Вам также необходимо будет провести связи между различными частями системы. Это можно сделать с помощью функции Net, которую вы также найдете в левом меню.

Я разместил и соединил все компоненты, которые были у меня в предыдущем разделе проекта. Я также добавил несколько разъемов (чтобы получить доступ к контактам микроконтроллера, как на плате Arduino Uno) для связи контроллера Arduino с внешним миром, а также разъем для аккумулятора.

Вот результат, который у меня был на моем экране:

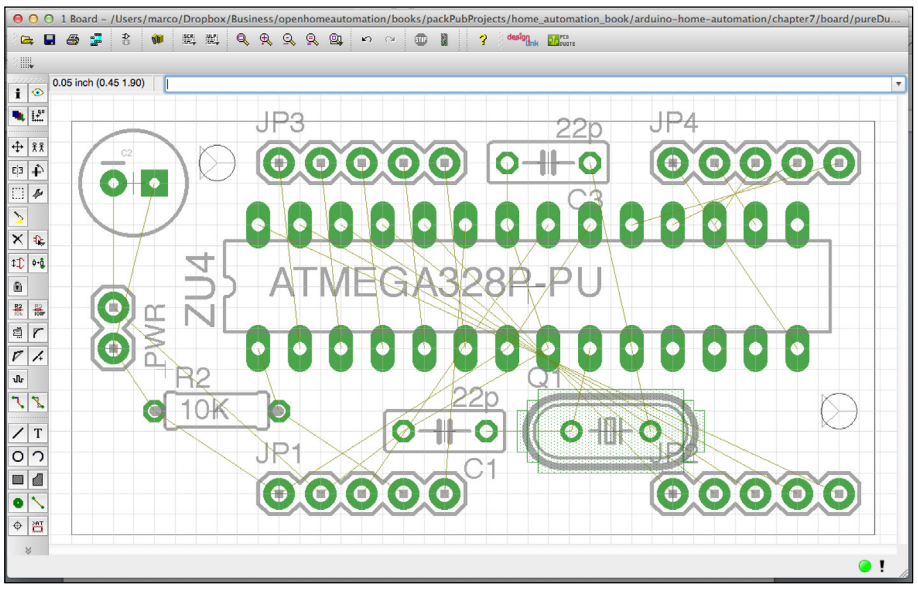


Отсюда вы сможете выполнить второй шаг в проектировании печатной платы: макет. Это реальное физическое представление вашей платы, которое будет отправлено производителю. Вы можете нажать кнопку Layout в верхнем меню, и EAGLE спросит вас, хотите ли вы создать макет автоматически. Нажмите Да, и вы должны увидеть следующий экран:

