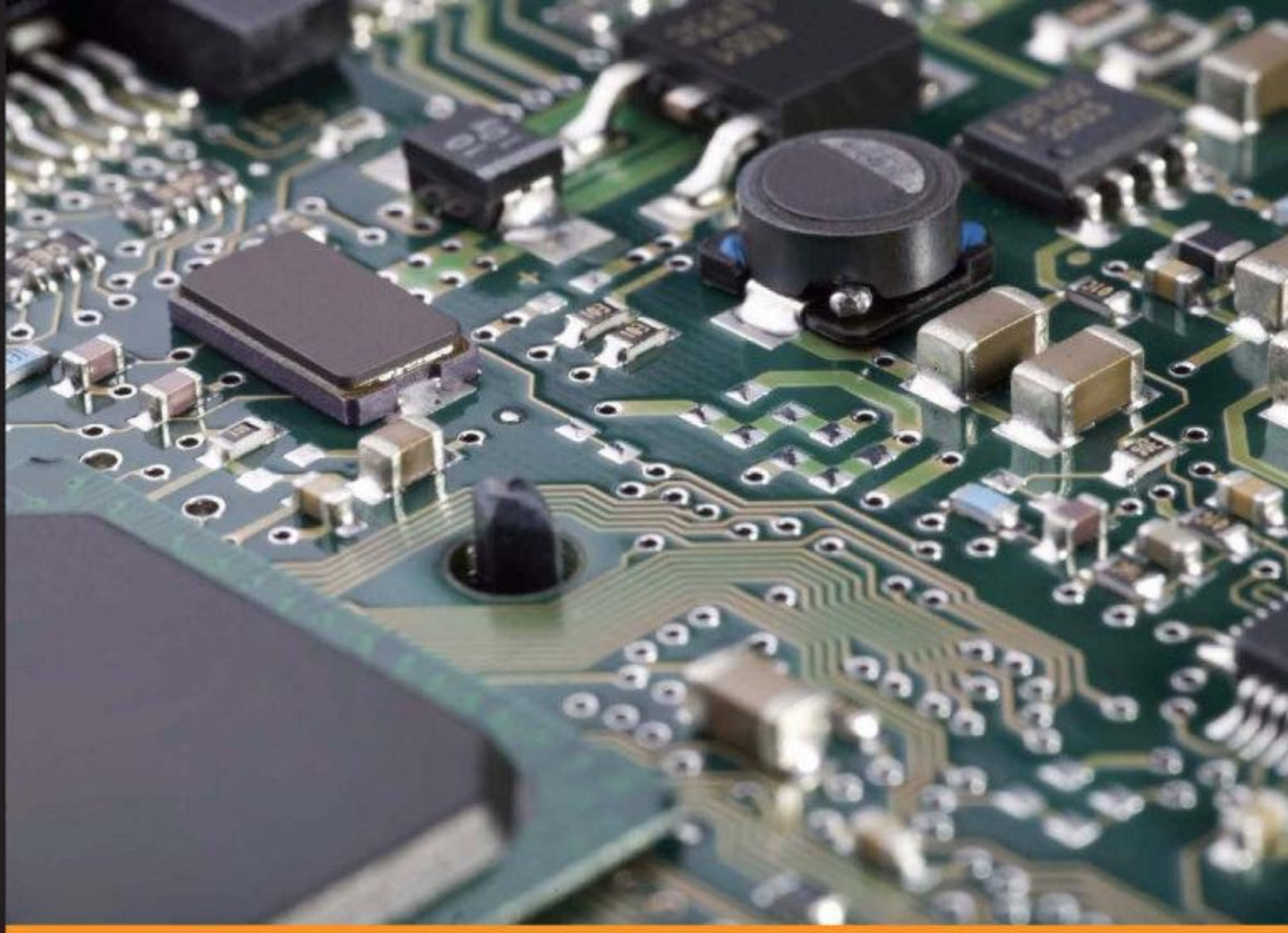


Community Experience Distilled

Arduino для секретных агентов

Превратите свое крошечное устройство Arduino в гаджет секретного агента и создайте ряд шпионских проектов с помощью этого практического руководства для хакеров

Marco Schwartz

Arduino для секретных агентов

Превратите свое крошечное устройство Arduino в гаджет секретного агента и создайте ряд шпионских проектов с помощью этого практического руководства для хакеров.

Marco Schwartz

об авторе

Marco Schwartz (Марко Шварц) - инженер-электрик, предприниматель и блогер. Он получил степень магистра в области электротехники и информатики в Supélec, Франция, и степень магистра в области микроэлектроники в Ecole Polytechnique Fédérale deLausanne (EPFL) в Швейцарии.

У него более пяти лет опыта работы в области электротехники. Интересы Марко сосредоточены вокруг электроники, домашней автоматики, платформ Arduino и Raspberry Pi, проектов оборудования с открытым исходным кодом и 3D-печати.

У него есть несколько веб-сайтов об Arduino, в том числе веб-сайт Open Home Automation, посвященный созданию систем домашней автоматики с использованием оборудования с открытым исходным кодом.

Марко написал еще одну книгу по домашней автоматике и Arduino под названием «Домашняя автоматика с Arduino: автоматизируйте свой дом с помощью оборудования с открытым исходным кодом». Он также написал книгу о том, как создавать проекты Интернета вещей с помощью Arduino, под названием « Things with the Arduino Yun, Packt Publishing.».

О рецензенте

Roberto Gallea, Кандидат наук, с 2007 г. занимается компьютерными науками. Работал в Университете Палермо, Италия. Он занимается исследованиями в таких областях, как получение медицинских изображений, мультимедиа и компьютерное зрение. В 2012 году он начал совершенствовать свои академические и личные проекты с использованием аналоговой и цифровой электроники, в частности, с использованием открытой аппаратной и программной платформы Arduino. Помимо академических интересов, он также занимается личными проектами, которые связаны с производством предметов ручной работы, включающих невидимую электронику, такую как музыкальные инструменты, мебель и светодиодные устройства. Он также сотрудничал с труппами современного танца в области цифрового сценического дизайна и дизайна костюмов.

Содержание

Предисловие

Платформа Arduino позволяет легко создавать электронные проекты в различных областях, таких как домашняя автоматика, Интернет вещей, носимые устройства и даже здравоохранение. Это также идеальная платформа для создания потрясающих проектов для секретных агентов, чем мы и собираемся заниматься в этой книге.

Используя мощь и простоту платформы Arduino, мы увидим, как создать несколько проектов, которые могут быть легко использованы любым честолюбивым секретным агентом.
От аудиорекордеров до GPS-трекеров - после прочтения этой книги вы сможете создать свой собственный набор инструментов секретного агента, используя платформу Arduino.

О чем эта книга

Глава 1, Простая система сигнализации с Arduino, посвящена созданию системы сигнализации, основанной на платформе Arduino, с датчиком движения и визуальной сигнализацией.

Глава 2 «Создание шпионского микрофона» посвящена созданию секретной записывающей системы, которая может записывать разговоры и шумы в комнате.

Глава 3 «Создание детектора клопов» посвящена созданию очень полезного устройства для любого секретного агента: детектора, проверяющего, есть ли в комнате устройства (клопы) других секретных агентов.

Глава 4, Контроль доступа с помощью датчика отпечатков пальцев, посвящена созданию системы контроля доступа с использованием вашего собственного отпечатка пальца.

Глава 5, Открытие замка с помощью SMS, посвящена созданию проекта, в котором секретный агент может открыть замок, просто отправив текстовое сообщение на устройство Arduino.

Глава 6, Посвящена созданию шпионской камеры, к которой можно получить доступ из любой точки мира и которая может записывать изображения в Dropbox при обнаружении движения изображения.

Глава 7 «Мониторинг секретных данных из любого места» посвящена изучению того, как тайно записывать любые данные и как регистрировать эти данные в облаке.

Глава 8 «Создание GPS-трекера с помощью Arduino» посвящена созданию одного из самых полезных устройств для секретного агента: GPS-трекера, который указывает его положение на карте в реальном времени.

Глава 9 «Создание робота-шпиона Arduino» посвящена созданию небольшого робота-наблюдателя, который сможет шпионить от вашего имени.

Что вам понадобится для этой книги

Во всей книге мы будем использовать платформу Arduino, поэтому вам определенно понадобится последняя версия программного обеспечения Arduino IDE.

Мы будем использовать широкий спектр плат, шилдов и компонентов Arduino.
Вы найдете все подробности об этих требованиях в соответствующих главах.

Для кого эта книга

Эта книга предназначена для тех, кто хочет создавать интересные проекты секретных агентов на платформе Arduino. Например, он предназначен для тех людей, которые уже имеют опыт использования платформы Arduino и хотят расширить свои знания, создав проекты для секретных агентов. Это также для людей, которые хотят узнать об электронике и программировании, поскольку Arduino - идеальная платформа для этого.

Соглашения

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

Кодовые слова в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, v-пути, фиктивные URL-адреса, пользовательский ввод и дескрипторы Twitter показаны следующим образом: «Кроме того, если thealarm_mode возвращается к false, нам нужно немедленно деактивировать сигнал тревоги. . "

Блок кода устанавливается следующим образом:

```
if (alarm_mode == false) {

    // No tone & LED off
    noTone(alarm_pin);
    digitalWrite(led_pin, LOW);
  }
```

Любой ввод или вывод командной строки записывается следующим образом:

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 25" -o
"output_http.so -p 8080 -w /www/webcam" &
```

Новые термины и **важные слова** выделены жирным шрифтом. Слова, которые вы видите на экране, например, в меню или диалоговых окнах, появляются в тексте следующим образом: «Теперь в параметрах приложения вам нужны две вещи: ключ приложения и секрет приложения».

Предупреждения или важные примечания отображаются в таком поле.

Советы и хитрости выглядят так
.

1

Простая система сигнализации с Arduino

Я хочу начать эту книгу с простого проекта, который захочет иметь любой секретный агент, простой системы сигнализации, которая будет активироваться всякий раз, когда датчик обнаруживает движение. Эта простая система не только забавна в изготовлении, но также поможет нам изучить основы программирования и электроники Arduino, которые мы будем использовать во всей этой книге.

По сути, это будет простая сигнализация (зуммер, который издает звук, плюс красный светодиод) в сочетании с датчиком движения. Пользователь также сможет отключить звук, нажав кнопку.

В этой главе мы собираемся сделать следующее:

- Во-первых, мы собираемся посмотреть, каковы требования к этому проекту с точки зрения аппаратного и программного обеспечения
- Затем мы увидим, как собрать аппаратные части для этого проекта.
- После этого мы настроим нашу систему с помощью Arduino IDE.

Компоненты и программы

Во-первых, давайте посмотрим, какие компоненты необходимы для этого проекта. Поскольку это первая глава книги, мы потратим здесь немного больше времени, чтобы подробно описать различные компоненты, поскольку это компоненты, которые мы будем использовать во всей книге.

Первым компонентом, который будет центральным в проекте, является плата Arduino Uno:

В нескольких главах этой книги он будет «мозгом» проектов, которые мы будем делать. Во всех проектах я буду использовать официальную плату Arduino Uno R3. Однако вы можете использовать эквивалентную плату другого производителя или другую плату Arduino, например плату Arduino Mega.

Еще одним важным компонентом нашей системы сигнализации будет зуммер:

Это очень простой компонент, который используется для создания простых звуков с помощью Arduino. Вы не можете проигрывать с ним MP3, но для системы охранной сигнализации он просто идеален. Вы, конечно, можете использовать любой доступный зуммер; цель - просто издать звук.

После этого нам понадобится детектор движения:

Здесь я использовал очень простой детектор движения PIR. Этот датчик будет измерять инфракрасный (ИК) свет, излучаемый движущимися объектами в его поле зрения, например, перемещающимися людьми. Интерфейс с Arduino действительно прост и довольно дешев. Вы можете использовать любой бренд для этого датчика; ему просто нужен уровень напряжения 5 В, чтобы быть совместимым с платой Arduino Uno.

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

- Arduino Uno
- Buzzer
- PIR
- LED
- 330 Ом резистор
- Кнопка
- 1kОм резистор
- Макетная плата
- Проволочные перемычки

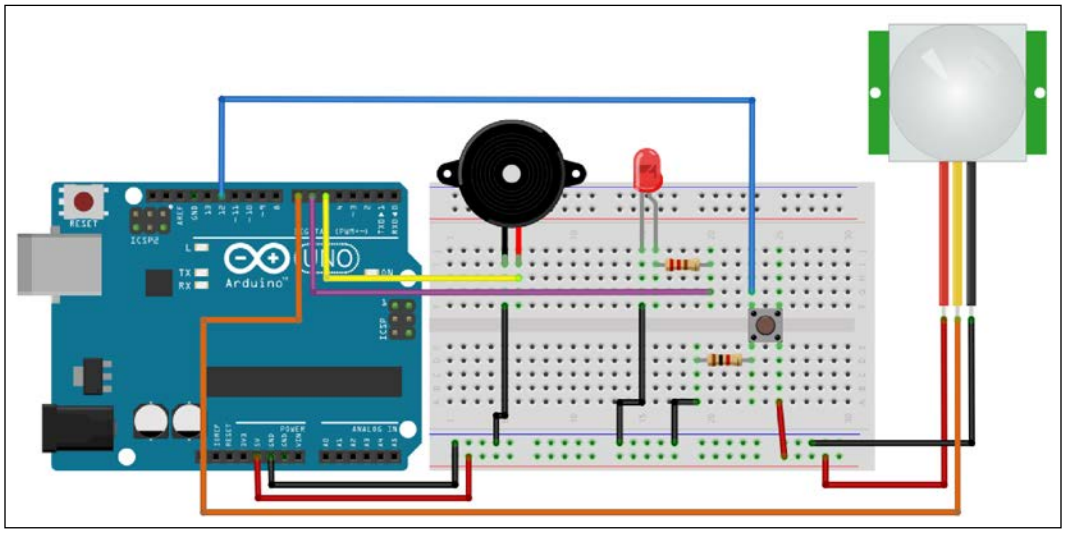
Что касается программного обеспечения, единственное, что нам понадобится в первой главе, - это последняя версия IDE Arduino, которую вы можете загрузить по следующему URL-адресу: https://www.arduino.cc/en/main/software.

Обратите внимание, что мы собираемся использовать IDE Arduino во всех проектах этой книги, поэтому обязательно установите последнюю версию.

Аппаратное обеспечение

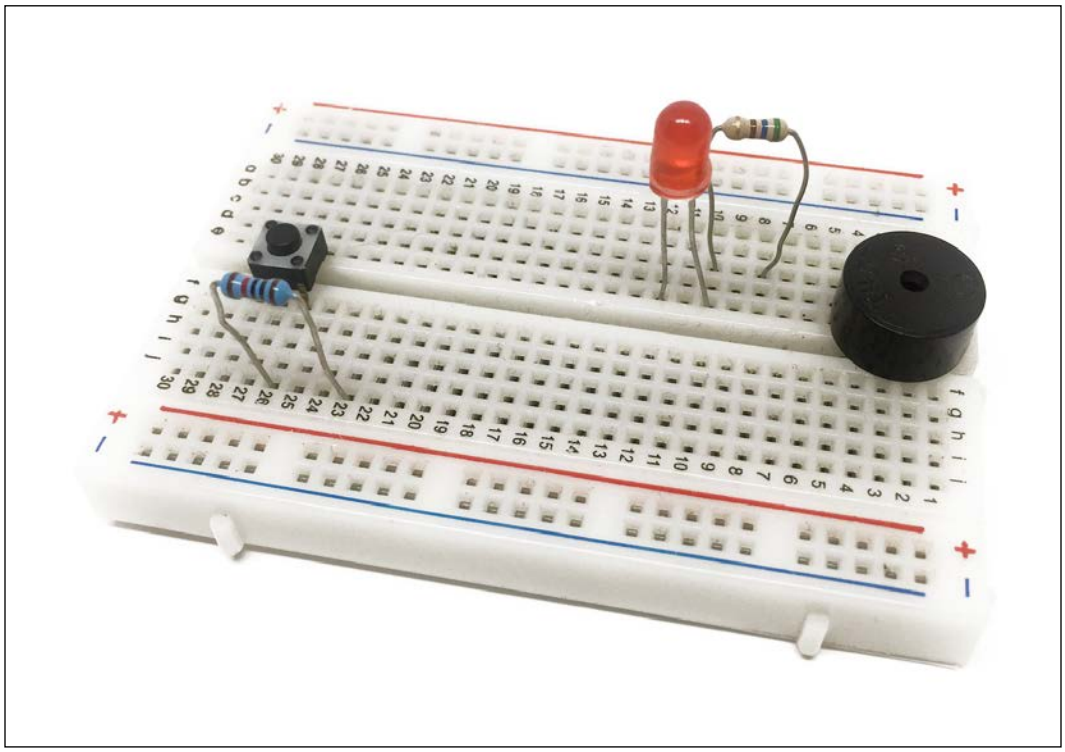
Сейчас мы собираемся собрать устройство для этого проекта. Поскольку это первый проект этой книги, он будет довольно простым. Однако компонентов довольно много, поэтому обязательно выполните все действия.

Вот схема, которая поможет вам в этом процессе:



Начнем с размещения всех компонентов на плате. Сначала поместите зуммер, кнопку и светодиод на плату, как показано на схеме. Затем подключите резистор 330 Ом последовательно с анодом светодиода (самый длинный вывод) и подключите резистор 1 кОм к одному выводу кнопки.

Вот как это должно выглядеть на этом этапе:



Теперь мы собираемся подключить каждый компонент к плате Arduino.

Начнем с питания. Подключите вывод 5V платы Arduino к одной красной шине питания макета, а контакт GND платы Arduino к одной синей шине питания макета.

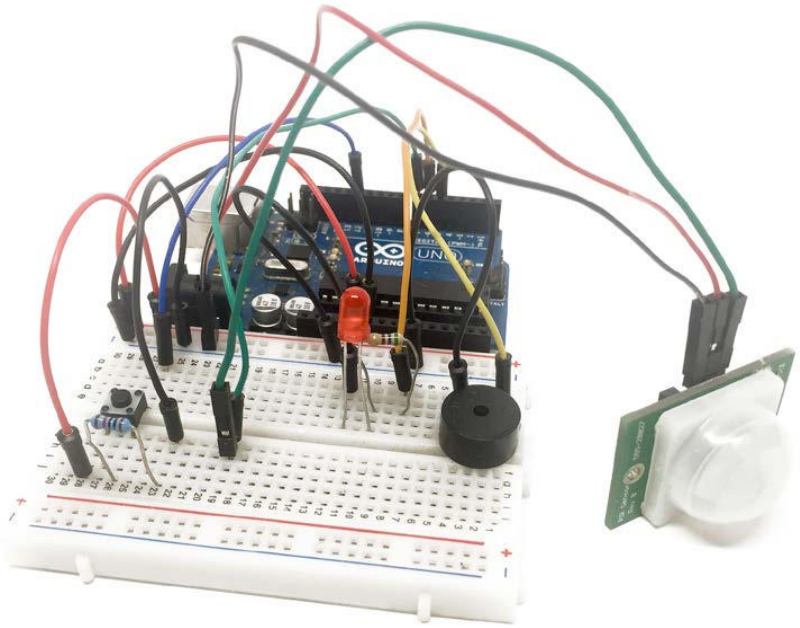
Затем мы собираемся подключить зуммер. Подключите один контакт зуммера к контакту №5 платы Arduino, а другой контакт - к синей шине питания макета.

После этого подключим светодиод. Подключите свободный вывод резистора к выводу номер 6 на плате Arduino, а свободный вывод светодиода (катод) к земле через синюю шину питания.

Давайте также подключим кнопку к нашей плате Arduino. Обратитесь к схеме, чтобы быть уверенным в подключении, поскольку она немного сложнее. По сути, вам нужно подключить свободный контакт резистора к земле и подключить контакт, который подключен к кнопке, к контакту 5V через красную шину питания. Наконец, подключите другую сторону кнопки к контакту 12 платы Arduino.

Наконец, давайте подключим датчик движения PIR к плате Arduino. Подключите контакт VCC датчика движения к красной шине питания, а контакт GND - к синей шине питания. Наконец, подключите вывод SIG (или вывод OUT) к выводу 7 Arduino.

Вот окончательный результат:



Если ваш проект похож на этот рисунок, поздравляю, вы только что собрали свой первый проект секретного агента! Теперь вы можете перейти к следующему разделу.

Настройка охранной сигнализации

Теперь, когда устройство для нашего проекта готово, мы можем записать код для проекта, чтобы у нас была работоспособная система сигнализации. Цель состоит в том, чтобы заставить зуммер издавать звук при обнаружении движения, а светодиод - мигать. Однако всякий раз, когда нажимается кнопка, зуммер должен отключаться.

.

Вот полный код этого проекта:

```
// Код для простой сигнализации

// Pins
const int alarm_pin = 5;
const int led_pin = 6;
const int motion_pin = 7;
const int button_pin = 12;

// Аварийная сигнализация
boolean alarm_mode = false;

// Переменные для мигающего светодиода
int ledState = LOW;
long previousMillis = 0;
              long interval = 100;  // Интервал мигания (миллисекунды)

void setup()
{
  // Set pins to output
  pinMode(led_pin,OUTPUT);
  pinMode(alarm_pin,OUTPUT);

  // Set button pin to input
  pinMode(button_pin, INPUT);