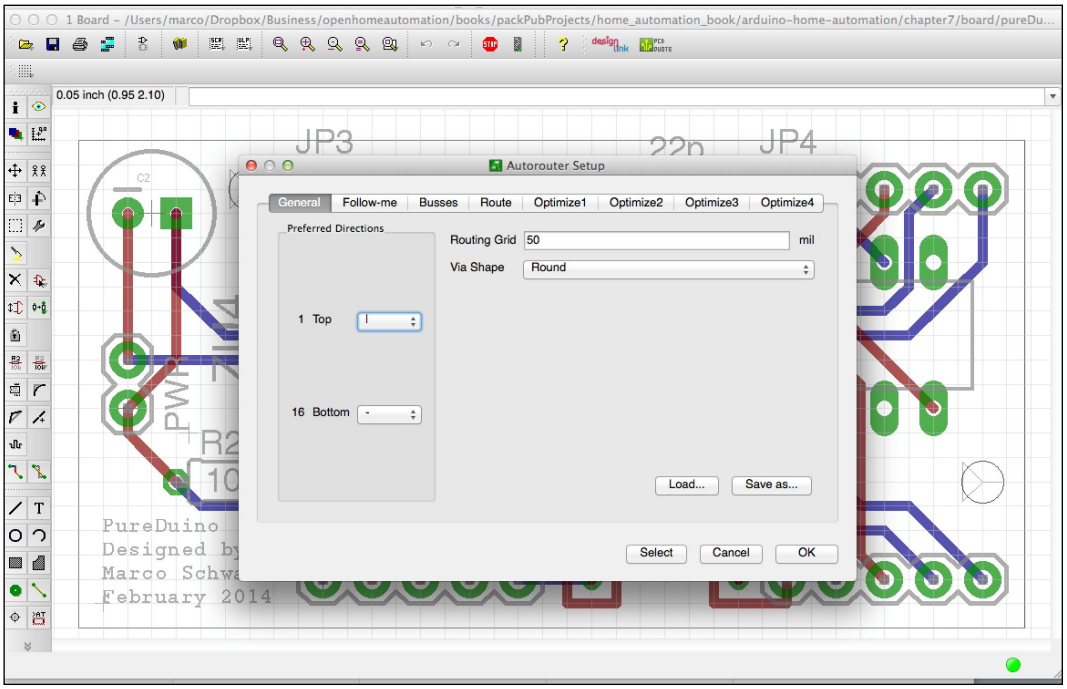


Отсюда ваша задача - разместить на плате компоненты (которые мы выбрали на предыдущем шаге при создании схемы), расположить их по своему усмотрению и развести плату. Маршрутизация означает соединение различных компонентов на плате с помощью проводников. Это можно сделать автоматически, используя функцию автоматической маршрутизации EAGLE. Вы можете видеть, что EAGLE уже указывает, какие компоненты соединены вместе, отображая тонкие линии между их контактами.

В моем примере платы я сначала разместил все компоненты, чтобы использовать как можно меньше места на плате, оставив при этом некоторое пространство для размещения на плате. Это показано на следующем снимке экрана:

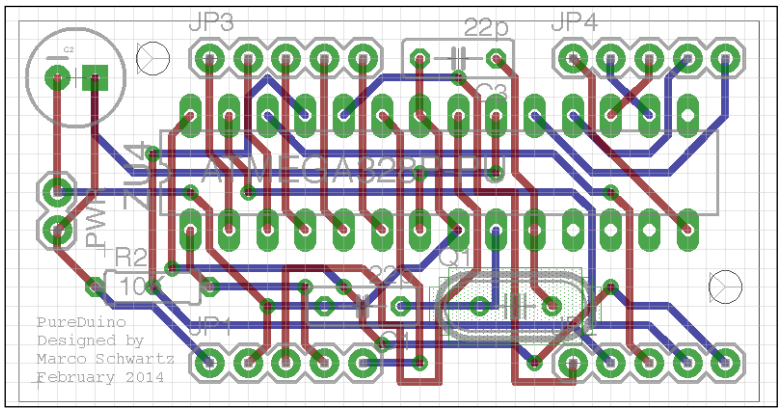


Затем я использовал функцию автоматической маршрутизации EAGLE для автоматического подключения всех компонентов на плате в соответствии со схемой, как показано на следующем снимке экрана:



Еще я вставил два отверстия, чтобы плату можно было закрепить на корпусе. Наконец, я вставил текст в нижний левый угол платы. Это легко сделать с помощью встроенного инструмента «Текст», который также можно найти в меню слева.

Вот окончательный результат в EAGLE:



Вы можете видеть, что контроллер Arduino расположен в середине платы, в окружении контактов ввода-вывода и компонентов, необходимых для работы Arduino.

Код для этой части можно найти в репозитории GitHub проекта по следующей ссылке:

https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter7

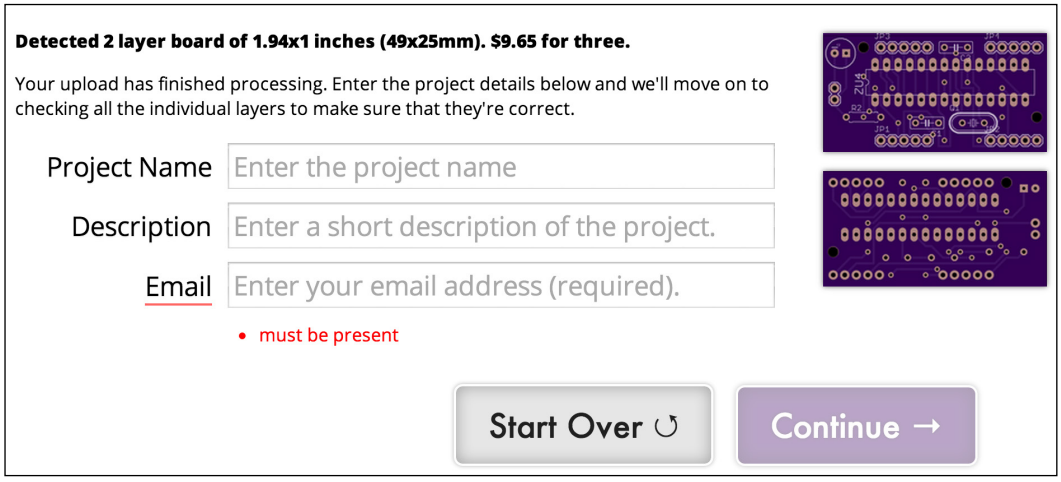
Изготовление платы

Когда этап проектирования завершен, пора изготовить плату, заказать компоненты и собрать ее. В этом разделе мы увидим, как выполнить каждый из этих шагов.