  // Подождите, прежде чем включить зуммер
  delay(5000);
}

void loop()
{
  // Обнаружено движение?
  if (digitalRead(motion_pin)) {
    alarm_mode = true;
  }

  // Если включен режим будильника,замигает светодиод,и запищит зуммер.
  if (alarm_mode){
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (ledState == LOW)
```

```
        ledState = HIGH;
      else
        ledState = LOW;
    // Включить светодиод
    digitalWrite(led_pin, ledState);
    }
    tone(alarm_pin,1000);
  }

  // If alarm is off
  if (alarm_mode == false) {

    // Нет звука и светодиод не горит
    noTone(alarm_pin);
    digitalWrite(led_pin, LOW);
  }

  // Если кнопка нажата, отключить будильник
  int button_state = digitalRead(button_pin);
  if (button_state) {alarm_mode = false;}
}
```

Скачивание примера кода

Пример этого кода можно взять из прилагаемой папки с примерами кодов

Теперь мы рассмотрим более подробно различные части кода. Он начинается с объявления, к каким пинам платы Arduino подключены зуммер,LED,датчик PIR и кнопка:

```
const int alarm_pin = 5;//к пину 5 подключим зуммер
const int led_pin = 6; // к пину 6 подключим LED
const int motion_pin = 7;// к пину 7 подключим PIR датчик
const int button_pin = 12; //к пину 12 - кнопку
```

После этого в функции скетча setup () мы объявляем эти выводы как входы или выходы, как показано ниже:

```
// Установите контакты как выходы
pinMode(led_pin,OUTPUT);
pinMode(alarm_pin,OUTPUT);

// Установить кнопку как вход
pinMode(button_pin, INPUT);
```

Затем в функции скетча loop () проверяем, включена ли тревога, проверяя состояние датчика движения:

```
if (digitalRead(motion_pin)) {
  alarm_mode = true;
}
```

Обратите внимание, что если мы обнаруживаем какое-либо движение, мы немедленно устанавливаем для параметра alarm_modevariable значение true (истина). Мы увидим, как код использует эту переменную прямо сейчас.

Теперь, если alarm_modevariable имеет значение true, мы должны включить будильник, заставить зуммер издавать звук, а также зажечь светодиод. Это делается с помощью следующего фрагмента кода:

```
if (alarm_mode){
    unsigned long currentMillis = millis();
    if(currentMillis - previousMillis > interval) {
      previousMillis = currentMillis;
      if (ledState == LOW)
        ledState = HIGH;
      else
        ledState = LOW;
    // Switch the LED
    digitalWrite(led_pin, ledState);
    }
    tone(alarm_pin,1000);
  }
```

Кроме того если alarm_mode возвращает false, нам нужно немедленно деактивировать тревогу, остановив воспроизведение звука и выключив светодиод. Это делается с помощью следующего кода:

```
if (alarm_mode == false) {

    // No tone & LED off
    noTone(alarm_pin);
    digitalWrite(led_pin, LOW);
  }
```

Наконец, мы постоянно считываем состояние кнопки. Если кнопка нажата, мы сразу же отключим будильник:

```
int button_state = digitalRead(button_pin);
if (button_state) {alarm_mode = false;}
```

Обычно мы должны позаботиться об эффекте дребезга кнопки, чтобы убедиться, что у нас нет ошибочных показаний при нажатии кнопки. Однако здесь мы заботимся только о фактическом нажатии кнопки, поэтому нам не нужно добавлять дополнительный код устранения дребезга кнопки.

Теперь, когда мы написали код проекта, пора перейти к самой захватывающей части главы: тестирование системы охранной сигнализации!

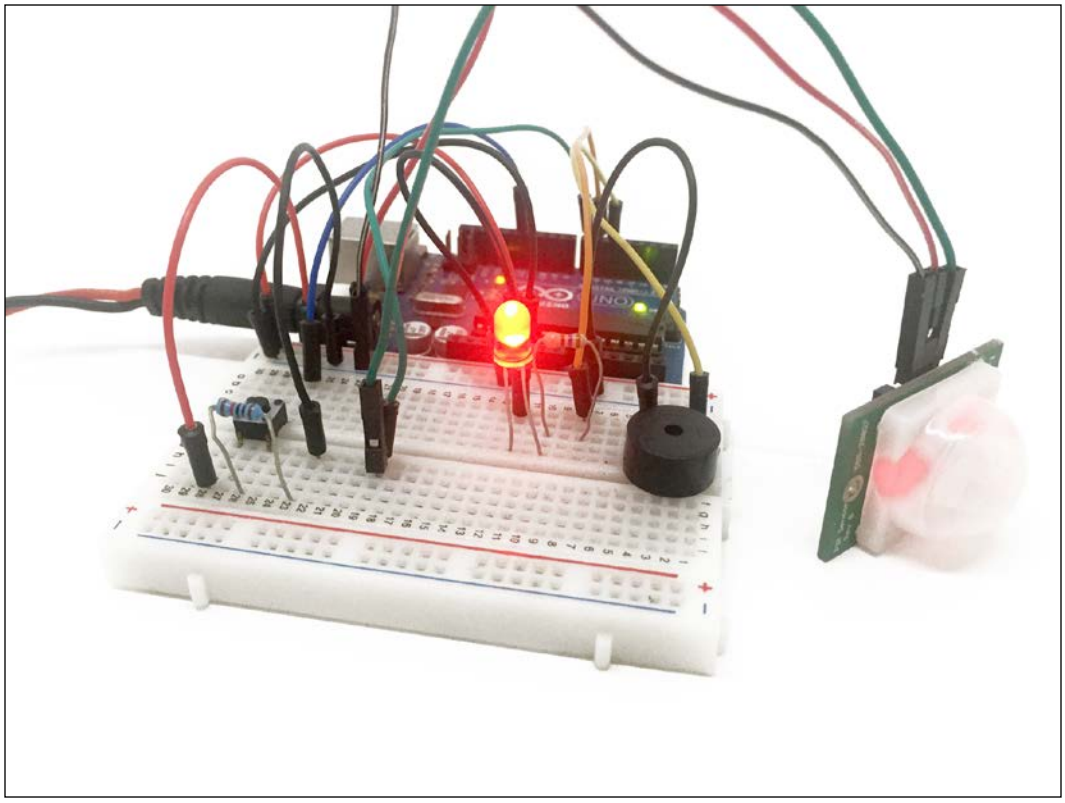
Тестирование охранной сигнализации

Теперь мы готовы протестировать нашу простую систему сигнализации. Просто возьмите код для этого проекта (из папки с кодами - chapter01) и вставьте его в Arduino IDE.

IВ среде IDE выберите вашу плату (например, Arduino Uno), а также соответствующий последовательный порт.

Теперь вы можете загрузить код в свою плату. Как только это будет сделано, просто проведите рукой перед датчиком движения PIR; должна сработать сигнализация . Затем просто нажмите кнопку, чтобы выключить звук.

На рисунке представлен результат нашей работы по созданию простой охранной сигнализации:



Если все у вас работает, как и ожидалось, поздравляем, вы только что создали свой первый проект секретного агента: простую систему сигнализации на базе Arduino!

Если на этом этапе проект не заработает, вы можете проверить несколько вещей.
Во-первых, еще раз проверьте правильность подключения компонентов;есть ли контакты проводников с компонентами.

Кроме того, вы должны убедиться, что когда вы проводите рукой перед датчиком PIR, он горит. Если это не так, скорее всего, ваш датчик движения PIR неисправен и его необходимо заменить.

Резюме

В этой первой главе мы создали простую сигнализацию на основе Arduino с использованием всего лишь нескольких компонентов.

Есть несколько способов пойти дальше и улучшить этот проект. Вы можете добавить в проект больше функций, просто добавив больше строк в код. Например, вы можете добавить таймер, чтобы будильник срабатывал только через определенное время, или вы можете создать режим, в котором нажатие кнопки фактически активирует или деактивирует режим будильника.

В следующей главе мы собираемся создать еще один проект, очень полезный для секретных агентов: устройство записи звука на базе Arduino!

2

Создание шпионского микрофона

В этой главе мы собираемся построить очень полезное устройство для любого секретного агента: шпионский микрофон. Проект будет основан на Arduino, с электретным микрофоном с встроенным усилителем и SD-картой

Ниже приведены шаги, которые мы собираемся предпринять для создания этого проекта:

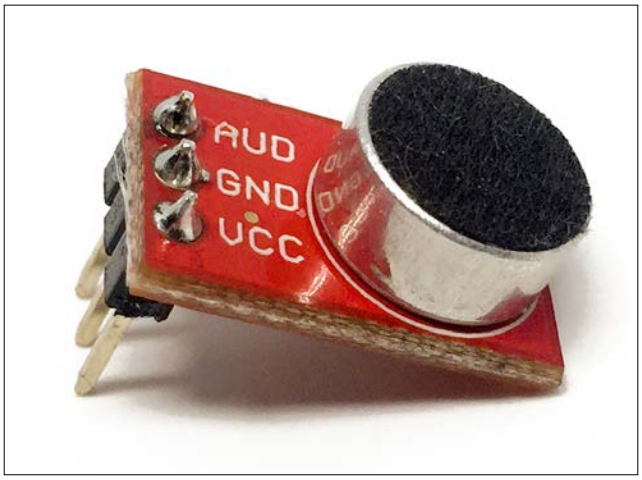
- Мы научимся производить запись в течение определенного времени, которое может быть настроено пользователем.
- Затем записанный аудиофайл будет записан на SD-карту и будет доступен с любого компьютера.
- Перед этим мы протестируем все компоненты проекта индивидуально.

Ну что,начнем!

Компоненты и программы

Давайте сначала посмотрим, какие компоненты необходимы для этого проекта. Как обычно мы будем использовать плату Arduino Uno в качестве «мозга» проекта.

Затем нам понадобится микрофон. Я использовал простой электретный микрофон SparkFun со встроенным усилителем, как показано на следующем рисунке:



Здесь самое главное, чтобы микрофон был с усилителем. Например, SparkFun усиливает в 100 раз, что позволяет Arduino Uno записывать обычные уровни звука (например, голоса).

Еще вам понадобится карта microSD с переходником:

Есть много способов записи данных на SD-карту с Arduino. Самым простым решением, которое я выбрал здесь, является использование шилда. У меня был Ethernet shield, корторый имеет встроенный картридер microSD.

Вы, конечно, можете использовать любой шилд или даже плату с устройством чтения карт памяти microSD .

Вам также понадобится макетная плата и несколько перемычек для выполнения необходимых соединений.

Наконец, ниже приводится список всех компонентов, которые мы будем использовать в этом проекте:

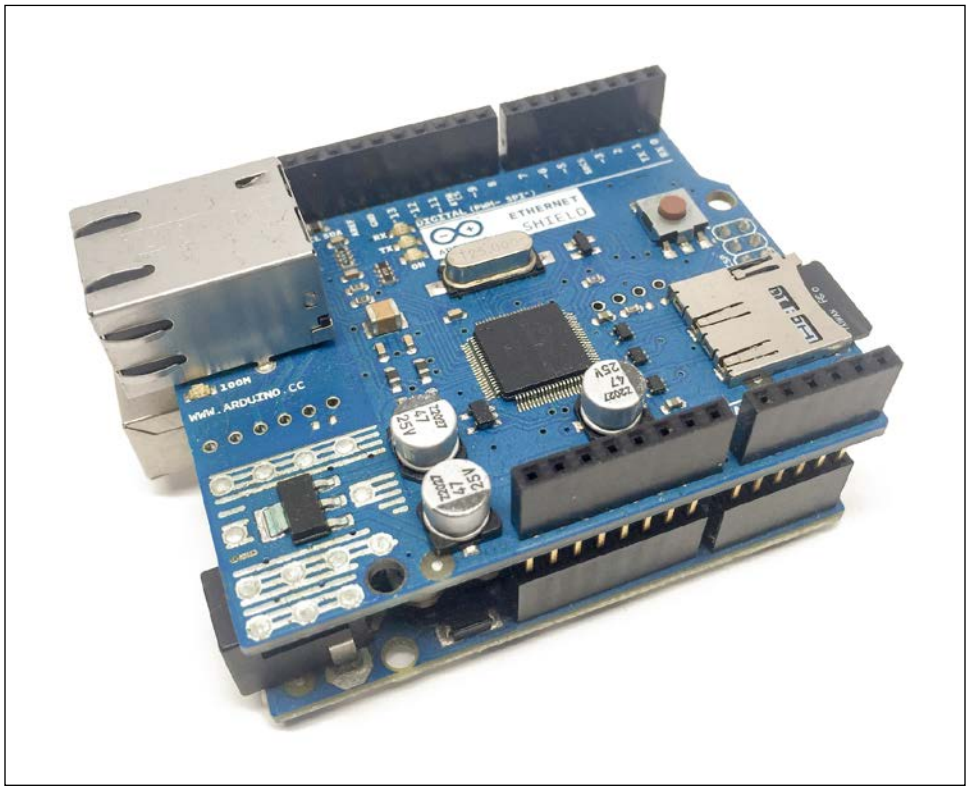
- Arduino Uno
- Arduino Ethernet шилд
- Электретный микрофон
- Макетная плата
- Перемычки

Что касается программного обеспечения, вам понадобится специальная версия библиотеки SD-карты под названием SdFat. Мы не можем использовать здесь обычную SD-библиотеку Arduino, так как мы будем выполнять некоторые действительно быстрые операции записи на SD-карту, которые не могут быть обработаны SD-библиотекой, поставляемой с программным обеспечением Arduino. Вы можете скачать эту библиотеку с https://github.com/ greiman / SdFat (имеется в прилагаемой папке).

Использование SD-карты

Первое, что мы собираемся сделать в этом проекте, - это проверить, можем ли мы получить доступ к SD-карте. Это гарантирует, что мы не столкнемся с проблемами, связанными с SD-картой, позже в проекте.

Это изображение Ethernet Shield, который я использовал, с картой microSD, установленной справа:



Давайте теперь посмотрим на код, который мы будем использовать для проверки функциональности SD-карты. Ниже приводится полный код этого раздела:

```
// Включить SD-библиотеку
#include <SPI.h>
#include <SD.h>

// Настройте переменные с помощью функций библиотеки служебных программ SD:
Sd2Card card;
SdVolume volume;
SdFile root;

// измените его, чтобы он соответствовал вашему SD-шилду или модулю;
// шилд Arduino Ethernet: контакт 4
// шилды и модули SD Adafruit: контакт 10
// Sparkfun SD шилд: pin 8
```

```
const int chipSelect = 4;

void setup()
{
  // Откройте последовательную связь и дождитесь открытия порта:
  Serial.begin(115200);
  while (!Serial) {
   ; // дождитесь подключения последовательного порта. Требуется только
                                                              //для Леонардо
  }

  Serial.print("\nInitializing SD card...");

  // мы будем использовать код инициализации из служебных библиотек
  // так как мы просто проверяем, работает ли карта!
  if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your
shield or module?");
    return;
  } else {
    Serial.println("Wiring is correct and a card is present.");
  }

  // print the type of card
  Serial.print("\nCard type: ");
  switch (card.type()) {
    case SD_CARD_TYPE_SD1:
      Serial.println("SD1");
      break;
    case SD_CARD_TYPE_SD2:
      Serial.println("SD2");
      break;
    case SD_CARD_TYPE_SDHC:
      Serial.println("SDHC");
      break;
    default:
      Serial.println("Unknown");
  }

  // Теперь попробуем открыть раздел «том» / «раздел» - он должен
  //быть FAT16 or FAT32
```

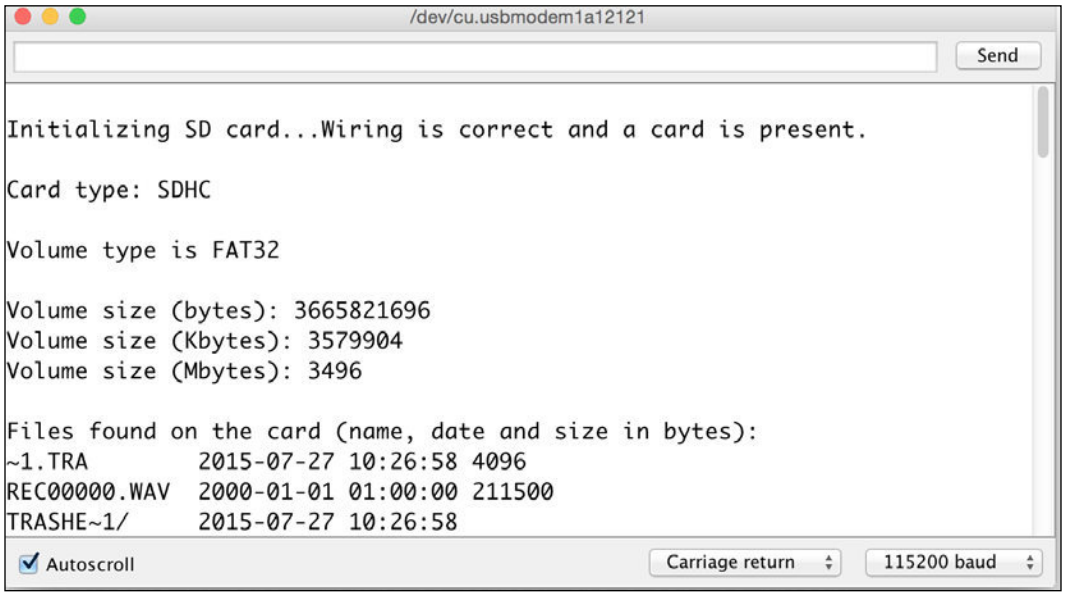
```
  if (!volume.init(card)) {
    Serial.println("Could not find FAT16/FAT32 partition.\nMake sure
you've formatted the card");
    return;
  }


  // распечатать тип и размер первого тома типа FAT
  uint32_t volumesize;
  Serial.print("\nVolume type is FAT");
  Serial.println(volume.fatType(), DEC);
  Serial.println();

  volumesize = volume.blocksPerCluster();    // кластеры
collections of blocks
   volumesize *= volume.clusterCount();       // у нас будет много
                              кластеров
     volumesize *= 512;                            // Блоки SD-карты
                                        всегда имеют размер 512 байт
  Serial.print("Volume size (bytes): ");
  Serial.println(volumesize);
  Serial.print("Volume size (Kbytes): ");
  volumesize /= 1024;
  Serial.println(volumesize);
  Serial.print("Volume size (Mbytes): ");
  volumesize /= 1024;
  Serial.println(volumesize);


  Serial.println("\nFiles found on the card (name, date and size in
bytes): ");
  root.openRoot(volume);

  // перечислить все файлы на карте с датой и размером
  root.ls(LS_R | LS_DATE | LS_SIZE);
}


void loop(void) {

}
```

Этот код проверяет множество вещей на SD-карте, таких как формат файла и доступное пространство, а также перечисляет все файлы, имеющиеся на SD-карте. Однако нас действительно интересует, может ли SD-карта считываться платой Arduino. Это делается с помощью следующего фрагмента кода:

```
if (!card.init(SPI_HALF_SPEED, chipSelect)) {
    Serial.println("initialization failed. Things to check:");
    Serial.println("* is a card inserted?");
    Serial.println("* is your wiring correct?");
    Serial.println("* did you change the chipSelect pin to match your
shield or module?");
    return;
  } else {
    Serial.println("Wiring is correct and a card is present.");
  }
```

Теперь давайте протестируем код. Вы можете просто скопировать этот код (имеется в прилагаемой папке с кодами - chapter02) и вставить его в IDE Arduino.

Затем загрузите его на плату Arduino Uno и откройте монитор последовательного порта. Убедитесь, что скорость последовательного порта установлена на 115200 бит / с. Вот что вы увидите:



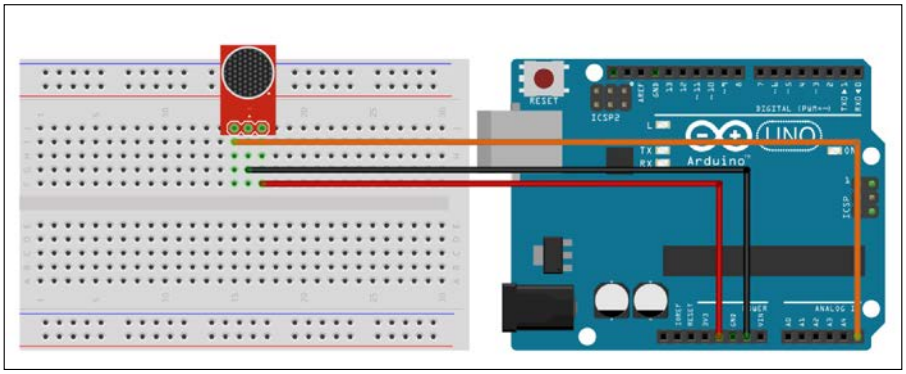
Если вы это видите, поздравляю, ваша SD-карта и кардридер правильно настроены и готовы к размещению шпионских аудиозаписей!

Тестирование микрофона

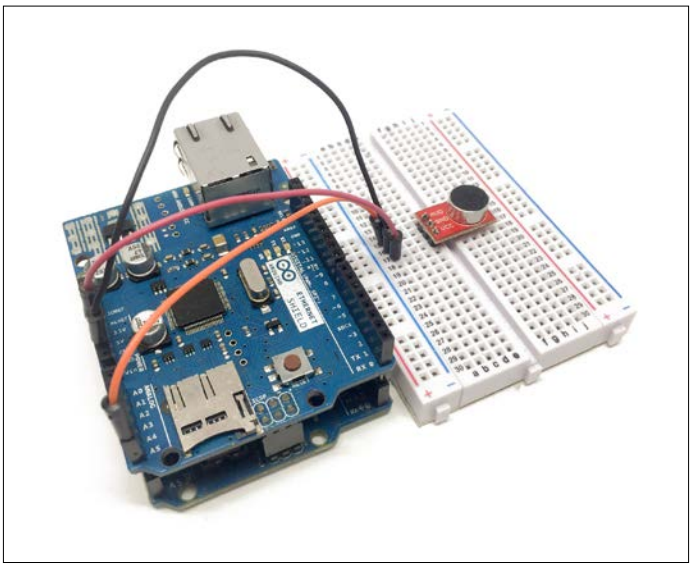
Теперь мы собираемся убедиться, что микрофон работает правильно, и, в частности, проверить, может ли он записывать,например, уровни голоса. У меня возникла проблема, когда я тестировал прототип этого проекта с микрофоном, который не усиливал;Я просто ничего не слышал при записи.

Первым делом необходимо подключить микрофон к плате Arduino. Есть 3 контакта для подключения микрофона: VCC, GND и AUD. Подключите VCC к контакту Arduino 5V, GND к контакту Arduino GND и AUD к аналоговому контакту Arduino A5.

Следующая схема поможет вам:



Вот изображение окончательного результата:



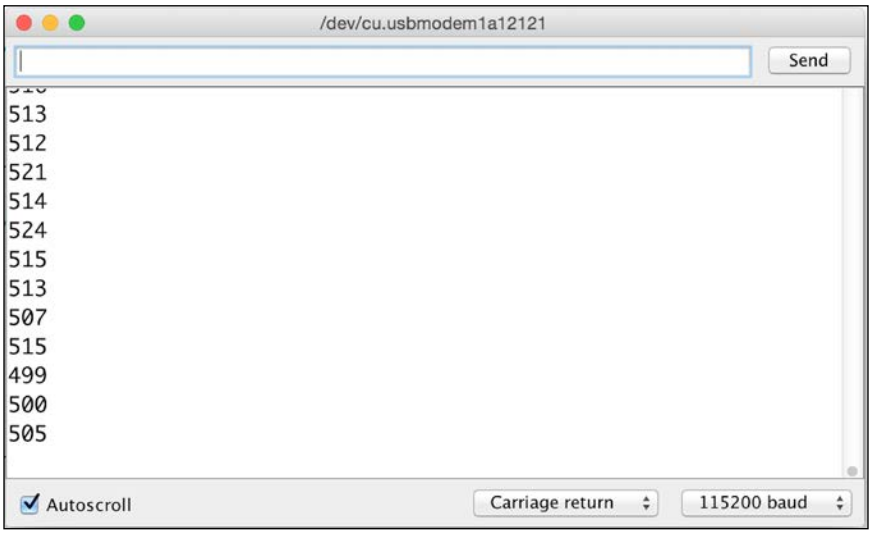
Теперь мы воспользуемся очень простым скетчем, чтобы считать сигнал с микрофона и распечатать его на последовательном мониторе:

```
// Тест микрофона

void setup() {

  // Start Serial
  Serial.begin(115200);
}

void loop() {

  // Считать вход на аналоговом выводе 5:
  int sensorValue = analogRead(A5);

  // Распечатать прочитанное вами значение:
  Serial.println(sensorValue);
  delay(1);        // задержка между чтениями для стабильности
}
```

Этот скетч, по сути, непрерывно считывает данные с контакта A5, к которому подключен микрофон, и печатает их на последовательном мониторе.

Теперь скопируйте и вставьте этот скетч в IDE Arduino и загрузите его на плату. Также откройте последовательный монитор.

Результат на последовательном мониторе:



Глядя на монитор последовательного порта, говорите в микрофон. Вы должны сразу увидеть некоторые вариации сигнала, считываемого платой. Это означает, что ваш голос записывается микрофоном и усиления достаточно, чтобы микрофон записал нормальный уровень голоса.

Создание шпионского микрофона

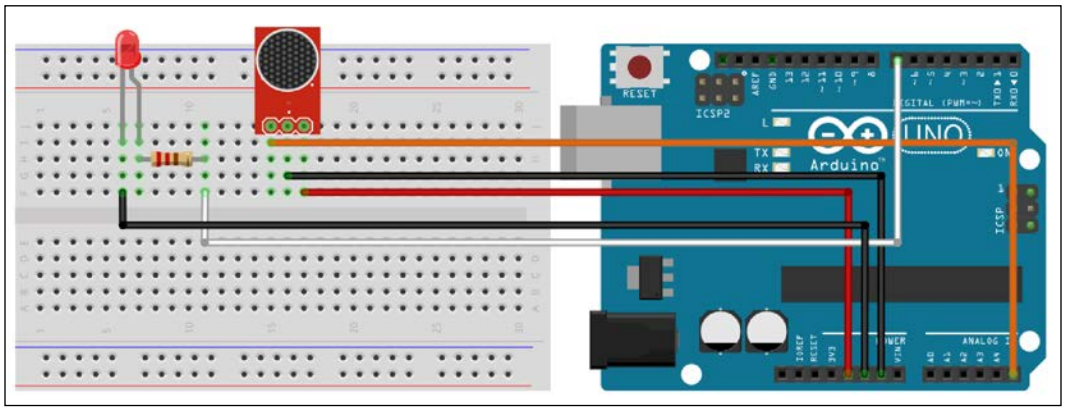
В этом разделе мы собираемся собрать все воедино и создать наш шпионский микрофон.

Оборудование для проекта почти готово, если вы следовали предыдущему разделу.

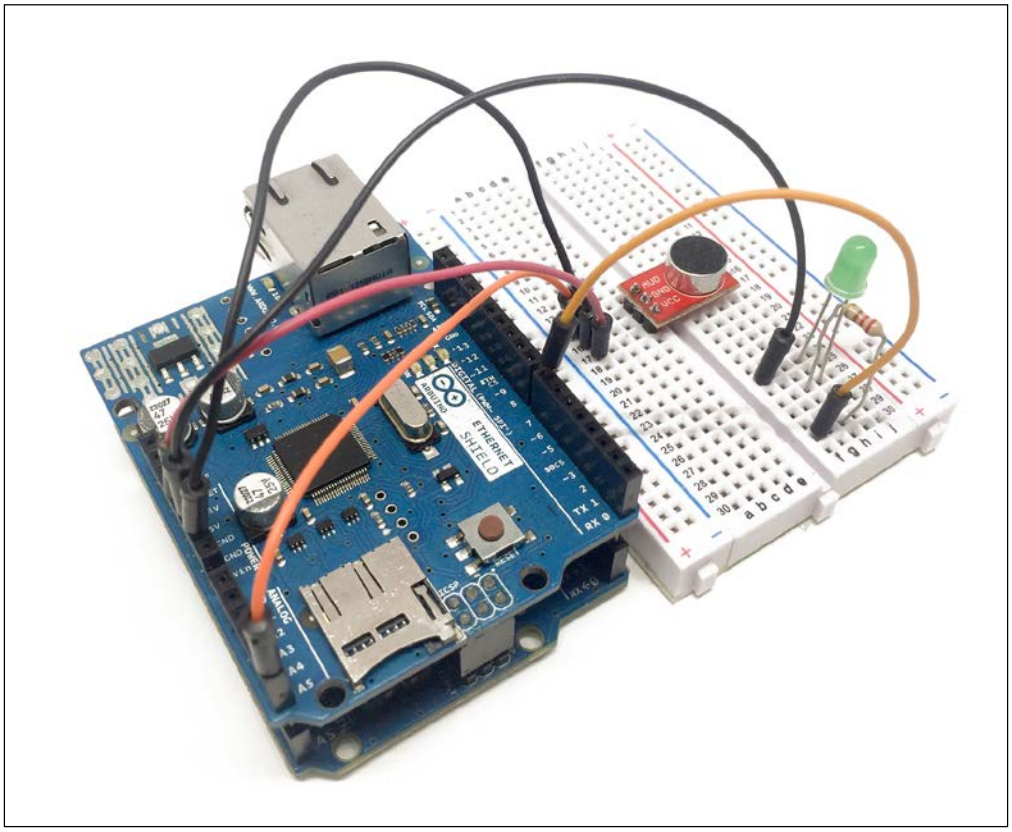
Вам просто нужно снова вставить SD-карту в ридер.

Я также добавил светодиод на контакт 7, чтобы знать, когда идет запись. Если вы хотите сделать то же самое, вам просто понадобится светодиод и резистор на 330 Ом. Конечно, удалите этот светодиод, если вы действительно хотите использовать устройство в качестве шпионского микрофона, иначе оно будет замечено.

Схема, которая поможет вам, выглядит следующим образом:



Ниже приводится изображение полностью собранного проекта.:



Теперь мы рассмотрим детали кода проекта. По сути, мы хотим, чтобы устройство записывало звук с микрофона в течение определенного времени, а затем остановливало запись.

Поскольку код длинный и сложный, мы рассмотрим здесь только самые важные части.

Первым шагом является включение библиотеки SdFatlibrary:

```
#include <SdFat.h>
```

Затем мы присвоим имена, которые необходимы для записи на SD-карту:

```
SdFat sd;
SdFile rec;
```

После этого мы определим, на каком контакте должен быть дополнительный светодиод записи:

```
const int ledStart = 7;
```

Мы также определим переменную для внутреннего счетчика проекта:

```
byte recordingEnded = false;
unsigned int counter;
unsigned int initial_count;
unsigned int maxCount = 10 * 1000; // 10 Секунд
```

Обратите внимание, что здесь вам нужно будет изменить переменную maxCount в соответствии со временем, в течение которого устройство должно записывать. Здесь я просто использовал 10 секунд по умолчанию в качестве теста.

Затем мы инициализируем АЦП (аналого-цифровой преобразователь):

```
Setup_timer2();
Setup_ADC();
```

После этого мы инициализируем SD-карту, а также мигаем светодиодом в случае успеха:

```
if (sd.begin(chipSelect, SPI_FULL_SPEED)) {
  for (int dloop = 0; dloop < 4; dloop++) {
    digitalWrite(ledStart,!digitalRead(ledStart));
    delay(100);
  }
}
```

Затем мы фактически начнем запись с помощью следующей функции:

```
StartRec();
```

Мы также инициализируем счетчик следующей строкой кода:

```
initial_count = millis();
```

Затем в функции скетча loop () мы обновим счетчик, который помогает отслеживать истекшее время:

```
counter = millis() - initial_count;
```

Продолжая использовать функцию loop (), мы фактически остановим запись, если достигнем максимального количества времени, которое мы определили ранее:

```
if (counter > maxCount && !recordingEnded) {
  recordingEnded = true;
  StopRec();
}
```

Обратите внимание, что весь код этого раздела можно найти в репозитории GitHub книги по адресу https://github.com/marcoschwartz/arduino-secret-agents.

Запись на **SD-**карту

В последнем разделе главы мы фактически протестируем проект и запишем аудио.

Сначала скопируйте весь код (chapter02 в папке с кодами) и вставьте его в IDE Arduino. Скомпилируйте его и загрузите на плату Arduino. Обратите внимание, что как только вы это сделаете, устройство начнет запись звука. Если вы подключили дополнительный светодиод к контакту 7, этот светодиод также должен гореть во время фазы записи.

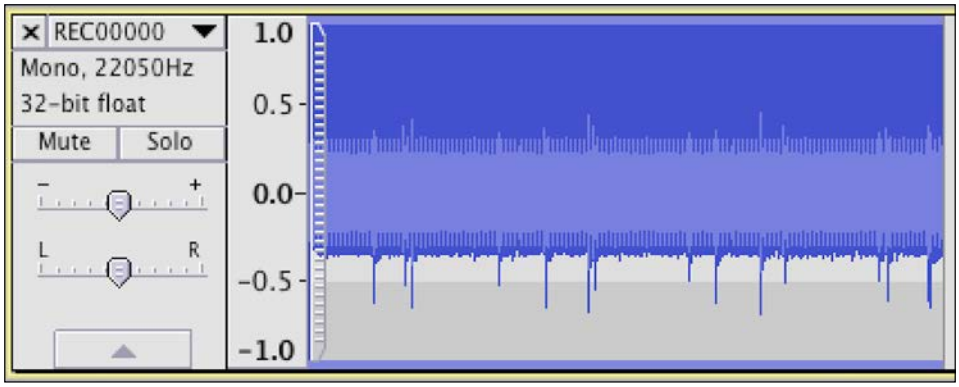
Теперь вы можете немного поговорить или воспроизвести свою любимую песню, чтобы убедиться, что микрофон записывает реальный звук.

Затем, по истечении времени, указанного в коде, остановите проект, отключив питание. Затем извлеките SD-карту и вставьте ее в свой компьютер.

На вашем компьютере перейдите к SD-карте, и вы увидите, что один файл был записан:

Теперь вы можете просто открыть этот файл в своем любимом аудиоплеере и послушать то, что только что было записано.

Я, например, открыл его с помощью бесплатного программного обеспечения для редактирования аудио Audacity, чтобы посмотреть, как выглядит звуковой сигнал:



Поздравляем, вы только что создали свой собственный шпионский микрофон! Теперь вы можете варьировать настройками, например, используя более продолжительное время записи.

Резюме

В этой главе вы узнали, как создать шпионский микрофон на основе простого микрофона с усилителем, Arduino и карты microSD. Шпионский микрофон можно настроить для непрерывной записи звука в течение заданного времени.

Конечно, сейчас вы можете многое сделать, чтобы улучшить этот проект. Вы можете, например, использовать батарею для питания проекта, чтобы сделать его автономным. Затем вам просто нужно поместить его в комнату, в которой вы хотите записать разговор, и просто вернуться позже, чтобы получить SD-карту с записью.

Вы также можете подключить к проекту датчик движения и использовать его для автоматического запуска записи при обнаружении движения в комнате, чтобы полностью перехватить разговоры.

Еще один интересный проект - использовать уровни звука, которые воспринимаются проектом, для фактического запуска или остановки записи. Например, вы можете переписать программное обеспечение проекта, чтобы постоянно контролировать уровень звука в помещении и автоматически запускать запись при превышении заданного порогового значения, что указывает на то, что кто-то говорит. Затем программа могла бы автоматически останавливать запись, когда уровень звука снова становится низким.

В следующей главе книги мы собираемся сделать еще один полезный проект для любого шпиона: например, детектор ЭМИ (электромагнитного излучения) для обнаружения записывающего устройства (клопа) в комнате.

3

Создание детектора ЭМИ

В этой главе мы собираемся создать очень полезный инструмент, который должен быть у каждого секретного агента: детектор клопов. Мы построим простое устройство, которое позволит вам определить, есть ли поблизости какие-либо электромагнитноизлучающие (ЕМИ) устройства, например записывающее устройство или скрытая беспроводная камера.

В этой главе мы рассмотрим следующие темы:

- Мы построим устройство с простой проволочной антенной,которое будет отображать показания электромагнитного излучения(ЭМИ) на ЖК-экране.
- Мы также добавим светодиод,который будет указывать, когда ЭМИ превышает определенный порог.

Ну что,приступаем!

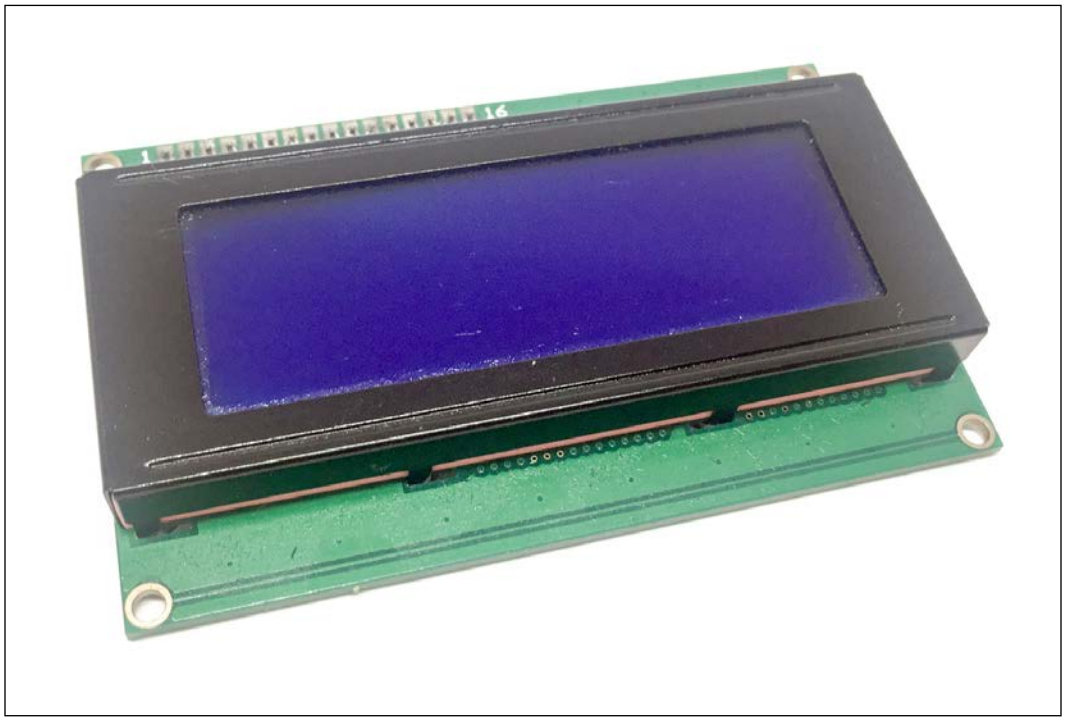
Компоненты и программы

Сначала посмотрим, какие компоненты необходимы для этого проекта.

Вам, конечно же, понадобится Arduino Uno, которая будет работать как мозг проекта и обрабатывать всю информацию.

Вам также понадобится провод длиной от 10 до 20 см, который будет работать как антенна. Вместе с этим проводом вам понадобится резистор 1 МОм.

Для отображения данных мы будем использовать простой ЖК-дисплей I2C. Я использовал экран 4 x 20 I2C от DFRobot:



Вы, конечно, можете использовать любой ЖК-дисплей для этого проекта, вам просто нужно будет использовать соответствующую ЖК-библиотеку.

Я также использовал красный светодиод вместе с резистором 330 Ом,который будет загораться, когда ЭМИ превышает заданный уровень.

Наконец, вам также понадобится макетная плата и перемычки для различных подключений.

Вот список всех компонентов, которые мы будем использовать для этого проекта:

- Arduino Uno
- I2C LCD дисплей
- Красный LED
- Резистор 330 Ом
- Проволока длинная (не менее 10 см)

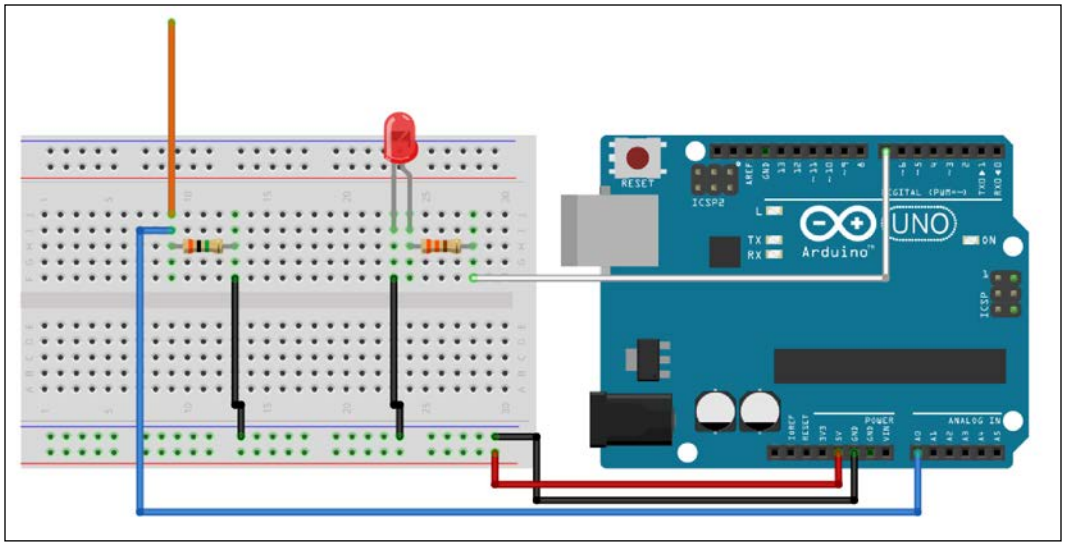
- Резистор 1 МОм
- Макетная плата
- Перемычки

Что касается программного обеспечения, вам понадобится библиотека для ЖК-дисплея. Поскольку в этом проекте мы будем использовать ЖК-дисплей I2C, я рекомендую следующую библиотеку, которую вы можете загрузить с `http://hmario.home.xs4all.nl/arduino/LiquidCrystal_I2C/`.

После того, как библиотека будет правильно установлена, вы можете перейти к следующему шагу.

Аппаратное обеспечение

Теперь мы приступим к настройке аппаратной части проекта. Поскольку у нас относительно небольшое количество компонентов, настройка этого проекта будет действительно простой и понятной. Эта схема поможет вам:



Как видите, ЖК-дисплей на этой схеме отсутствует. Сначала мы увидим, как подключить все остальные компоненты, а затем посмотрим, как подключить ЖК-дисплей в конце этого раздела.

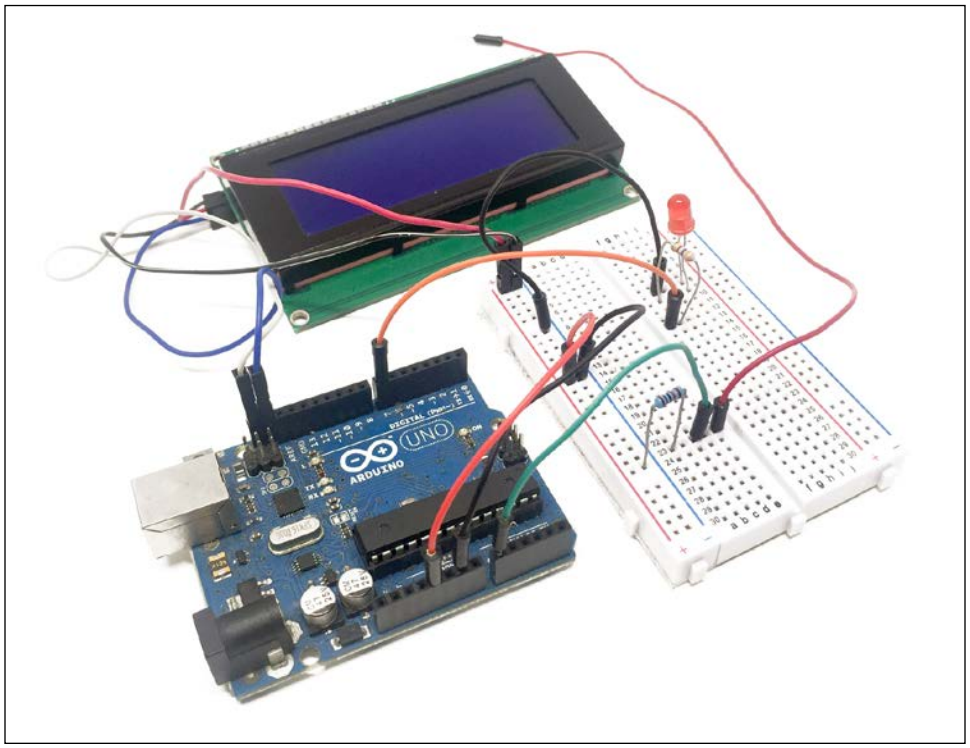
Сначала подключите питание в макетной плате: подключите GND к синей шине питания макетной платы, а контакт + 5V к красной шине питания.

Затем мы собираемся подключить антенну: сначала поместите ее на макетную плату последовательно с резистором 1 МОм. Затем подключите другой конец резистора к земле и, наконец, подключите антенну к аналоговому выводу A0.

Светодиод просто поместите последовательно с резистором, как показано на схеме. Убедитесь, что вы подключили резистор к аноду светодиода, который является самым длинным выводом светодиода. Наконец, подключите другой конец резистора к цифровому выводу 7, а другой конец светодиода - к земле.

Для ЖК-экрана подключение действительно простое, благодаря интерфейсу I2C. Сначала подключите питание: VCC идет на красную шину питания, а GND - на синюю шину питания. Для вывода данных подключите SDA и SCL к соответствующим выводам на плате Arduino, которые находятся рядом с цифровым выводом 13.

Следующее изображение является окончательным результатом:



Поздравляю! Теперь посмотрим, как протестировать ЖК-дисплей.

Тестирование ЖК-дисплея

Прежде чем мы создадим наш детектор ЭМИ, мы хотим убедиться, что ЖК-дисплей работает правильно. Поэтому мы собираемся протестировать его, напечатав на нем очень простое сообщение.

Ниже приводится скетч для этого:

```
// Необходимые библиотеки
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// описание параметров дисплея
LiquidCrystal_I2C lcd(0x27,20,4);

void setup()
{
  // Инициализировать LCD
  lcd.init();

  // Распечатать сообщение на ЖК-дисплее
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("Hello Secret Agent!");
}



void loop()
{
}
```

Как видите, скетч довольно прост. Теперь вы можете подключить проект Arduino к компьютеру с помощью USB-кабеля, а затем скопировать и вставить скетч в свою Arduino IDE.

Затем загрузите скетч на свою плату. Вот что вы должны увидеть:

Если вы видите это сообщение, поздравляю, вы готовы перейти к следующему шагу!

Создание детектора ЭМИ

Теперь мы собираемся погрузиться в суть этого проекта и настроить его так, чтобы он мог обнаруживать активность ЭМИ вокруг антенны.

Ниже приводится полный код этого проекта:

```
// Необходимые библиотеки
#include <Wire.h>
#include <LiquidCrystal_I2C.h>


// Количество чтений
#define NUMREADINGS 15

// Параметры для детектора ЭМИ
int senseLimit = 15;
int probePin = 0;
int ledPin = 7;
int val = 0;
int threshold = 200;

// Усреднение измерений
```

```
int readings[NUMREADINGS];
int index = 0;
int total = 0
int average = 0;

// Время между чтениями
int updateTime = 40;

// Определяем параметры дисплея
LiquidCrystal_I2C lcd(0x27,20,4);

void setup()
{
  // Initialise LCD
  lcd.init();

  // Установить светодиод как выход
  pinMode(ledPin, OUTPUT);

  // Распечатать приветственное сообщение на
  //ЖК-дисплее

  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("EMF Detector Started");delay(1000);
  lcd.clear();
}

void loop()
{
  // Читать с пробника (зонда)
  val = analogRead(probePin);
  Serial.println(val);

  // Проверить чтение
  (val >= 1){

    // Ограничение и отображение с предельным значениями
    val = constrain(val, 1, senseLimit);
    val = map(val, 1, senseLimit, 1, 1023);

    // Усреднение чтения
    total -= readings[index];readings
    [index] = val;
```

```
      total += readings[index];
      index = (index + 1);

      if (index >= NUMREADINGS)
        index = 0;

      average = total / NUMREADINGS;

      // Печать на ЖК-дисплей
      lcd.setCursor(0,1);
      lcd.print("   ");

      lcd.setCursor(0,0);
      lcd.print("EMF level: ");
      lcd.setCursor(0,1);
      lcd.print(average);

      // Загорается светодиод, если обнаружена активность ЭМИ
      if (average > threshold) {
        digitalWrite(ledPin, HIGH);
      }
      else {
        digitalWrite(ledPin, LOW);
      }

      // Wait until next reading
      delay(updateTime);
    }
  }
```

Теперь давайте посмотрим на детали этого кода. Он начинается с включения необходимых библиотек:

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
```

Затем мы определяем количество считываний, которое мы хотим делать на каждой итерации, чтобы убедиться, что мы получаем среднее значение ЭМИ:

```
#define NUMREADINGS 15
```

Затем мы определим некоторые параметры для проекта, такие как различные контакты, к которым подключаются компоненты:

```
int senseLimit = 15;
int probePin = 0;
int ledPin = 7;
int val = 0;
int threshold = 200;
```

После этого мы определим некоторые переменные, которые будут использоваться для получения среднего значения показаний:

```
int readings[NUMREADINGS];
int index = 0;
int total = 0;
int average = 0;
```

Мы также установим время обновления, которое мы будем оставлять между каждым чтением EMF (ЭМИ).
По умолчанию мы установили 40 миллисекунд:

```
int updateTime = 40;
```

Мы также опишем параметры ЖК-дисплея:

```
LiquidCrystal_I2C lcd(0x27,20,4);
```

После этого в функции скетча setup () мы инициализируем ЖК-дисплей:

```
// Инициализировать ЖК-дисплей
  lcd.init();

  // Установить светодиод как выход
  pinMode(ledPin, OUTPUT);

  // Распечатать приветственное сообщение на ЖК-дисплее
  lcd.backlight();
  lcd.setCursor(0,0);
  lcd.print("EMF Detector Started");
  delay(1000);
  lcd.clear();
```

Затем в функции loop () скетча мы получим значение аналогового вывода, к которому подключена антенна:

```
val = analogRead(probePin);
```

После этого мы ограничим чтение предельным значением, которое было определено ранее, а затем снова сопоставим его между 1 и 1023:

```
val = constrain(val, 1, senseLimit);
val = map(val, 1, senseLimit, 1, 1023);
```

Затем мы возьмем среднее значение показаний:

```
total -= readings[index];
readings[index] = val;
total += readings[index];
index = (index + 1);

if (index >= NUMREADINGS)
  index = 0;
average = total / NUMREADINGS;
```

Наконец, мы отобразим среднее значение на ЖК-экране, а также включим светодиод, если среднее значение выше порогового значения:

```
lcd.setCursor(0,1);
lcd.print("   ");

lcd.setCursor(0,0);
lcd.print("EMF level: ");
lcd.setCursor(0,1);
lcd.print(average);

// Загорается светодиод, если обнаружена активность ЭМИ
if (average > threshold) {
  digitalWrite(ledPin, HIGH);
}
else {
  digitalWrite(ledPin, LOW);
}
```

Мы также будем ждать определенное время между каждым чтением:

```
delay(updateTime);
```

Обратите внимание, что вы можете найти полный код в репозитории GitHub проекта по адресу https://github.com/marcoschwartz/arduino-secret-agents.

Пришло время протестировать проект. Вы можете скопировать полный код (chapter03 в папке с кодами) и вставить его в свою IDE Arduino

Затем загрузите код (chapter03)в свой проект. Через некоторое время вы должны увидеть, что ЖК-экран инициализируется и остается пустым. Это потому, что он ожидает обнаружения значительной активности ЭМИ.

Я, например, подошел со своим телефоном к антенне и получил такой результат:

Если вы видите что-то на своем экране, поздравляем, вы только что создали детектор ЭМИ (EMF)! Теперь я предлагаю вам попробовать другие устройства, такие как камера наблюдения, устройство Wi-Fi, радиоконтроллер и т. д.

Резюме

В этой главе мы создали простой детектор электромагнитного поля, который секретный агент может использовать, чтобы определить, есть ли какие-либо устройства (клопы) в комнате, например, в аудиорекордере или в камере видеонаблюдения.

Конечно, есть несколько способов улучшить этот проект. Вы можете, например, подключить к проекту аккумулятор и интегрировать его в небольшой корпус, чтобы создать простой и портативный детектор ЭМИ.

В следующей главе мы собираемся построить еще одно полезное устройство для секретного агента. Мы увидим, как использовать сканер отпечатков пальцев для защиты доступа к важной информации.

4

Контроль доступа с помощью датчика отпечатков пальцев

В этой главе мы собираемся создать более сложный проект секретного агента: систему контроля доступа с использованием датчика отпечатков пальцев. Мы подключим датчик отпечатков пальцев к Arduino вместе с реле и ЖК-экраном.

На основе этого железа построим несколько крутых проектов. Ниже приведены шаги, которые мы предпримем для этого проекта:

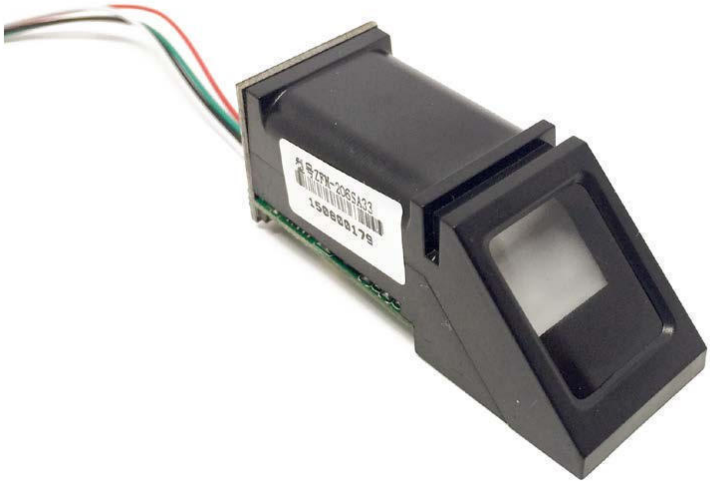
- Во-первых, мы собираемся записать ваш отпечаток пальца в датчик, чтобы вы могли получить доступ
- Затем мы будем использовать датчик отпечатков пальцев для работы с реле.
- Наконец, мы создадим систему с ЖК-экраном для предоставления доступа к секретной части данных, хранящейся в Arduino.

Итак приступаем!

Компоненты и программы

Как обычно, мы будем использовать плату Arduino Uno как мозг проекта.

Самая важная часть этого проекта - датчик отпечатков пальцев. Ниже приведено изображение датчика, который я использовал:



С датчиком отпечатка пальца необходимо иметь библиотеку,которая заточена именно под данную модель. В противном случае код этого проекта работать не будет.

Вам также понадобится ЖК-дисплей для последней части этого проекта. Я использовал ЖК-экран I2C от DFRobot, который мы уже использовали ранее в книге.

Еще я использовал релейный модуль 5V фирмы Pololu, который действительно удобно подключать к Arduino. Реле в основном позволяет нам управлять широким спектром устройств, от простого светодиода до электроприборов.

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

- Arduino Uno
- Adafruit датчик отпечатка пальца
- DFRobot 4x20 LCD дисплей

- Релейный модуль 5В Pololu
- Макетная плата
- Перемычки

Что касается программного обеспечения, вам понадобятся две библиотеки Arduino: библиотека LiquidCrystal_I2C для ЖК-дисплея и библиотека датчика отпечатков пальцев Adafruit.

Вы можете получить их оба с помощью диспетчера библиотек Arduino.

Вы также можете посетить репозиторий GitHub библиотеки Fingerprint Sensor, чтобы узнать больше о различных функциях, доступных на https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library.

Аппаратная конфигурация

Сначала мы посмотрим, как собрать различные части этого проекта.

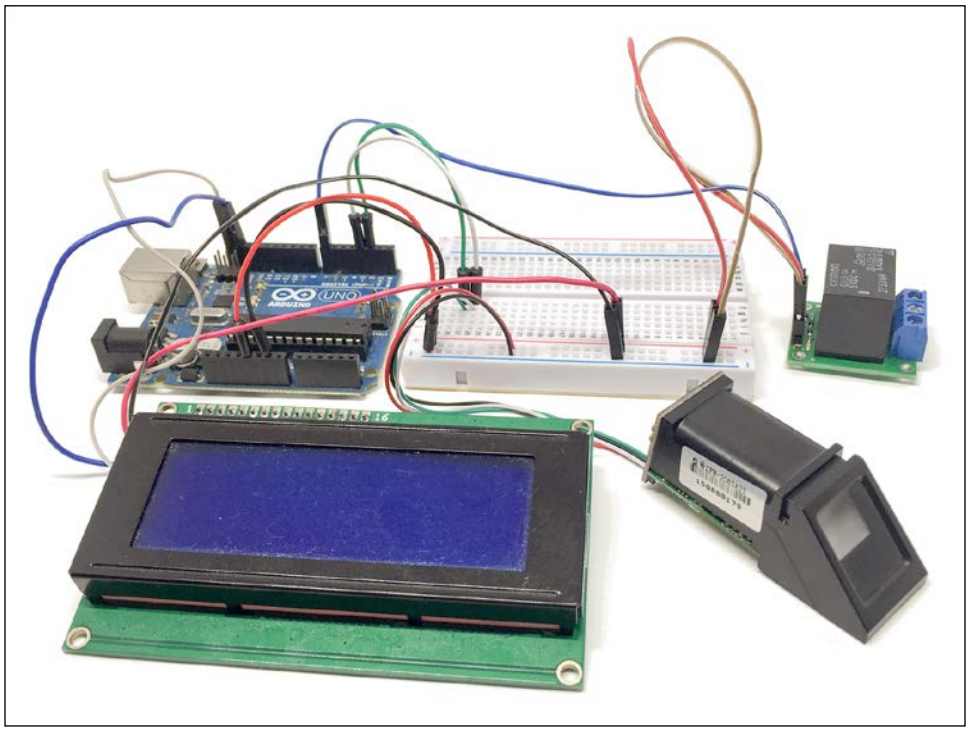
Начнем с подключения питания. Подключите вывод 5V от платы Arduino к красной шине питания, а GND от Arduino к синей шине питания на макетной плате.

Теперь подключим датчик отпечатка пальца. Сначала подключите питание, соединив кабели соответствующего цвета на макете. Затем подключите белый провод от датчика к контакту 3 Arduino, а зеленый провод к контакту 2.

После этого мы собираемся подключить релейный модуль. Подключите контакт VCC к красной шине питания, контакт GND к синей шине питания, а контакт EN к контакту 7 Arduino.

Наконец, подключим ЖК-дисплей. Сначала подключите питание: VCC к красной шине питания и вывод GND к синей шине питания. После этого подключите выводы I2C (SDA и SCL) к плате Arduino. Контакты I2C находятся рядом с контактом 13 на плате Arduino Uno.

Ниже приводится окончательный результат:



Теперь вы можете перейти к следующему разделу этой главы и создать захватывающие приложения на основе датчика отпечатков пальцев!

Регистрация вашего отпечатка пальца

Первое, что нам нужно сделать, это зарегистрировать хотя бы один отпечаток пальца, чтобы его впоследствии можно было распознать датчиком. Мы сделаем это в этом разделе. Вот большая часть кода для этого раздела:

```
// Библиотеки
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

// Функция регистрации отпечатков пальцев
uint8_t getFingerprintEnroll(uint8_t id);

// Задание данных
```

```
SoftwareSerial mySerial(2, 3);

// описание параметров датчика отпечатков пальцев
Adafruit_Fingerprint finger = Adafruit_Fingerprint
(andmySerial);

void setup()
{
  // Start serial
  Serial.begin(9600);
  Serial.println("fingertest");

  // Установите скорость передачи данных для последовательного
  //порта датчика
  finger.begin(57600);

  // Убедитесь, что датчик присутствует
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
}

void loop()
{
  // Дождитесь идентификатора отпечатка пальца
  Serial.println("Type in the ID # you want to save this finger
as...");
  uint8_t id = 0;
  while (true) {
    while (! Serial.available());
    char c = Serial.read();
    if (! isdigit(c)) break;
    id *= 10;
    id += c - '0';
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll(id) );
}
```

Обратите внимание, что я не включил все подробности о функциях датчика отпечатков пальцев, поскольку они слишком длинные для отображения. Вы, конечно, можете найти весь код в репозитории GitHub этой книги - в прилагаемой папке с кодами.