На первом этапе я обычно использую услуги OSH Park для создания своих плат небольшими сериями. Это дешево, быстро, и вы можете отправлять файлы EAGLE напрямую, не конвертируя их в другой формат. Вы также можете использовать сервисы других производителей, например Seeedstudio или Iteadstudio. Ссылка для доступа к производственным услугам OSH Park следующая:

https://oshpark.com/

С главной страницы действительно легко изготовить вашу плату. Вам просто нужно загрузить файл платы EAGLE, и веб-сайт автоматически проанализирует его. После этого вы попадете на следующую страницу подтверждения:



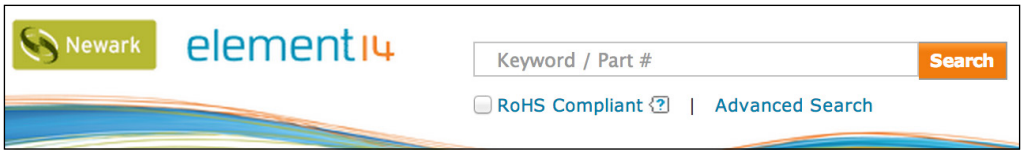
Затем вас попросят ввести дополнительные сведения, и вы сможете заказать и оплатить свою плату. Примерно через две недели вы получите ее по почте.

Для второго шага при заказе необходимых компонентов, я рекомендую использовать Newark. В них есть все компоненты, необходимые для платы, которую я использовал в качестве примера. Ссылка следующая:

http://www.newark.com/

Конечно, у вас есть много альтернатив Newark. Вы также можете использовать такие веб-сайты, как Mouser, Digikey, Farnell или SparkFun.

На веб-сайте Newark вы можете легко найти нужные компоненты с помощью основного поля поиска, как показано на следующем снимке экрана:



Когда вы получите плату и компоненты, вы можете спаять плату с помощью паяльника. Для моей платы после ее сборки я повторил тот же процесс, что и в предыдущем разделе. Сначала я запрограммировал контроллер с помощью платы Arduino Uno, а затем вернул контроллер на свою собственную плату для тестирования.

Дизайн и 3D печать корпуса для проекта домашней автоматики

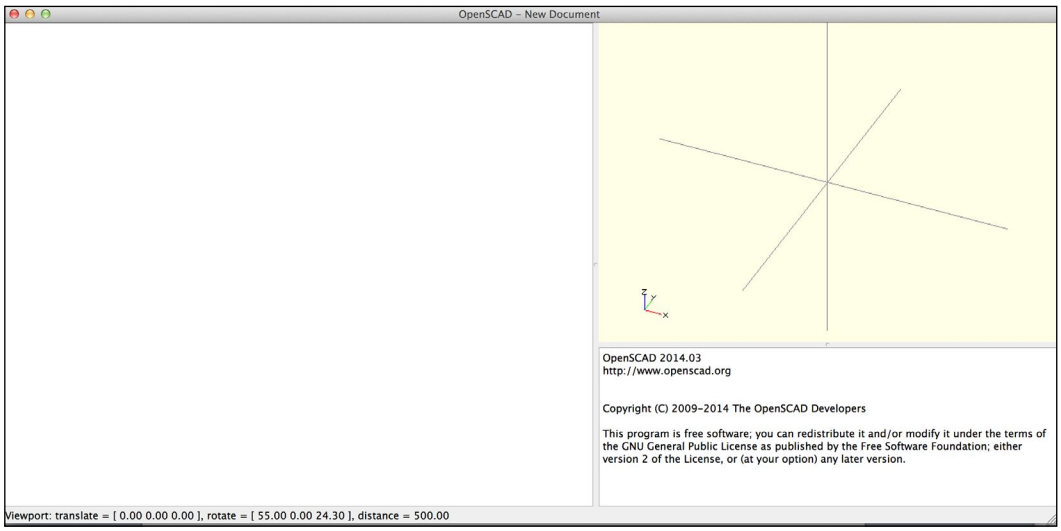
Третий шаг в разработке собственной домашней автоматики - это разработка индивидуального корпуса для вашего продукта. В прошлом это был действительно дорогостоящий процесс, так как вам нужно было найти производителя вашего продукта и, как правило, произвести несколько тысяч единиц, чтобы иметь хорошую цену за каждую произведенную единицу.