Теперь давайте рассмотрим детали кода. Он начинается с включения необходимых библиотек:

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
```

Затем мы объявляем функцию, которую будем использовать для регистрации нашего отпечатка пальца:

```
uint8_t getFingerprintEnroll(uint8_t id);
```

Реализация последовательного интерфейса на цифровых выводах 2 и 3

```
SoftwareSerial mySerial(2, 3);
```

Мы также создадим образец отпечатка пальца для датчика:

```
Adafruit_Fingerprint finger = Adafruit_Fingerprint(andmySerial);
```

Теперь в функции скетча setup () мы инициализируем последовательную связь:

```
Serial.begin(9600);
Serial.println("fingertest");
```

Затем мы инициализируем связь с датчиком:

```
finger.begin(57600);
```

Также проверим, присутствует ли датчик:

```
if (finger.verifyPassword()) {
  Serial.println("Found fingerprint sensor!");
} else {
  Serial.println("Did not find fingerprint sensor :(");
  while (1);
}
```

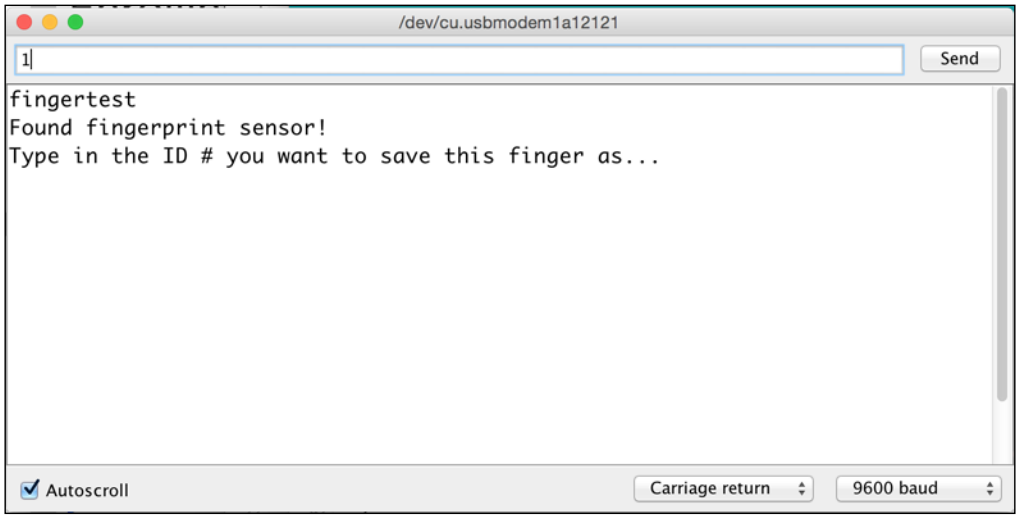
В функции цикла () кода мы сначала ожидаем ввода от пользователя, который является идентификатором отпечатка пальца, который мы хотим сохранить. Затем мы выполняем процесс сохранения отпечатка пальца. Все это делается с помощью следующего фрагмента кода:

```
// Wait for fingerprint ID
Serial.println("Type in the ID # you want to save this finger as...");
  uint8_t id = 0;
  while (true) {
    while (! Serial.available());
```

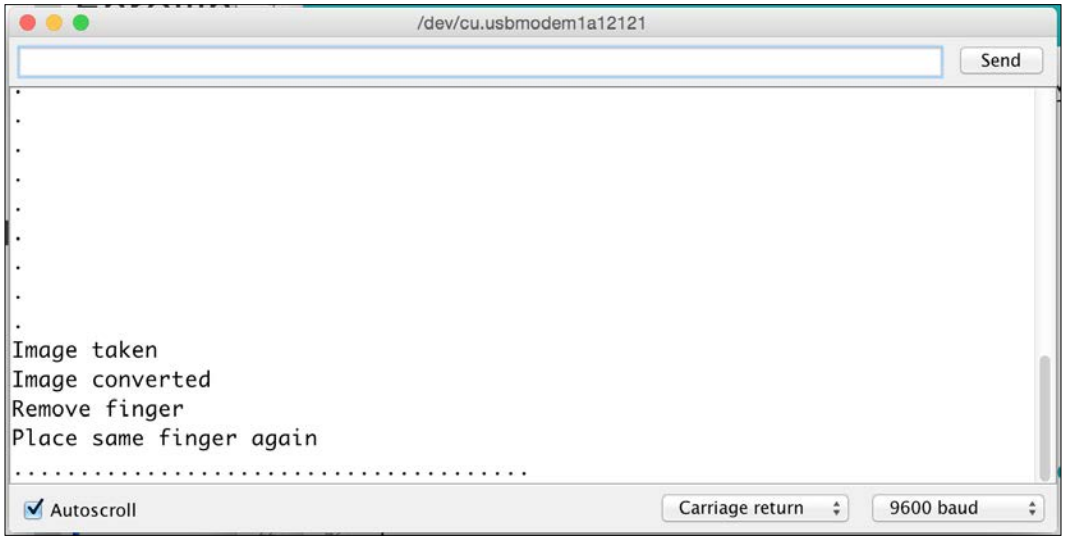
```
    char c = Serial.read();
    if (! isdigit(c)) break;
    id *= 10;
    id += c - '0';
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll(id) );
```

Пришло время протестировать процесс регистрации. Сначала получите полный код, например, из репозитория книги на GitHub, который доступен по адресу https://github.com/marcoschwartz/arduino-secret-agents или из папки с кодами.

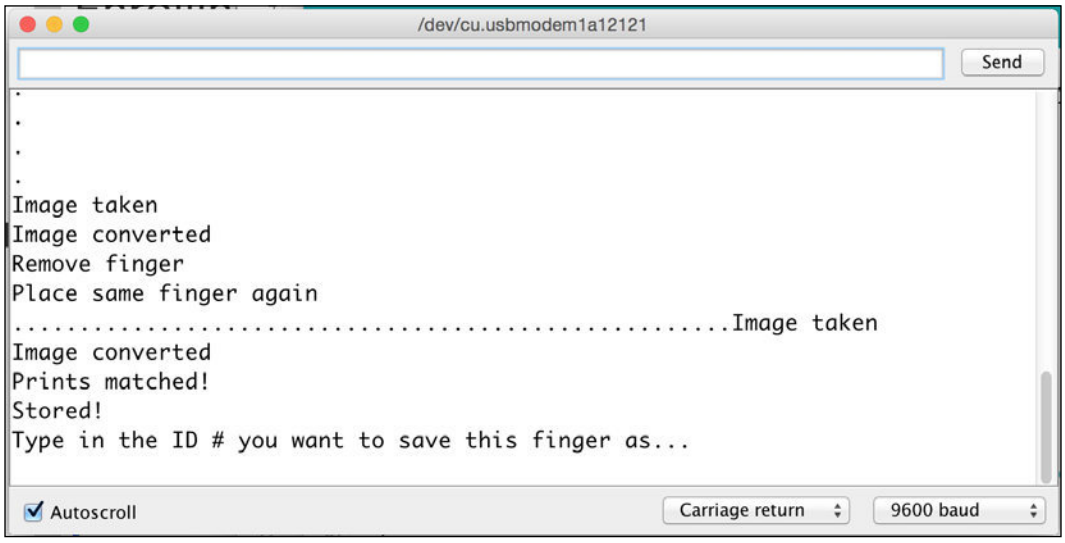
Затем скопируйте код в Arduino IDE. После этого загрузите код на плату Arduino и откройте последовательный монитор со скоростью 9600 бод. Вы должны увидеть следующее:



При появлении запроса введите идентификатор отпечатка пальца, который вы хотите сохранить, и презентатора. Теперь программа попросит вас приложить палец к датчику. Сделайте это, и через некоторое время вы увидите, что изображение было снято, и теперь вы можете убрать палец:



Затем, как показано на рисунке, еще раз приложите палец к датчику. Затем программа подтвердит, что отпечаток пальца сохранен:



Если вы видите это сообщение, это означает, что ваш отпечаток пальца теперь сохранен в датчике, и вы можете перейти к следующему разделу.

Управление доступом к реле

Теперь, когда ваш отпечаток пальца сохранен в датчике, вы можете создать свое первое приложение, используя устройство, которое мы создали ранее. Мы собираемся управлять работой реле каждый раз, когда датчик распознает наш отпечаток пальца. Ниже приведен полный код для этой части, за исключением деталей функции распознавания:

```
// Libraries
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>

// Функция получения отпечатка пальца
int getFingerprintIDez();

// задание параметров
SoftwareSerial mySerial(2, 3);

// задание параметров датчика отпечатков пальцев
Adafruit_Fingerprint finger = Adafruit_Fingerprint(andmySerial);

// подключение реле на 7 вывод
int relayPin = 7;
bool relayState = false;

// Ваш сохраненный идентификатор пальца
int fingerID = 0;

// Счетчики
int activationCounter = 0;
int lastActivation = 0;

void setup()
{

  // старт Serial
  Serial.begin(9600);

  // Установить скорость передачи данных для последовательного порта датчика
  finger.begin(57600);

  // Проверить наличие датчика
```

```
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
  Serial.println("Waiting for valid finger...");

  // Set relay as output
  pinMode(relayPin, OUTPUT);
}

void loop()
{
  // Получить отпечаток пальца № ID
  int fingerprintID = getFingerprintIDez();

  // Активация?
  if ( (activationCounter - lastActivation) > 2000) {

  if (fingerprintID == fingerID && relayState == false) {
    relayState = true;
    digitalWrite(relayPin, HIGH);
    lastActivation = millis();
  }
  else if (fingerprintID == fingerID && relayState == true) {
    relayState = false;
    digitalWrite(relayPin, LOW);
    lastActivation = millis();
  }

  }
  activationCounter = millis();
  delay(50);
}
```

Как видите, многие элементы совпадают с скетчем, который мы видели в предыдущем разделе. Мы увидим только те элементы, которые важны и добавлены в этот новый скетч.

Мы должны определить, к какому выводу подключено реле, а также указать, что реле выключено по умолчанию:

```
int relayPin = 7;
bool relayState = false;
```

Затем мы определим идентификатор, под которым мы сохранили отпечаток пальца в предыдущем разделе. Я использовал 0, когда сохранил свой отпечаток пальца с идентификационным номером 0:

```
int fingerID = 0;
```

Кроме того, мы не хотим, чтобы реле постоянно переключало состояние, когда мы касаемся датчика пальцем. Следовательно, нам нужны следующие две переменные, чтобы отсчитать 2 секунды, прежде чем состояние реле можно будет снова изменить:

```
int activationCounter = 0;
int lastActivation = 0;
```

Затем мы установим релейный вывод как выход:

```
pinMode(relayPin, OUTPUT);
```

Затем в функции скетча loop () мы проверяем, считывает ли датчик какой-либо идентификатор отпечатка пальца, который уже сохранен в датчике:

```
int fingerprintID = getFingerprintIDez();
```

Ниже приводится проверка на период активации:

```
if ( (activationCounter - lastActivation) > 2000) {
```

Затем мы проверим, соответствует ли идентификатор идентификатору, который мы определили ранее, а также проверим состояние реле. Если ID соответствует ID, который мы ввели в скетч, мы переключаем состояние реле:

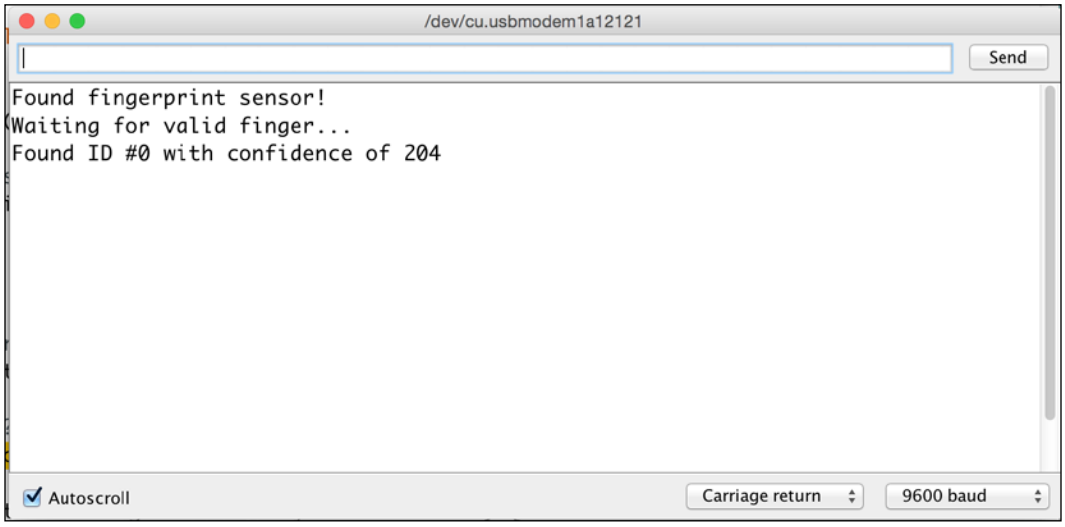
```
if (fingerprintID == fingerID && relayState == false) {
     relayState = true;
  digitalWrite(relayPin, HIGH);
  lastActivation = millis();
   }
else if (fingerprintID == fingerID && relayState == true) {
  relayState = false;
  digitalWrite(relayPin, LOW);
  lastActivation = millis();
}
```

Наконец, мы обновим счетчик активации и подождем 50 миллисекунд до следующего чтения:

```
activationCounter = millis();
delay(50);
```

Пришло время протестировать скетч. Загрузите код, например, из прилагаемой папки с кодами (capter04), а затем загрузите его на плату. Убедитесь, что вы изменили идентификатор в скетче, соответствующий отпечатку пальца, который вы сохранили ранее.

Затем откройте последовательный монитор со скоростью 9600 бод. Поместите палец, который вы прикладывали ранее, на датчик. На мониторе последовательного порта вы должны сразу увидеть следующее:



Вы также должны услышать щелчок реле, означающий, что оно только что изменило свое состояние. Теперь вы можете проделать ту же операцию через несколько секунд; вы должны услышать, как реле возвращается в исходное состояние.

Вы можете попробовать процедуру идентификации другим пальцем или попросить кого-нибудь другого - вообще ничего не произойдет.

Доступ к секретным данным

В последнем разделе этой главы мы собираемся использовать все устройство, которое мы подключили к проекту, для другого интересного приложения: доступа к секретной части данных с помощью вашего отпечатка пальца.

Это может быть, например, секретный код, доступ к которому вы хотите получить только по вашему отпечатку пальца. Для этого мы будем использовать ЖК-дисплей, избавляя от необходимости открывать монитор последовательного порта.

Ниже приводится полный код этой части, за исключением функций для чтения данных датчика отпечатков пальцев:

```
// Библиотеки
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

// Определяем параметры ЖК-дисплея
LiquidCrystal_I2C lcd(0x27,20,4);

// Функция получения отпечатка пальца
int getFingerprintIDez();

// Последовательный порт программы
SoftwareSerial mySerial(2, 3);

// определяем параметры датчика отпечатков пальцев
Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

// Параметры реле
int relayPin = 7;
bool relayState = false;

// Ваш сохраненный идентификатор пальца
int fingerID = 0;

// Счетчики
int activationCounter = 0;
int lastActivation = 0;

// Секретные данные
String secretData = "u3fks43";

void setup()
{

  // Начать последовательную передачу
  Serial.begin(9600);

  // Установите скорость передачи данных для последовательного
  // порта датчика
```

```
  finger.begin(57600);

  // Проверить наличие датчика
  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1);
  }
  Serial.println("Waiting for valid finger...");

  // Set relay as output
  pinMode(relayPin, OUTPUT);

  // Init display
  lcd.init();
  lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Scan your finger");
}

void loop()
{
  // Получить отпечаток пальца № ID
  int fingerprintID = getFingerprintIDez();

  // Активация?
  if ( (activationCounter - lastActivation) > 2000) {

    if (fingerprintID == fingerID) {

      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Access granted!");
      lcd.setCursor(0,1);
      lcd.print("Your secret data is:");
      lcd.setCursor(0,2);
      lcd.print(secretData);

      if (relayState == false) {
        relayState = true;
        digitalWrite(relayPin, HIGH);
      }
```

```
        else if (relayState == true) {
          relayState = false;
          digitalWrite(relayPin, LOW);
        }
        lastActivation = millis();
      }
    }
    activationCounter = millis();
    delay(50);
}
```

Как видите, этот код имеет некоторые общие части с кодом, который мы видели в предыдущем разделе. Поэтому здесь мы поговорим только о самых важных элементах.

Он начинается с включения необходимых библиотек:

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
```

Затем нам нужно описать параметры дисплея LiquidCrystal_I2CLCD:

```
LiquidCrystal_I2C lcd(0x27,20,4);
```

Мы также определим строку, содержащую наши секретные данные:

```
String secretData = "u3fks43";
```

Конечно, можете смело ставить сюда что угодно. Затем в функции setup () мы инициализируем ЖК-дисплей:

```
lcd.init();
lcd.backlight();
lcd.clear();
lcd.setCursor(0,0);
lcd.print("Scan your finger");
```

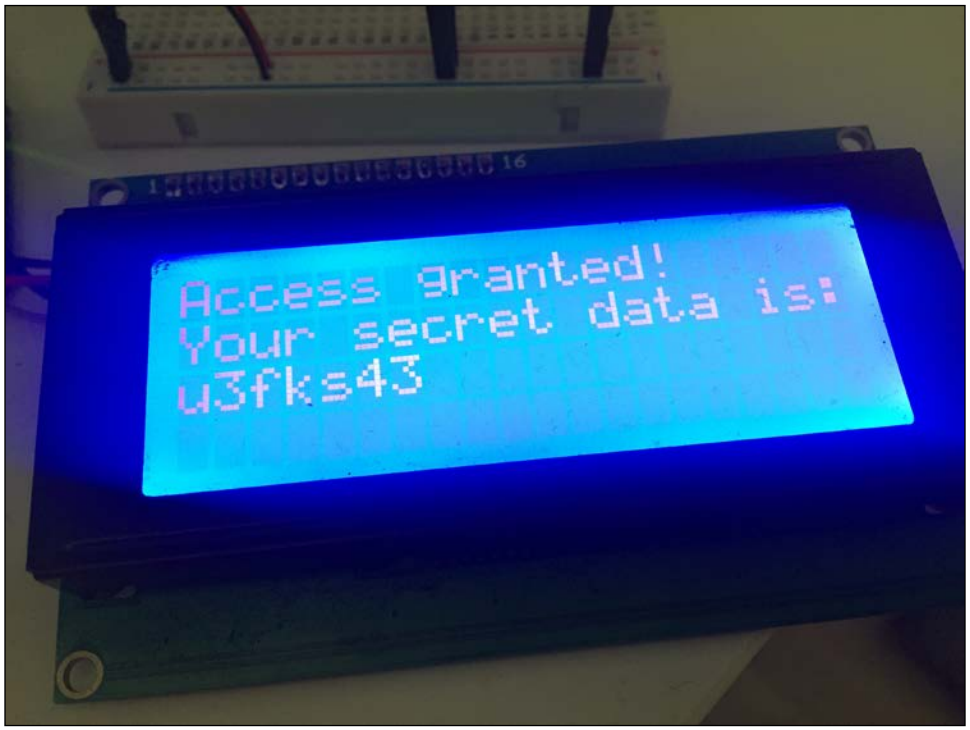
Затем в функции скетча loop (), если мы вышли за пределы периода активации, равного 2 секундам, мы сравним измеренный отпечаток пальца с идентификатором, который мы установили в скетче. Если это верно, мы напечатаем на ЖК-дисплее сообщение о том, что доступ был предоставлен вместе с секретными данными:

```
if (fingerprintID == fingerID) {

      lcd.clear();
      lcd.setCursor(0,0);
```

```
lcd.print("Access granted!");
lcd.setCursor(0,1);
lcd.print("Your secret data is:");
lcd.setCursor(0,2);
lcd.print(secretData);
```

Пришло время протестировать скетч. Загрузите весь код, например, из прилагаемой папки с кодами (chapter04) и загрузите его на плату.

Затем поместите свой палец, который вы ранее записали, на датчик. Вот что вы должны увидеть:



Если вы видите предыдущий результат, поздравляем, вы только что создали свою собственную систему контроля доступа по отпечаткам пальцев!

Резюме

В этом проекте мы построили систему контроля доступа с использованием датчика отпечатков пальцев с Arduino. Мы также создали на его основе несколько интересных приложений, в том числе способ получить доступ к секретным данным с помощью вашего отпечатка пальца.

Конечно, вы можете многое сделать, чтобы улучшить этот проект. Например, вы можете указать способ исчезновения секретного сообщения по прошествии определенного времени; избегая того, чтобы кто-то другой получил к нему доступ после того, как вы использовали проект.

Вы также можете подключить плату к Интернету (например, с помощью шилда Ethernet) и использовать датчик отпечатков пальцев для отправки заданного твита, когда вы кладете на него палец.

В следующей главе мы собираемся использовать другое оборудование для создания системы контроля доступа: шилд GSM, который позволит нам открывать дверь, просто отправив SMS!

5

Открытие замка с помощью SMS

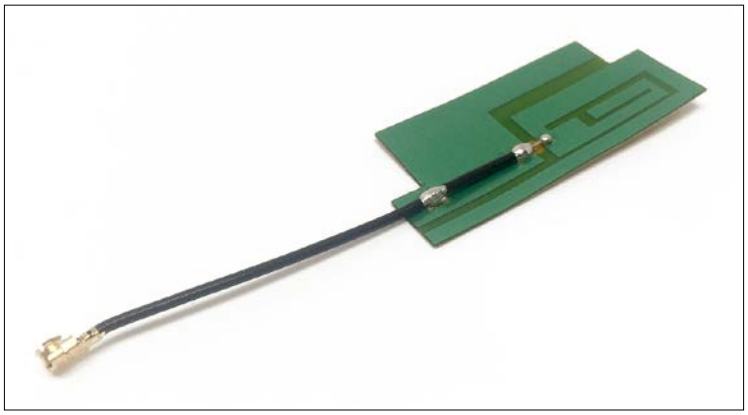
В этой главе мы собираемся создать еще одно отличное приложение для секретных агентов: открыть дверной замок просто отправив SMS! Вам просто нужно отправить сообщение на заданный номер, и тогда вы сможете управлять дверным замком или любым другим цифровым устройством, например, сигнализацией,включая / выключая ее.

Для этого мы собираемся предпринять ряд шагов, а именно:

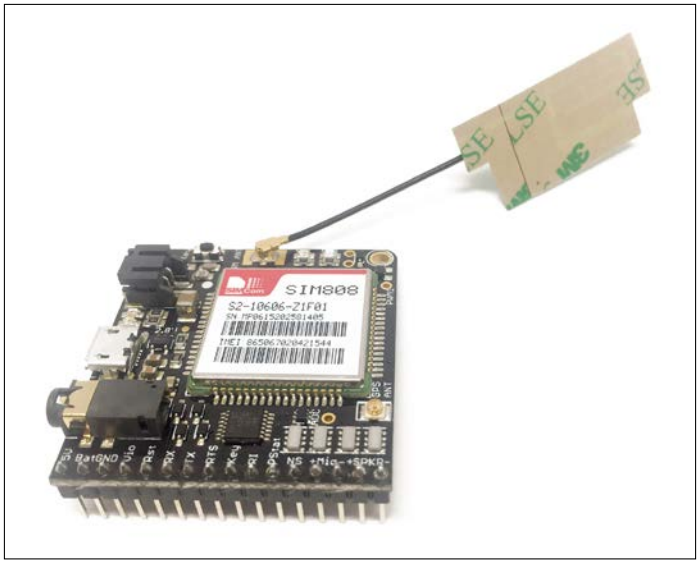
- Во-первых, мы собираемся использовать Arduino и щит FONA от Adafruit, чтобы иметь возможность получать и обрабатывать текстовые сообщения.
- Затем мы подключим это к реле и светодиоду, чтобы увидеть, можем ли мы действительно управлять устройством, отправляя SMS.
- Наконец, мы увидим, как подключить настоящий электронный замок к системе.

HARD и SOFT

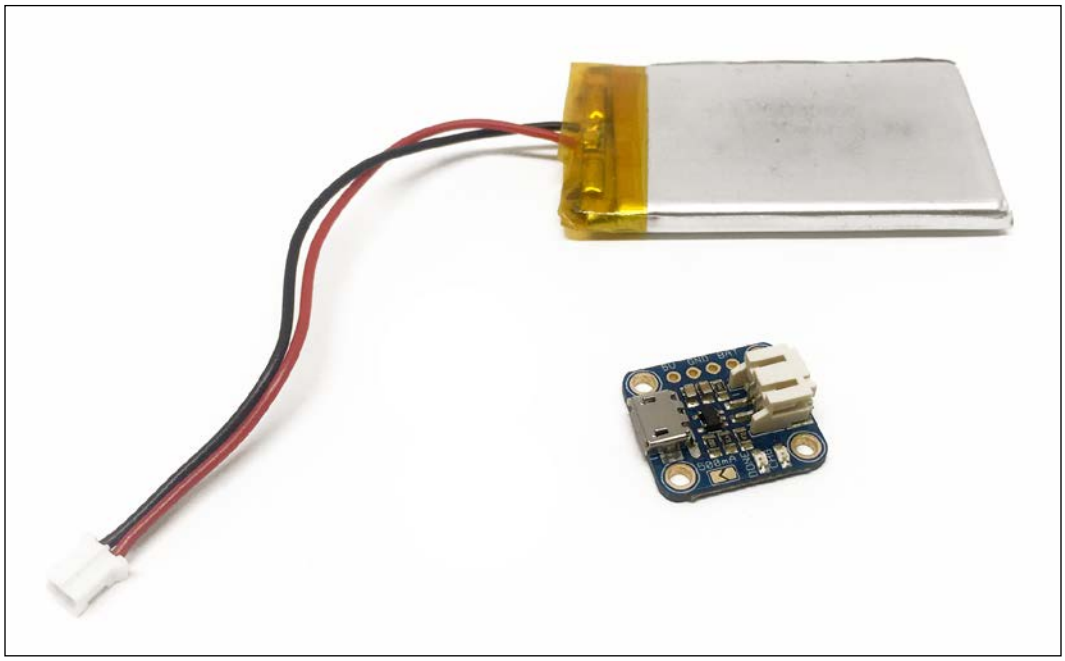
Во-первых, давайте посмотрим, какие компоненты необходимы для этого проекта. Наиболее важные части связаны с функциональностью GSM, которая является центральной частью нашего проекта.
Нам понадобится антенна, чтобы иметь возможность подключиться к локальной сети GSM. В этом проекте используется плоская антенна uFL:



Затем вам понадобится способ фактически использовать SIM-карту, подключиться к сети GSM и обрабатывать информацию с помощью Arduino. Есть много плат, которые могут это сделать, однако я рекомендую шилд Adafruit FONA, который очень удобно настраивать и использовать с Arduino. Ниже показано изображение шилда Adafruit FONA вместе с плоской антенной GSM.



Затем вам понадобится батарея для питания щита FONA, поскольку плата Arduino Uno не позволяет запитать чип, который находится в ядре шилда FONA (он может использоватьдо 2 А за раз!). Для этого я использовал 3,7 LiPo аккумулятор вместе с зарядным устройством для аккумулятора micro USB:



Очень важной частью проекта является SIM-карта, которую нужно вставить в шилд FONA. Вам нужна обычная SIM-карта (не micro или nano), которая активирована, не заблокирована PIN-кодом и может принимать текстовые сообщения. Вы можете получить ее у любого из местных операторов мобильной связи.

Затем, чтобы проверить функциональность проекта, мы также будем использовать светодиод (вместе с резистором 330 Ом) и реле на 5 В. Они будут имитировать поведение реального электронного замка.

Наконец, вам понадобится электронный замок. Эта часть не является обязательной, так как вы можете полностью протестировать все без него.
Для этого вам понадобится электронный замок Adafruit, резистор на 1 кОм и силовой транзистор.

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

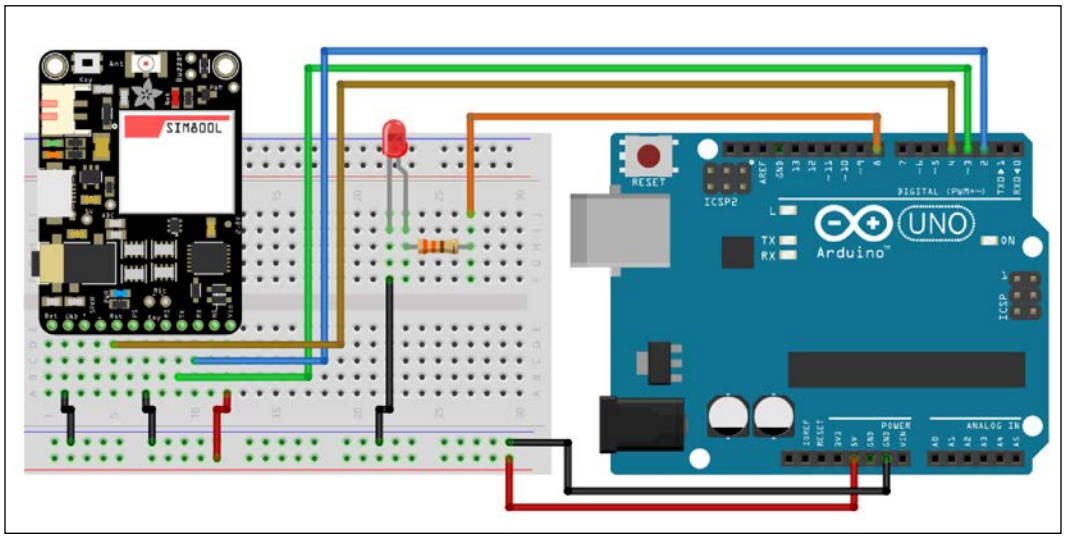
- Arduino Uno
- Adafruit Fona 808 шилд
- GSM uFL антенна
- GSM SIM карта
- 3.7V LiPo аккумулятор
- LiPo battery charger
- LED
- 330 Ohm resistor
- 5V реле `pololu`
- Adafruit электрический замок
- Диоды Шоттки
- Силовой транзистор
- Резистор 1 кОм
- 12V блок питания
- DC разъем пит.
- Макетная плата
- Перемычки)

Что касается программного обеспечения, вам понадобится только последняя версия Arduino IDE и библиотека Adafruit FONA. Вы можете установить эту библиотеку с помощью диспетчера библиотек Arduino IDE.

Аппаратные настройки

Пришло время собрать оборудование для проекта. Сначала мы подключим шилд Adafruit FONA, а затем другие компоненты. Перед сборкой оборудования убедитесь, что вы вставили SIM-карту в шилд FONA.

Следующая схема поможет вам:



Обратите внимание, что расположение выводов на вашем шилде FONA может отличаться, так как доступно множество версий. Также обратите внимание, что реле не представлено на этой схеме. Вот шаги, которые необходимо предпринять для сборки устройства:

1. Сначала подайте питание на макетную плату. Подключите вывод 5V от платы Arduino к красной линии питания на макетной плате, а контакт GND к синей линии питания.

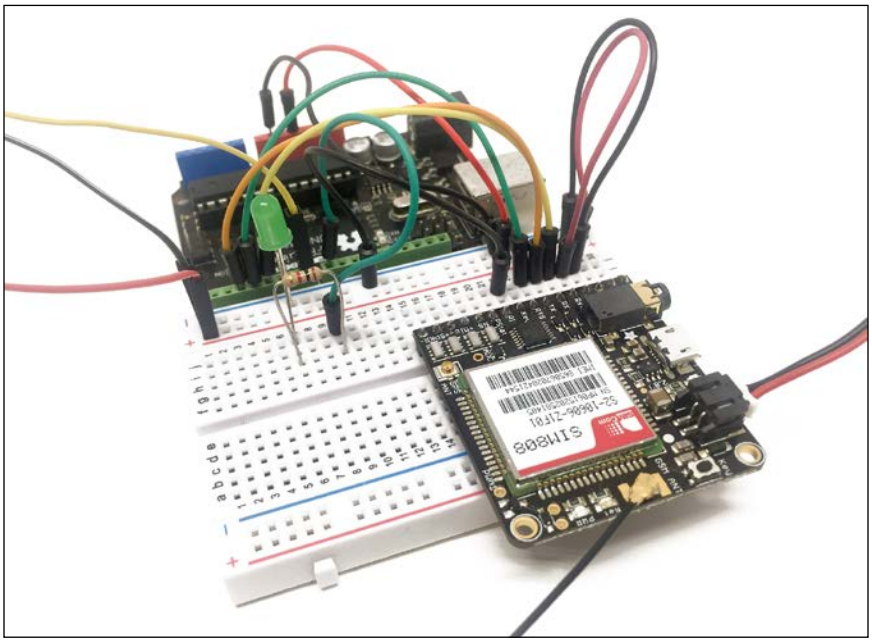
2. Затем поместите шилд FONA на макетную плату. Подключите контакт VIO к красной линии питания, а контакты GND и ключа к синей линии питания.

3. После этого подключите вывод RST к контакту 4 Arduino, TX - к выводу 3 Arduino, а RX - к выводу 2 Arduino. Также подключите LiPo аккумулятор 3,7 В и антенну к шилду FONA.

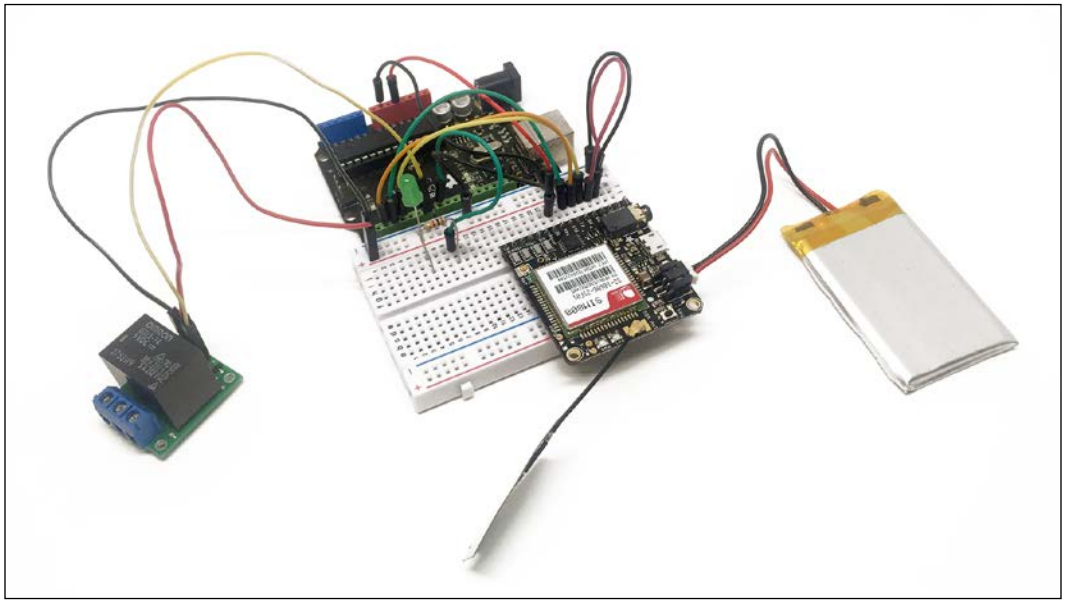
4. Наконец, подключите реле и светодиод к плате Arduino. Подключите вывод VCC и GND к источнику питания на макетной плате, а вывод EN - к выводу 7 платы Arduino.

5. На макетной плате длинный вывод светодиода подключите последовательно с резистором 330 Ом,а короткий вывод его - к земле. Затем подключите другой вывод резистора к выводу 8 платы Arduino.

Ниже приведено изображение завершенного проекта с увеличенным изображением платы Arduino и шилда FONA:



Ниже приводится обзор полностью собранного проекта:



Тестирование шилда FONA

Теперь, когда наше устройство готово, мы собираемся протестировать шилд FONA, чтобы убедиться, что он правильно подключен и может ли он подключаться к сети. В качестве теста мы отправим SMS на шилд, а также прочитаем все сообщения, имеющиеся на SIM-карте.

Это полный скетч Arduino для этой части, за вычетом некоторых вспомогательных функций, которые здесь подробно описываться не будут:

```
// Включить библиотеки
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>

// пины
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4

// Буфер для ответов
char replybuffer[255];

// Software serial
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;

// Fona
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

// Функция чтения
uint8_t readline(char *buff, uint8_t maxbuff, uint16_t timeout = 0);

void setup() {

  // Start Serial
  while (!Serial);
  Serial.begin(115200);
  Serial.println(F("FONA basic test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));

  // Init FONA
  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while(1);
  }
```

```
  Serial.println(F("FONA is OK"));

  // Print SIM card IMEI number.
  char imei[15] = {0}; // ДОЛЖЕН использоваться 16-символьный буфер для IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("SIM card IMEI: "); Serial.println(imei);
  }

}

void loop() {

  // Получить количество SMS
  int8_t smsnum = fona.getNumSMS();
  if (smsnum < 0) {
    Serial.println(F("Could not read # SMS"));
  } else {
    Serial.print(smsnum);
    Serial.println(F(" SMS's on SIM card!"));
  }

  // Read last SMS
  flushSerial();
  Serial.print(F("Read #"));
  uint8_t smsn = smsnum;
  Serial.print(F("\n\rReading SMS #")); Serial.println(smsn);

  // Retrieve SMS sender address/phone number.
  if (! fona.getSMSSender(smsn, replybuffer, 250)) {
    Serial.println("Failed!");
  }
  Serial.print(F("FROM: ")); Serial.println(replybuffer);

  // Retrieve SMS value.
  uint16_t smslen;
  if (! fona.readSMS(smsn, replybuffer, 250, &smslen)) { // pass in
buffer and max len!
    Serial.println("Failed!");
  }
  Serial.print(F("***** SMS #")); Serial.print(smsn);
  Serial.print(" ("); Serial.print(smslen); Serial.println(F(") bytes
*****"));
  Serial.println(replybuffer);
```

```
    Serial.println(F("*****"));

    // Flush input
    flushSerial();
    while (fona.available()) {
      Serial.write(fona.read());
    }

    // Wait
    delay(10000);

  }
```

Теперь давайте рассмотрим детали этого скетча. Он начинается с включения необходимых библиотек:

```
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
```

Затем мы определим пины, к которым подключен шилд FONA:

```
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4
```

Мы также определим буфер, который будет содержать SMS, которое мы будем читать с дисплея:

```
char replybuffer[255];
```

Затем мы будем использовать экземпляр SoftwareSerial для связи со шилдом:

```
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = andfonaSS;
```

Мы также создадим экземпляр библиотеки FONA:

```
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
```

Затем в функции скетча setup () мы запустим объект Serial для отладки и напечатаем приветственное сообщение:

```
while (!Serial);
Serial.begin(115200);
Serial.println(F("FONA basic test"));
Serial.println(F("Initializing....(May take 3 seconds)"));
```

После этого инициализируем шилд FONA:

```
fonaSerial->begin(4800);
if (! fona.begin(*fonaSerial)) {
  Serial.println(F("Couldn't find FONA"));
  while(1);
}
Serial.println(F("FONA is OK"));
```

Затем мы прочитаем IMEI SIM-карты, чтобы проверить, работает ли шилд и присутствует ли карта:

```
char imei[15] = {0}; // MUST use a 16 character buffer for IMEI!
uint8_t imeiLen = fona.getIMEI(imei);
if (imeiLen > 0) {
  Serial.print("SIM card IMEI: "); Serial.println(imei);
}
```

Теперь, в функции скетча loop (), мы сначала подсчитаем количество текстовых сообщений, присутствующих в карточке:

```
int8_t smsnum = fona.getNumSMS();
if (smsnum < 0) {
  Serial.println(F("Could not read # SMS"));
} else {
  Serial.print(smsnum);
  Serial.println(F(" SMS's on SIM card!"));
}
```

Затем мы также прочитаем последний и распечатаем его на последовательном мониторе:

```
// Read last SMS
flushSerial();
Serial.print(F("Read #"));
uint8_t smsn = smsnum;
Serial.print(F("\n\rReading SMS #")); Serial.println(smsn);

 // Retrieve SMS sender address/phone number.
if (! fona.getSMSSender(smsn, replybuffer, 250)) {
  Serial.println("Failed!");
}
Serial.print(F("FROM: ")); Serial.println(replybuffer);

// Retrieve SMS value.
uint16_t smslen;
```

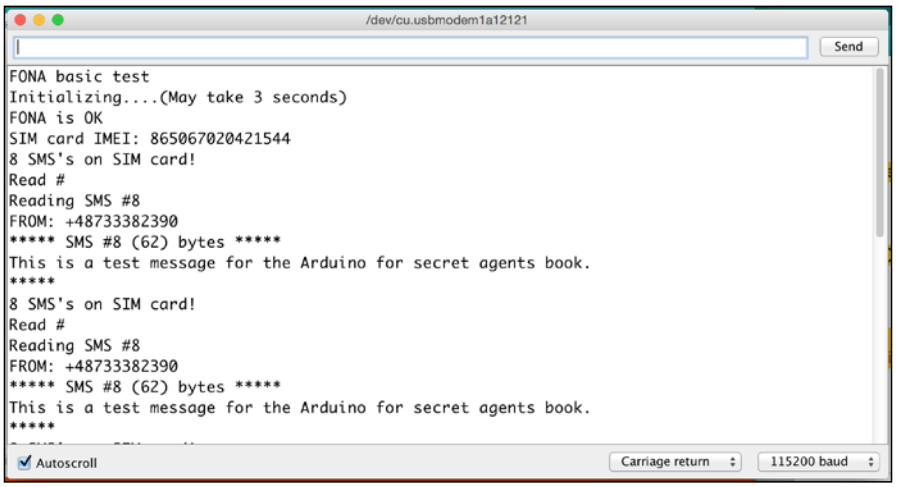
```
if (! fona.readSMS(smsn, replybuffer, 250, &smslen)) { // pass in
buffer and max len!
  Serial.println("Failed!");
}
Serial.print(F("***** SMS #")); Serial.print(smsn);
Serial.print(" ("); Serial.print(smslen); Serial.println(F(") bytes
*****"));
Serial.println(replybuffer);
Serial.println(F("*****"));
```

Как только это будет сделано, мы запустим последовательную передачу данных и подождем 10 секунд до следующего чтения:

```
// Flush input
flushSerial();
while (fona.available()) {
  Serial.write(fona.read());
}

// Wait
delay(10000);
```

Обратите внимание, что полный код этого раздела можно найти в репозитории GitHub книги по адресу https://github.com/marcoschwartz/arduino-secret-agents.

Пришло время протестировать шилд FONA. Убедитесь, что вы получили весь код и открыли его в своей Arduino IDE. Затем загрузите код на плату и откройте монитор последовательного порта. Вы должны увидеть следующий снимок экрана:



Если все работает нормально, вы должны увидеть номер IMEI SIM-карты, а затем последнее SMS-сообщение на карте. Вы можете проверить это, отправив сообщение на телефонный номер SIM-карты; он должен сразу отображаться при следующем чтении SIM-карты. Если это сработает, поздравляем! Теперь вы можете перейти к следующему разделу.

Управление реле

В этой части мы собираемся запрограммировать плату Arduino для удаленного управления реле и светодиодами, которые подключены к плате, путем отправки SMS на шилд FONA. По большей части код похож на код, который мы видели в предыдущем разделе, здесь я подробно остановлюсь только на новых элементах кода.