К счастью, 3D-печать все изменила. Теперь мы можем создавать прототипы отдельных изделии по очень низкой цене, используя эту технологию. В этом разделе мы увидим, как создать трехмерный дизайн простого корпуса для нашей системы Arduino. Мы также посмотрим, как отправить его производителю, чтобы его можно было распечатать в 3D.

И для этого мы будем использовать программу под названием OpenSCAD. Это программа с открытым исходным кодом для проектирования, в котором вы можете писать код для создания своих проектов. Это идеально, если вы не дизайнер, а скорее программист, а также позволяет создавать параметрические проекты, которые можно легко изменять и публиковать. Вы можете загрузить OpenSCAD по следующему URL-адресу:

http://www.openscad.org/

Когда вы открываете новый проект в OpenSCAD, вы должны получить следующее:



Вы можете увидеть, что в окне OpenSCAD в основном три части. Слева есть место для кода, справа - трехмерный вид объекта, а в правом нижнем углу - окно для печати информации о проекте.

Например, чтобы сделать простой куб размером 5 x 5 x 5 мм, вы можете просто ввести: `cube ([5,5,5]);`

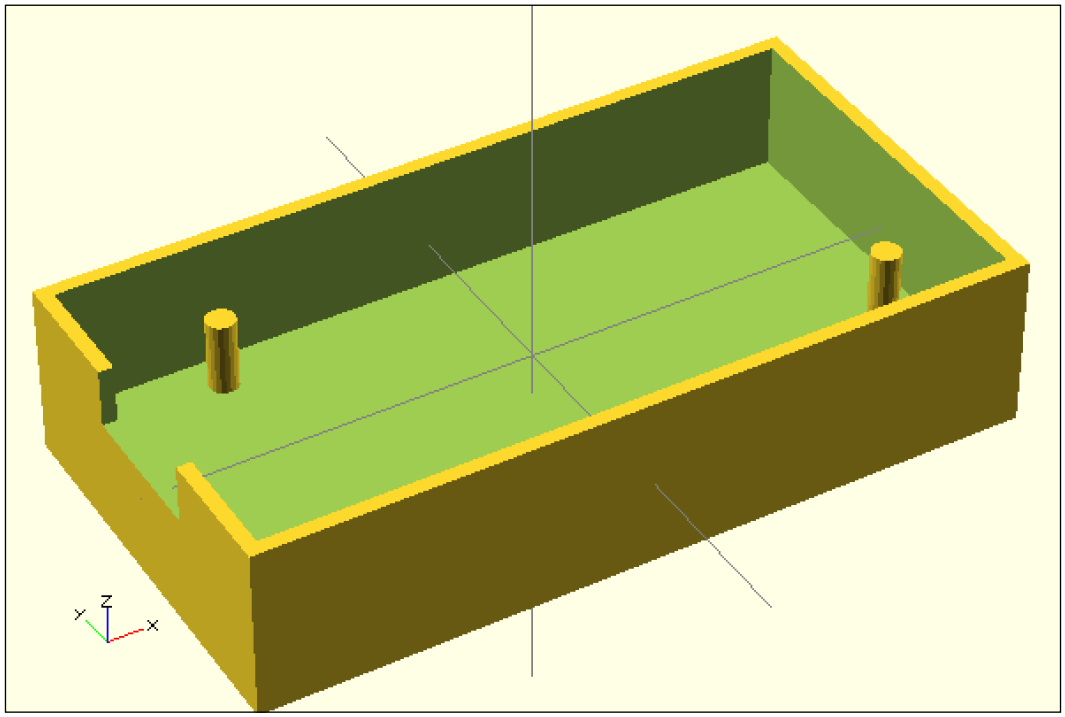
Что касается других функций, таких как различие между двумя объектами, вы можете посмотреть официальную документацию программы по следующему URL-адресу:

`http://www.openscad.org/documentation.html`

Для платы, которую я взял в качестве примера, я разработал простой корпус,где она будет крепиться, и сделал вырез для внешних подключений.

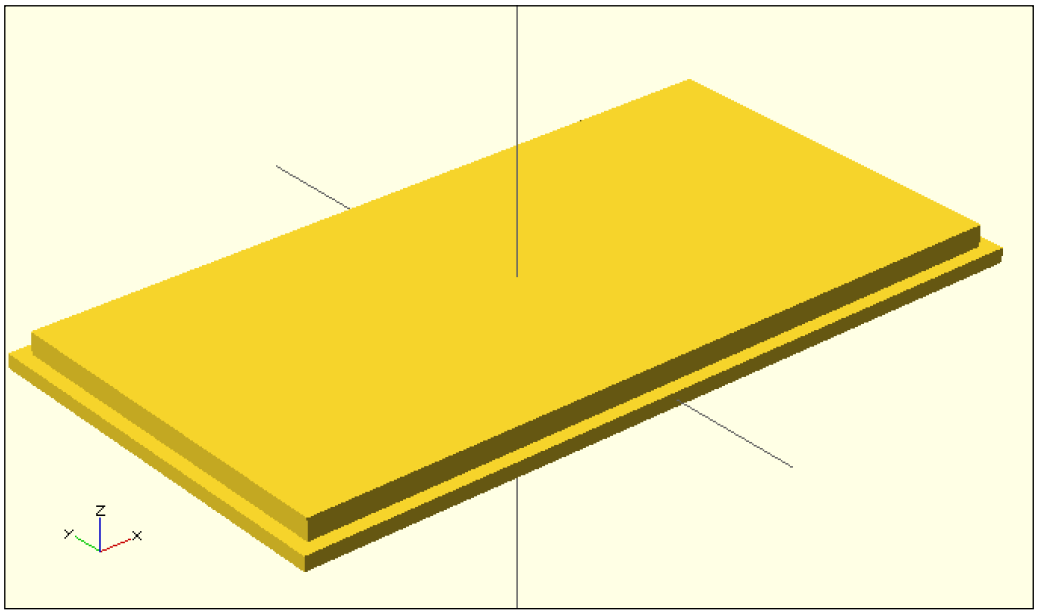
Череж состоит из разницы между двумя объектами куба. Я добавил еще один куб, чтобы проделать вырез сбоку, и два цилиндра для крепления платы.

Ниже приводится одна часть дела:



Я также разработал крышку, чтобы можно было закрыть корпус, используя те же принципы. Он основан на объединении двух кубических объектов, поэтому хорошо вписывается в основную часть корпуса.

Вот результат в OpenSCAD:



Когда ваш дизайн будет готов, вам нужно будет экспортировать его в специальный файловый формат, чтобы его можно было распечатать в 3D. Для этого перейдите в File|Export и экспортируйте файл в формат STL, который является форматом, необходимым для многих 3D-принтеров и производителей 3D-печати.

Код для этой части можно найти в репозитории GitHub проекта по следующему URL-адресу:
`https://github.com/openhomeautomation/arduino-home-automation/tree/master/chapter7`

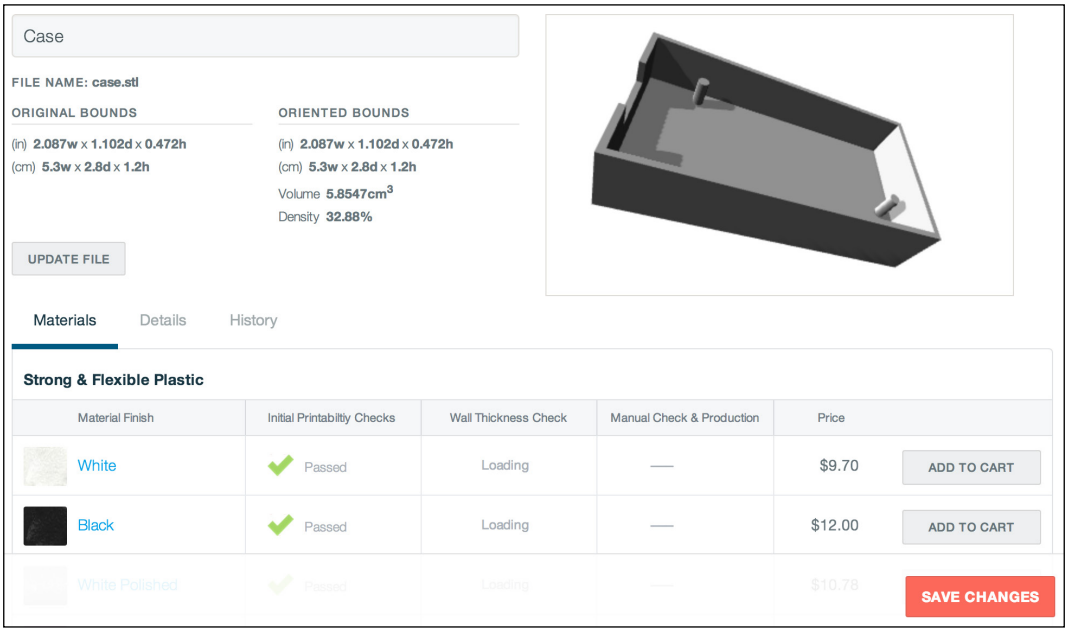
Как только ваш дизайн будет готов, вы должны его создать. Есть много способов сделать это. Конечно, один из вариантов - распечатать корпус на вашем собственном 3D-принтере, если он у вас есть. Если нет, вам придется воспользоваться одной из многих служб 3D-печати, доступных в Интернете. .