Нам нужно определить контакты, к которым подключены реле и светодиод:

```
#define RELAY_PIN 7
#define LED_PIN 8
```

Затем мы определим набор переменных для счетчика реле / блокировки. Это необходимо, потому что, например, после открытия блокировки мы хотим, чтобы она автоматически закрывалась по прошествии определенного времени в целях безопасности. Здесь мы используем задержку в 5 секунд:

```
bool lock_state = false;
int init_counter = 0;
int lock_counter = 0;
int lock_delay = 5000;
```

Затем мы определим переменные для подсчета количества SMS, хранящихся на карте:

```
int8_t smsnum = 0;
int8_t smsnum_old = 0;
```

Мы также установим два «пароля» для активации или деактивации реле. Я использовал здесь очень простые парольные фразы; однако вы можете использовать, который хотите:

```
String password_on = "open";
String password_off = "
close";
```

Затем в функции скетча thesetup () мы установим пины реле и светодиода в качестве выхода:

```
pinMode(RELAY_PIN, OUTPUT);
pinMode(LED_PIN, OUTPUT);
```

В функции скетча `loop ()` мы получим общее количество `SMS`, хранящихся на
SIM-карте:`smsnum = fona.getNumSMS();`

По сути, мы хотим проверить, было ли получено новое SMS-сообщение. Для этого мы сравним это новое чтение со старым количеством SMS, хранящимся на SIM-карте:

```
if (smsnum > smsnum_old) {
```

В этом случае мы сохраняем сообщение в объекте String, чтобы мы могли выполнять с ним некоторые операции:

```
String message = String(replybuffer);
```

Затем, если полученное сообщение совпадает с паролем «On», который мы установили ранее, мы активируем реле и светодиод, а также запустим счетчик:

```
if (message.indexOf(password_on) > -1) {
  Serial.println("Lock OPEN");
  digitalWrite(RELAY_PIN, HIGH);
  digitalWrite(LED_PIN, HIGH);
  lock_state = true;
  init_counter = millis();
}
```

Мы бы сделали что-то подобное, если бы получили сообщение «Off»:

```
if (message.indexOf(password_off) > -1) {
    Serial.println("Lock CLOSE");
    digitalWrite(RELAY_PIN, LOW);
    digitalWrite(LED_PIN, LOW);
    lock_state = false;
  }
```

После этого обновим счетчик:

```
lock_counter = millis();
```

Затем мы проверяем, прошла ли задержка с момента последней активации реле / блокировки. В этом случае мы автоматически отключим реле / закроем замок:

```
if (lock_state == true andand (lock_counter - init_counter) > lock_
delay) {
  Serial.println("Lock CLOSE");
  digitalWrite(RELAY_PIN, LOW);
  digitalWrite(LED_PIN, LOW);
  lock_state = false;
}
```

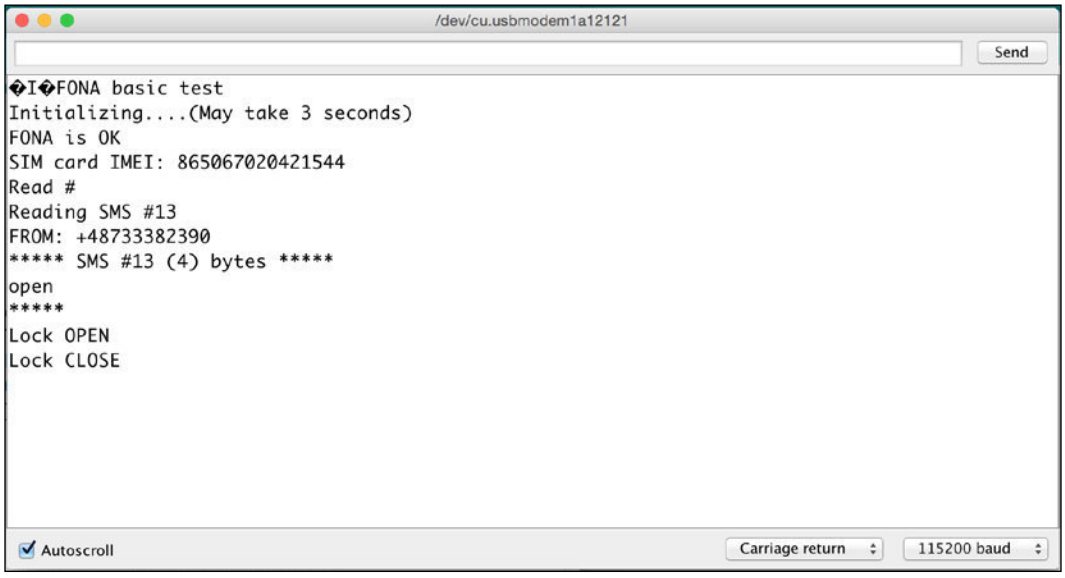
Наконец, мы сохраним текущее количество полученных текстовых сообщений в переменной smsnum_old:

```
smsnum_old = smsnum;
```

Обратите внимание, что полный код можно найти в репозитории GitHub проекта по адресу `https://github.com/marcoschwartz/arduino-secret-agents`.

Пора опробовать проект. Возьмите весь код и вставьте его в IDE Arduino. Убедитесь, что вы также изменили пароли включения / выключения в коде.

Затем загрузите код на плату и откройте монитор последовательного порта. Вы должны увидеть, что экран инициализируется, а затем ожидает нового текстового сообщения. Теперь отправьте текстовое сообщение на экран с сообщением, соответствующим паролю «On», который вы определили ранее. Вы должны увидеть следующий снимок экрана:



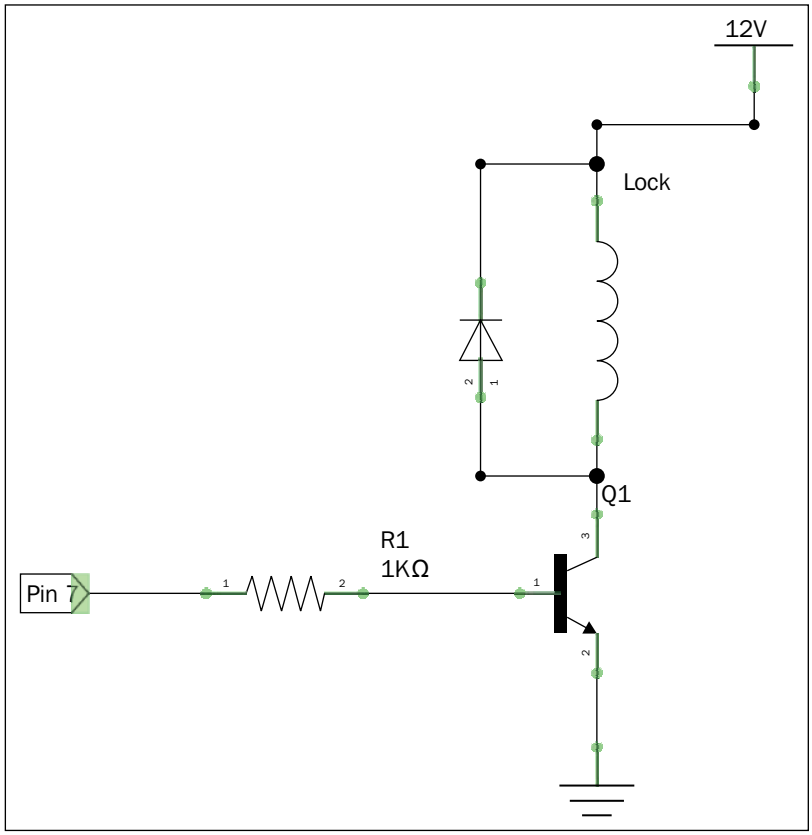
Вы также должны увидеть, как загорится светодиод и активируется реле. Вы можете либо подождать, пока система не деактивируется, либо просто повторно отправить кодовую фразу «Off».

Открытие и закрытие замка

Теперь, когда нам удалось фактически управлять реле и светодиодом, отправляя текстовые сообщения на дисплей FONA, мы можем подключить фактический замок. Этот последний шаг проекта не является обязательным, поскольку вы уже протестировали основные функции проекта, а подключение электронного дверного замка является немного более техническим. Это изображение замка, который я использовал:

Во-первых, вам нужно отрезать разъем JST от электронного замка, чтобы можно было подключить его к макетной плате. Я припаял 2-контактный разъем на конце электронного замка, чтобы он был совместим с моей макетной платой.

Затем вам нужно будет собрать оставшиеся дополнительные компоненты на макетной плате, как показано на следующей схеме:



Давайте посмотрим, как собрать эти компоненты вместе. Сначала поместите все компоненты на макетную плату. Затем подключите цифровой вывод 7 платы Arduino к базе транзистора через резистор 1 кОм. Также подключите эмиттер транзистора к земле проекта. Затем подключите электронный дверной замок между коллектором транзистора и источником питания 12 В. Наконец, установите выпрямительный диод параллельно дверному замку с ориентацией, указанной на схеме.

Тогда пора снова протестировать проект. Вы можете использовать тот же код, что и в предыдущем разделе. Поздравляем, теперь вы можете управлять дверным замком, отправив секретную кодовую фразу по SMS!

Резюме

В этой главе мы создали еще один очень полезный проект для секретных агентов; проект, в котором можно открыть дверной замок, отправив секретный код по SMS. Вы также можете использовать этот проект для многих других целей. Например, вы можете использовать тот же проект для удаленной активации сигнала тревоги, когда проект получает текстовое сообщение.

Обратите внимание, что вы также можете использовать дисплей FONA для отправки текстовых сообщений, что открывает дверь для еще более интересных проектов!

В следующей главе мы увидим, как создать типичный проект секретного агента: удаленную шпионскую камеру, за которой можно наблюдать из любой точки мира!

6

Создание облачной шпионской камеры

Теперь мы собираемся создать очень известный проект для секретных агентов: шпионскую камеру. Это будет камера, которую можно, например, установить в комнате за книгами и помочь вам контролировать комнату из другого места.

Мы собираемся построить два проекта на одном и том же оборудовании. Это будут ключевые темы, затронутые в этой главе:

- Первым проектом будет шпионская камера, которая делает снимок каждый раз, когда перед ним обнаруживается движение, и загружает его в вашу учетную запись Dropbox. После этого фотографии будут доступны шпиону из любой точки мира.

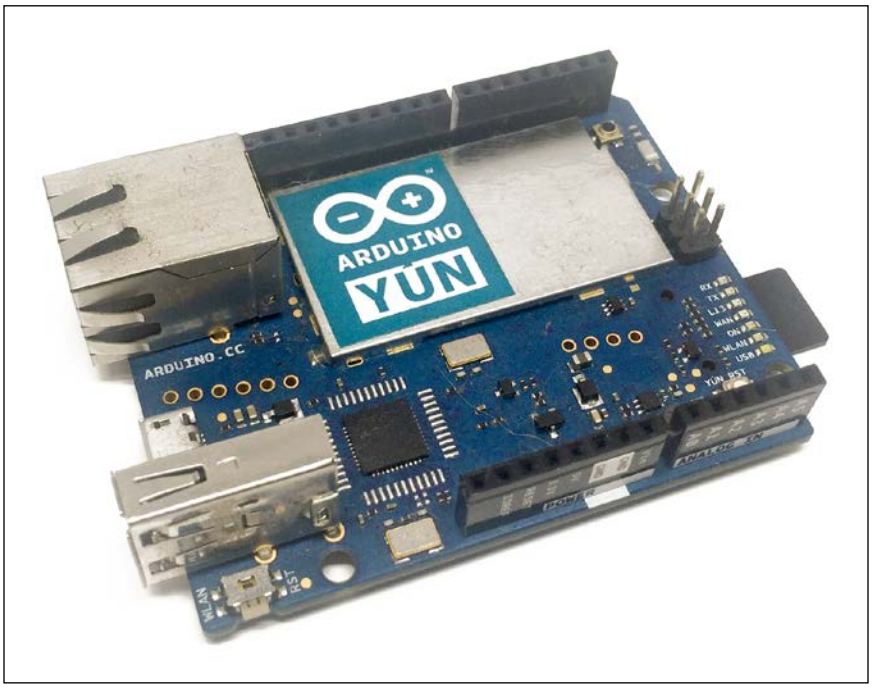
- Наконец, мы закончим главу, заставив камеру транслировать видео в реальном времени в локальной сети i-Fi. Это идеально подходит для шпиона, который хочет увидеть, что происходит в помещении, будучи спрятанным в другом помещении или снаружи.

Компоненты и программы

Во-первых, давайте посмотрим, какие компоненты необходимы для этого проекта.

На этот раз мы будем использовать не плату Arduino Uno, а плату Arduino Yun. Нам нужно не только подключение к Wi-Fi, но и встроенный USB-порт Yun. Это значительно упростит использование USB-камеры в нашем проекте.

Ниже приводится плата Arduino, которую я использовал для этого проекта.:



Тогда вам понадобится USB-камера. Вам нужна камера, совместимая с классом видео USB (UVC). В основном, самые последние USB-камеры совместимы с этим стандартом. Я рекомендую USB-камеру Logitech C270, которую я использовал для этого проекта:

Наконец, вам также понадобится датчик движения PIR, чтобы определять, есть ли движение перед камерой. Подойдет любая марка, вам просто нужно, чтобы она была совместима с уровнем 5В.
Это датчик, который я использовал в этом проекте:

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

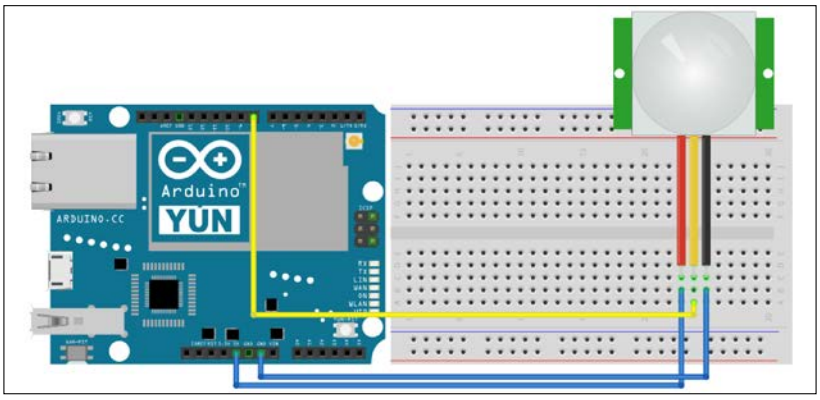
- Arduino Yun
- USB камера
  webcam-c270)
- PIR Датчик движения
- Карта microSD (по крайней мере 2 GB)
- Макетная плата
- перемычки

Что касается программного обеспечения, вам понадобится только Arduino IDE. Если вы используете Windows, вам также понадобится терминальное программное обеспечение. Я рекомендую использовать PuTTY, который вы можете скачать с http://www.putty.org/.

Конфигурация оборудования

Теперь давайте настроим оборудование для этого проекта. Это будет очень просто, и мы также настроим Wi-Fi соединение Yun.

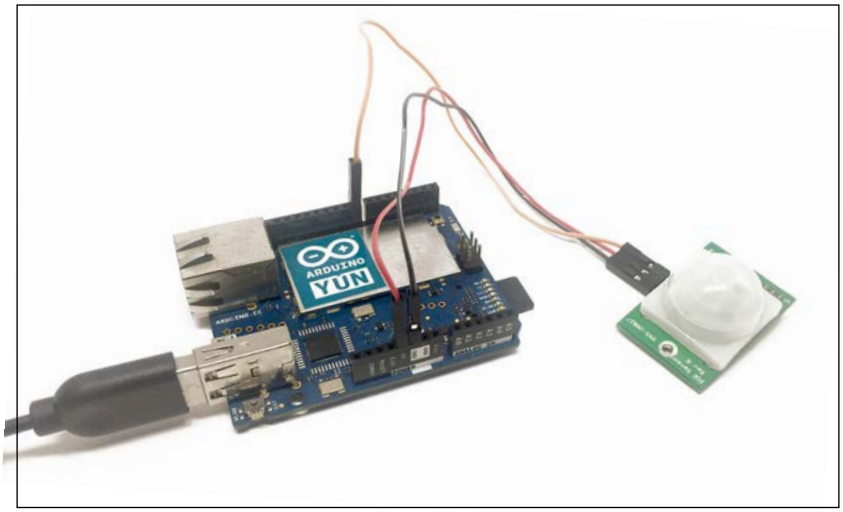
Это схема, которая поможет вам (USB-камера отсутствует)



Вам просто нужно подключить датчик движения PIR к Юню. Подключите VCC к контакту 5V Arduino, GND к GND, а выход датчика к контакту 8 Arduino Yun.

Наконец, вставьте USB-камеру в USB-порт, а карту microSD - в Arduino Yun.

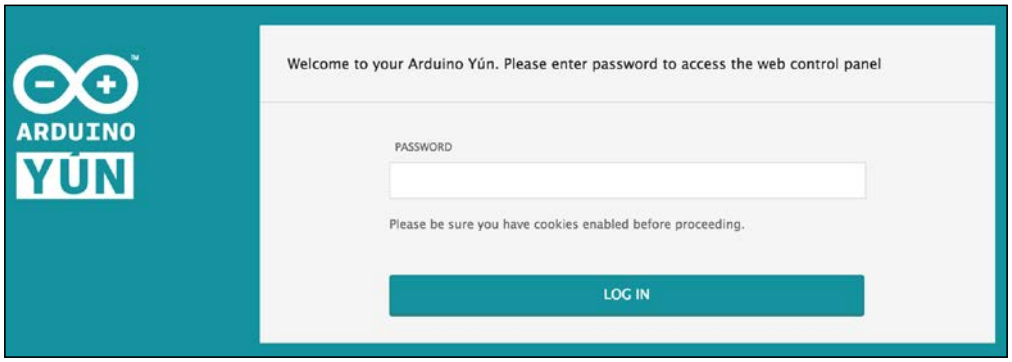
Вот как это должно выглядеть в итоге:



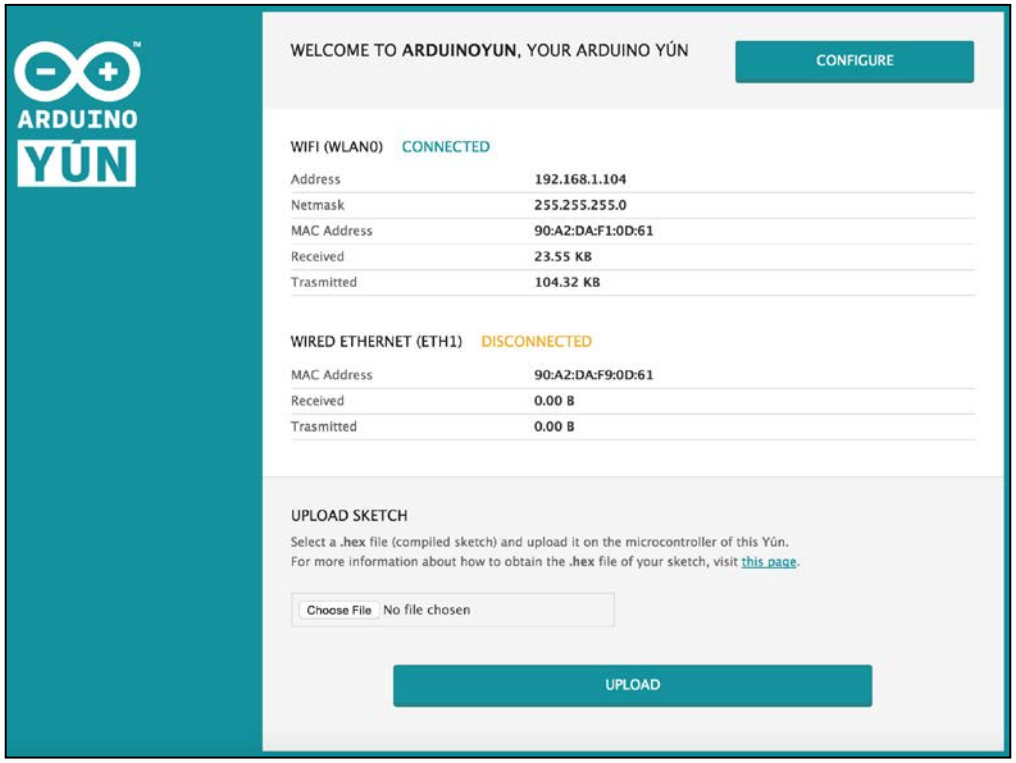
Вот как это должн

Теперь мы собираемся настроить Yun, чтобы он подключался к вашей сети Wi-Fi. Для этого лучше всего следовать последним инструкциям от Arduino, которые доступны на https://www.arduino.cc/en/Guide/ArduinoYun.

Затем вы сможете подключиться к своему Yun через ваш любимый веб-браузер и получить к нему доступ с паролем, который вы установили ранее:



После этого вы увидите, что ваш Yun работает.:



Теперь мы войдем в него из терминала, чтобы установить некоторые модули. Если вы работаете в Windows, я рекомендую использовать PuTTY для ввода этих команд. Начните с подключения вашего Yun (замените адрес на адрес вашего Yun) следующим образом:

```
ssh root@arduinoyun.local
```

Затем вас встретит следующее сообщение:

Теперь введите следующую команду:

```
opkg update
```

Когда это будет сделано, введите:

```
opkg install kmod-video-uvc
```

Затем введите:

```
opkg install fswebcam
```

Наконец, введите следующую команду:

```
opkg install mjpg-streamer
```

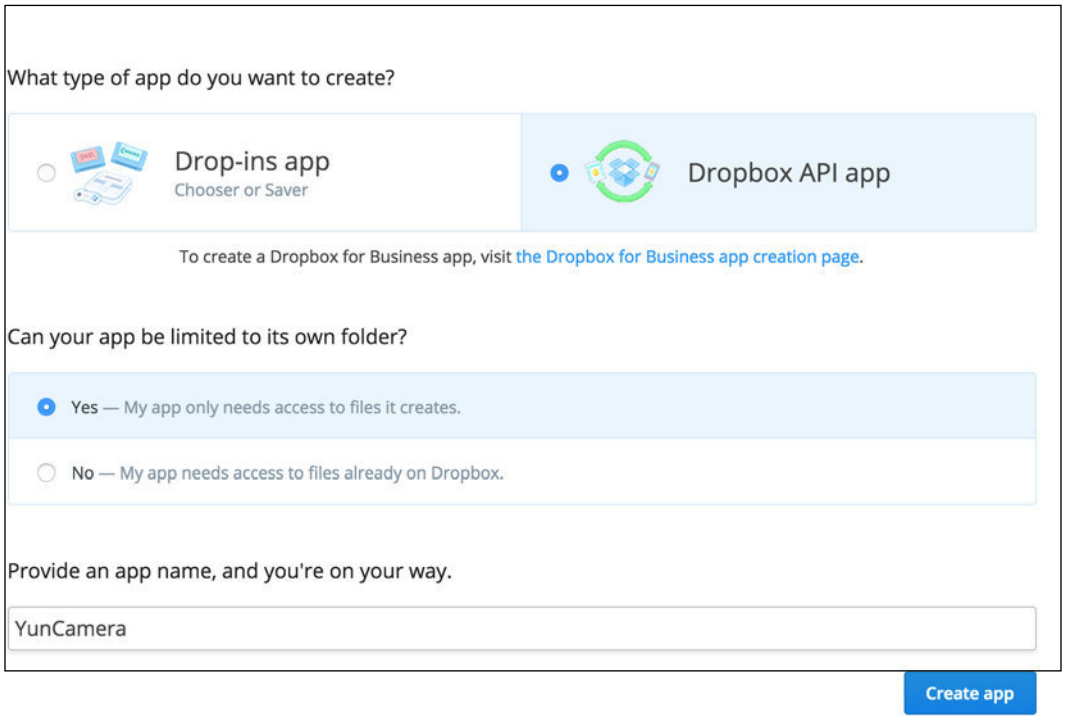
Поздравляем, ваш Yun теперь полностью готов к работе в этом проекте!