Рекомендую использовать Shapeways. У меня нет с ними коммерческих сделок, просто их интерфейс очень удобный, а цены конкурентоспособны. Для получения дополнительной информации вы можете перейти на следующий веб-адрес:

https://www.shapeways.com/

Вам будет предложено создать учетную запись и загрузить свой первый файл. Интерфейс Shapeways проанализирует файл и проверит, можно ли его напечатать без потенциальных проблем. Вы также получите оценку стоимости вашего дизайна.

Вот что у меня получилось с загруженным мной файлом:



Теперь вы можете выбрать свой материал и заказать; через несколько недель вы получите свой изделие. Затем вы можете собрать все: печатную плату, компоненты и корпус, напечатанный на 3D-принтере.

Существуют также другие услуги, которые вы можете использовать для 3D-печати вашего собственного корпуса для проекта. Еще я использовал Sculpteo:

http://www.sculpteo.com/

Вы также можете проверить свою местную фабрику, если она есть. Обычно они могут распечатать ваши изделия по гораздо более низкой цене. Чтобы узнать, есть ли поблизости подобные производства, где вы живете, проверьте официальный список по следующему URL-адресу:

http://wiki.fablab.is/wiki/Portal:Labs

Резюме

В этой главе мы увидели, как построить собственную домашнюю автоматическую систему на базе Arduino. Мы прошли три различных этапа процесса: создание прототипа вашей идеи на макете, разработка и изготовление печатной платы и, наконец, создание корпуса для проекта, напечатанного на 3D-принтере.

Давайте посмотрим на основные выводы из этой главы. Во-первых, мы точно определили, чего хотим достичь, и соответственно выбрали компоненты для проекта.

В качестве примера мы использовали маломощную схему Arduino, которая работает от батареи.

Затем мы построили прототип этого проекта на макете и буквально с нуля построили схему Arduino. Затем мы также занялись программной частью дизайна, используя библиотеку с низким энергопотреблением для Arduino.

После этого шага мы построили печатную плату на основе компонентов, которые использовали для прототипа макета. Мы увидели, как спроектировать печатную плату и как отправить ее производителю, чтобы он изготовил ее для вас. Наконец, мы увидели процесс проектирования и изготовления корпуса для вашего проекта, напечатанного на 3D-принтере.

Поскольку это конец книги, я хочу подвести итог тому, что мы сделали в этой книге. Мы впервые познакомились с основами домашней автоматики на платформе Arduino. Мы создавали проекты беспроводной домашней автоматики на базе Arduino с использованием таких технологий, как XBee, Wi-Fi и Bluetooth.

Затем мы подключили домашние проекты к Интернету с помощью облачного сервиса Xively. Идея подключения каждого объекта в вашем доме к Интернету называетсяИнтернетом вещей, и я верю, что в будущем он станет центральным элементом любой системы домашней автоматики.

Наконец, в последних двух главах книги мы увидели, как создать собственную домашнюю автоматику на базе Arduino, взломав существующие устройства или создав собственное устройство с нуля. Особенно в этой последней главе я действительно хотел показать вам различные шаги, которые вы можете применить к любому из ваших проектов: прототипирование с существующими компонентами, проектирование платы, ее изготовление и наконец изготовление индивидуального корпуса для вашей системы, напечатанного на 3D-принтере.

Я надеюсь, что проекты, представленные в этой книге, вызовут у вас желание пойти дальше и создать еще больше проектов для автоматизации вашего дома с помощью платформы Arduino. Возможности практически безграничны, и я действительно призываю вас экспериментировать и создавать собственные проекты на основе того, что вы узнали из этой книги. Экспериментируйте, творите и, самое главное, получайте от этого удовольствие!