Настройка учетной записи Dropbox

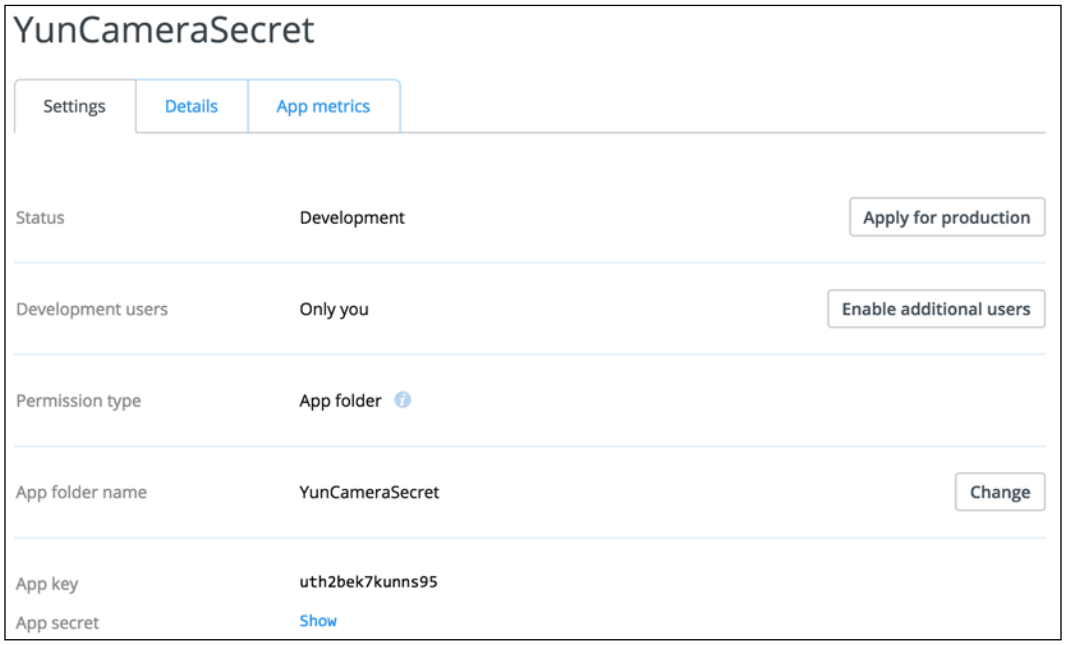
Пришло время настроить вашу учетную запись Dropbox. Во-первых, убедитесь, что у вас действительно есть учетная запись, просто посетив веб-сайт Dropbox. Затем нам нужно создать приложение Dropbox. Для этого перейдите на https://www.dropbox.com/developers/apps.

Затем вы можете создать новое приложение с помощью кнопки **Create ap**

Дайте ему имя и убедитесь, что он установлен, как показано на следующем снимке экрана:



Теперь в параметрах приложения нужны две вещи: ключ приложения и секрет приложения. Вы можете найти их обоих на одной странице:



Получив их, вы можете перейти к следующему шагу и настроить свою учетную запись Temboo.

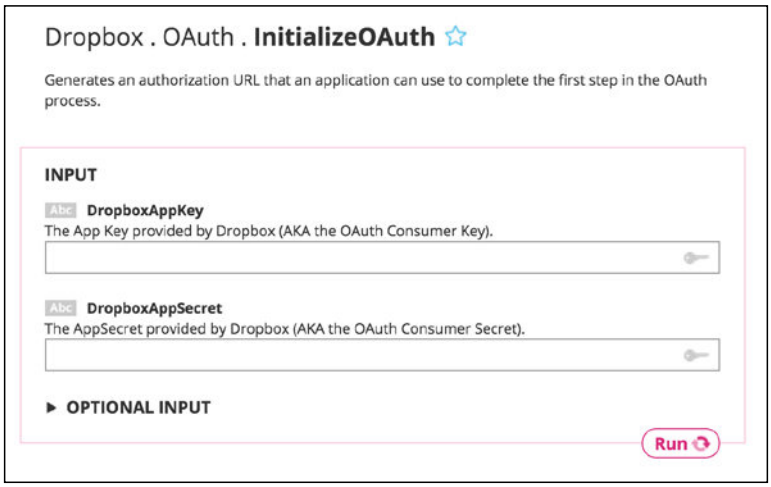
Настройка учетной записи Temboo

Мы собираемся использовать службу Temboo, чтобы связать ваше оборудование с приложением Dropbox, которое мы только что создали. Это позволит нам загружать файлы в Dropbox.

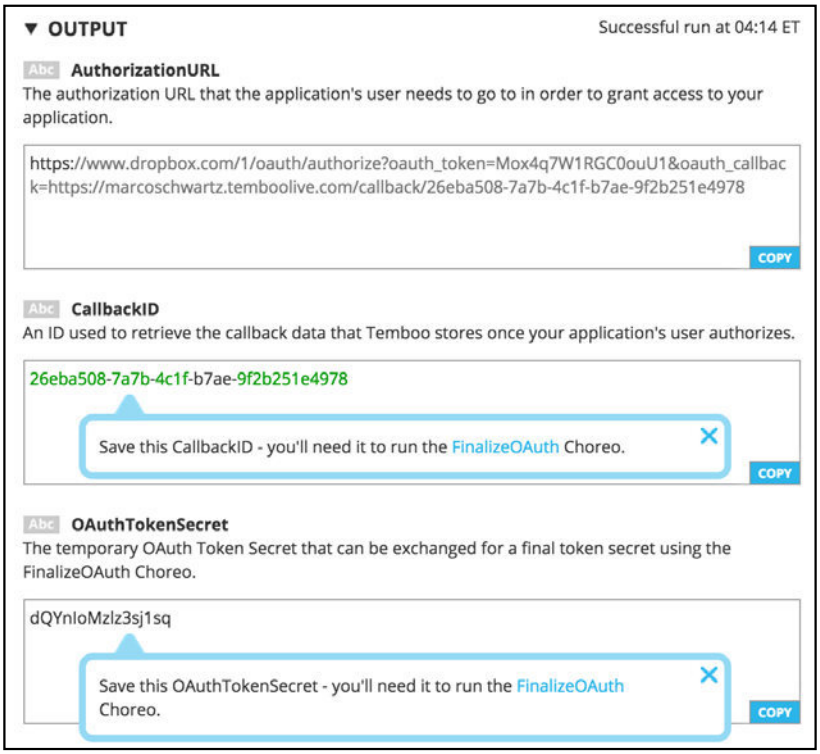
Сначала вам нужно создать новую учетную запись Temboo по следующему URL-адресу: https://www.temboo.com/library/.

Затем нам нужно фактически авторизовать нашу учетную запись `Temboo` (и, следовательно, наш проект `Arduino`), чтобы использовать ваше приложение `Dropbox`. Для этого перейдите в https://www.temboo.com/library/Library/Dropbox/OAuth/InitializeOAuth/.

Вам будет предложено ввести App key (ключ приложения) Dropbox и App secret (секрет приложения):

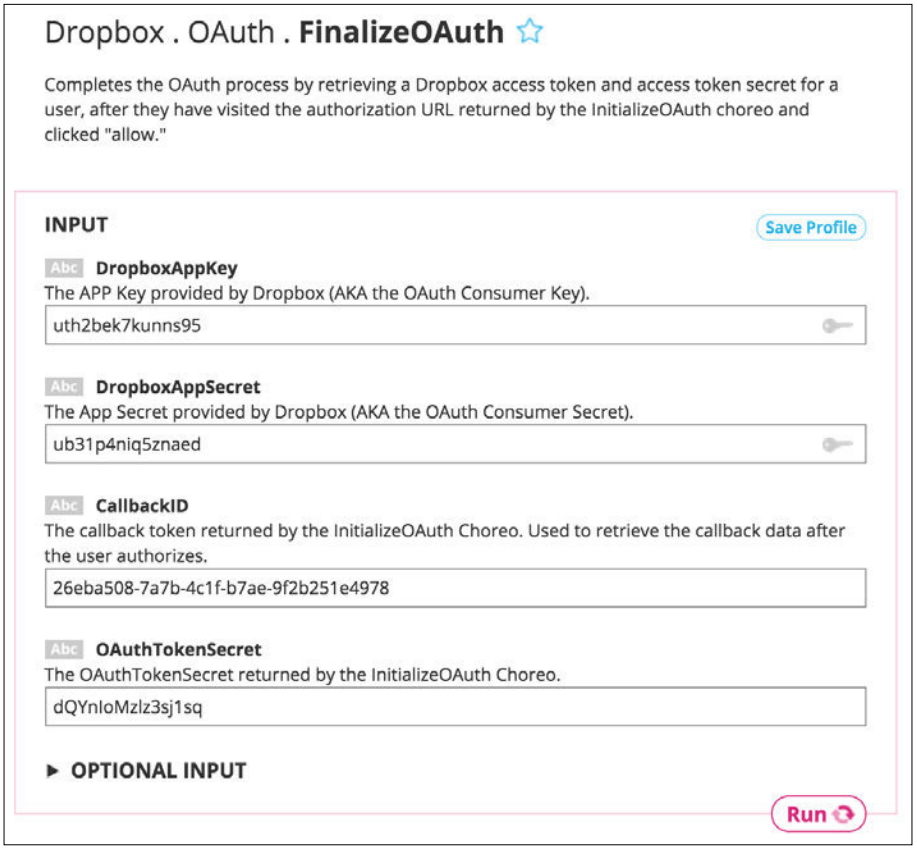


После того, как вы нажмете «Run» (Выполнить), вам нужно будет сделать две вещи. Во-первых,вам нужно перейти по ссылке, предоставленной вам Temboo:

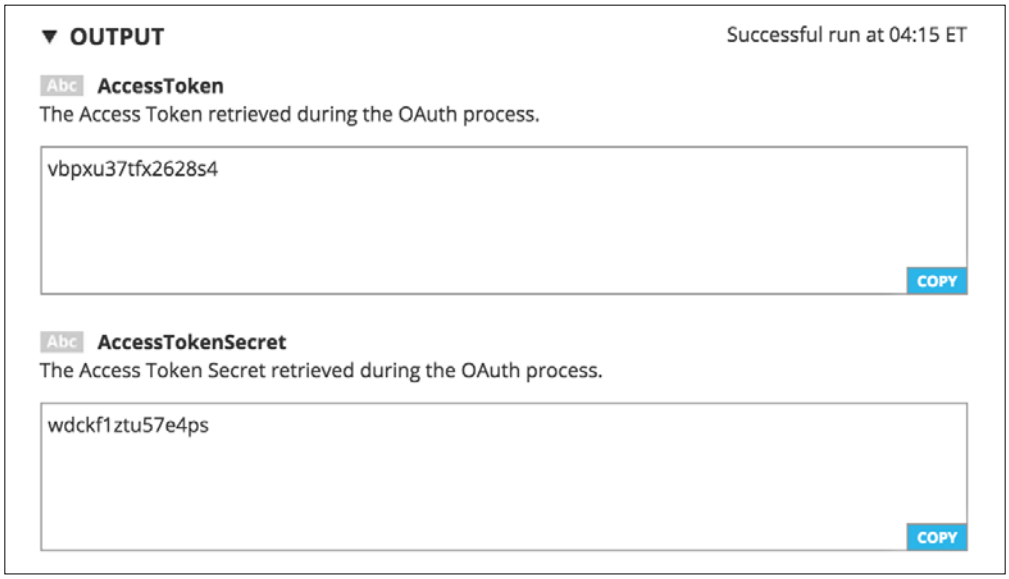


После этого вам нужно взять CallbackID и OAuthTokenSecret и перейти на страницу по адресу https://www.temboo.com/library/Library/Dropbox/OAuth/. FinalizeOAuth /.

На этой странице вы можете ввести всю информацию, которую вы получили на данный момент:



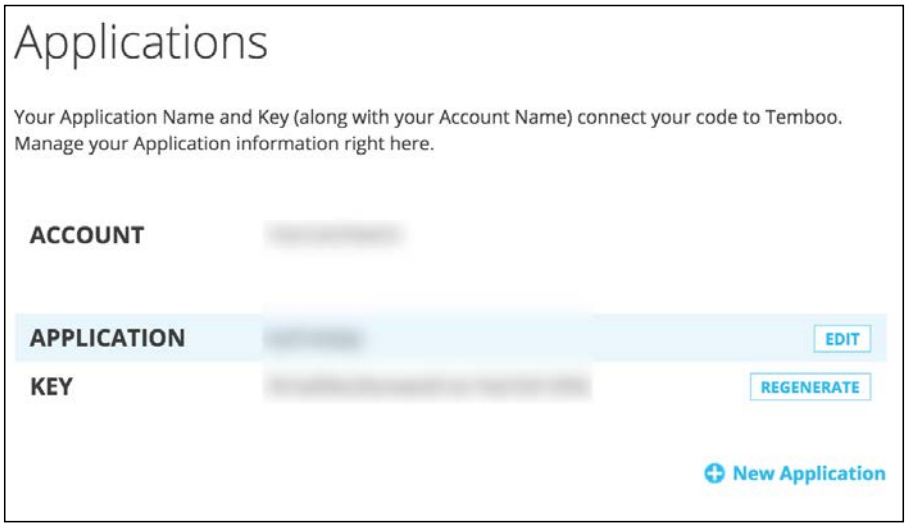
Вам будет предоставлен токен доступа и секрет токена, который вам понадобится для проекта шпионской камеры:



Note that the Dropbox API is subject to change in the future. Therefore, always check the Temboo page and follow the instructions given there if they are different from the ones presented in this book.

You just need one more thing from Temboo; some data about your account. Go to `https://www.temboo.com/account/applications/`.

Здесь вы можете увидеть информацию о приложении, которое вы создали при открытии учетной записи:



Оставьте это открытым, эта информация вам также понадобится позже.

Сохранение изображений в Dropbox

Наконец, мы собираемся создать наше первое приложение с использованием созданного нами оборудования. Здесь будут две части: скетч Arduino и скрипт Python. Скетч Arduino будет делать снимок в случае обнаружения движения и вызывать скрипт Python. Сценарий Python фактически загружает изображения в Temboo каждый раз, когда он вызывается скетчом Arduino.

Это полный скетч Arduino:

```
// Скетч для загрузки изображений в Dropbox при обнаружении движения
#include <Bridge.h>
#include <Process.h>//

//Процесс изображения
Process picture;

// Имя файла
String filename;

// Pin
```

```
int pir_pin = 8;

// Путь
String path = "/mnt/sda1/";

void setup() {

  // Bridge
  Bridge.begin();

  // Установить режим вывода
  pinMode(pir_pin,INPUT);
}

void loop(void)
{

  if (digitalRead(pir_pin) == true) {

    // Сгенерировать имя файла с отметкой времени
    filename = "";
    picture.runShellCommand("date +%s");
    while(picture.running());

    while (picture.available()>0) {
      char c = picture.read();
      filename += c;
    }
    filename.trim();
    filename += ".png";

    // Сделать фотографию
    picture.runShellCommand("fswebcam " + path + filename + " -r
1280x720");
    while(picture.running());

    // Загрузить в Dropbox
    picture.runShellCommand("python " + path + "upload_picture.py " +
path + filename);
    while(picture.running());
  }
}
```

Давайте посмотрим, какие части этого скетча самые важные. Во-первых, вам нужно включить необходимые библиотеки:

```
#include <Bridge.h>
#include <Process.h>
```

Затем мы определим путь к SD-карте, на которой будут храниться изображения:

```
String path = "/mnt/sda1/";
```

После этого мы инициализируем экземпляр Bridge, который позволит нам использовать файловую систему Yun, например:

```
Bridge.begin();
```

По-прежнему в функции setup () скетча, мы установим контакт датчика движения в качестве входа:

```
pinMode(pir_pin,INPUT);
```

После этого в функции скетча loop () мы проверим, обнаружил ли датчик движения движение:

```
if (digitalRead(pir_pin) == true) {
```

Если это так, мы сначала создадим имя файла для нового изображения, используя текущую дату и время:

```
filename = "";
   picture.runShellCommand("date +%s");
while(picture.running());

while (picture.available()>0) {
     char c = picture.read();
  filename += c;
}
filename.trim();
filename += ".png";
```

Затем мы воспользуемся утилитой fswebcam, чтобы сохранить это изображение на SD-карту:

```
picture.runShellCommand("fswebcam " + path + filename + " -r
1280x720");
while(picture.running());
```

И, наконец, мы вызовем скрипт Python для фактической загрузки изображения в Dropbox:

```
picture.runShellCommand("python " + path + "upload_picture.py " + path
+ filename);
while(picture.running());
```

Теперь давайте посмотрим на скрипт Python. Ниже приводится полный сценарий:

```
# coding=utf-8
# Script to upload files to Dropbox

# Import correct libraries
import base64
import sys
from temboo.core.session import TembooSession
from temboo.Library.Dropbox.FilesAndMetadata import UploadFile

print str(sys.argv[1])

# Encode image
with open(str(sys.argv[1]), "rb") as image_file:
    encoded_string = base64.b64encode(image_file.read())

# Declare Temboo session and Choreo to upload files
session = TembooSession('yourSession', 'yourApp', 'yourKey')
uploadFileChoreo = UploadFile(session)

# Get an InputSet object for the choreo
uploadFileInputs = uploadFileChoreo.new_input_set()

# Set inputs
uploadFileInputs.set_AppSecret("yourAppSecret")
uploadFileInputs.set_AccessToken("yourAccessToken")
uploadFileInputs.set_FileName(str(sys.argv[1]))
uploadFileInputs.set_AccessTokenSecret("yourTokenSecret")
uploadFileInputs.set_AppKey("yourAppKey")
uploadFileInputs.set_FileContents(encoded_string)
uploadFileInputs.set_Root("sandbox")

# Execute choreo
uploadFileResults = uploadFileChoreo.execute_with_
results(uploadFileInputs)
```

Теперь давайте посмотрим на самые важные части этого скрипта. Сначала мы импортируем библиотеки Temboo:

```
import base64
import sys
from temboo.core.session import TembooSession
from temboo.Library.Dropbox.FilesAndMetadata import UploadFile
```

Вам также необходимо будет указать данные своей учетной записи Temboo:

```
session = TembooSession('yourSession', 'yourApp', 'yourKey')
uploadFileChoreo = UploadFile(session)
```

Затем мы создадим новый набор входов для библиотеки Dropbox:

```
uploadFileInputs = uploadFileChoreo.new_input_set()
```

После этого вам нужно будет ввести все ключи, которые мы получили от Dropbox и Temboo:

```
uploadFileInputs.set_AppSecret("yourAppSecret")
uploadFileInputs.set_AccessToken("yourAccessToken")
uploadFileInputs.set_FileName(str(sys.argv[1]))
uploadFileInputs.set_AccessTokenSecret("yourTokenSecret")
uploadFileInputs.set_AppKey("yourAppKey")
uploadFileInputs.set_FileContents(encoded_string)
uploadFileInputs.set_Root("sandbox")
```

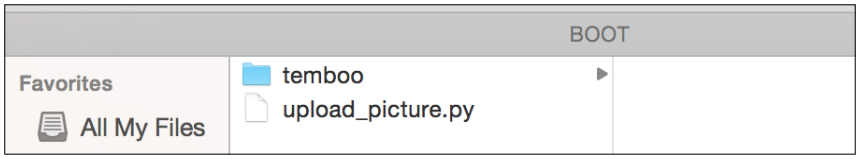
Наконец, мы выполним загрузку файла в Dropbox:

```
uploadFileResults = uploadFileChoreo.execute_with_
results(uploadFileInputs)
```

Пришло время протестировать проект! Обратите внимание: весь код можно найти в репозитории GitHub по адресу https://github.com/marcoschwartz/arduino-secret-agents.

Вам по-прежнему потребуется загрузить библиотеку Temboo Python с https://temboo.com/sdk/python.

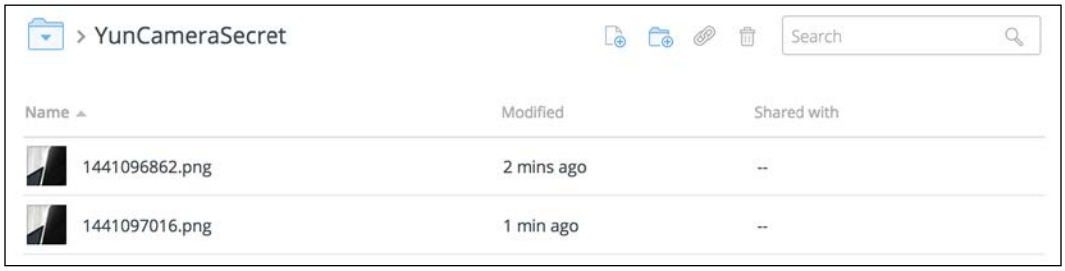
Затем убедитесь, что вы изменили файлы Python своими данными. Кроме того, снова вставьте SD-карту в компьютер с помощью адаптера. Поместите файл и SDK Temboo на SD-карту, как показано на следующем снимке экрана:



После этого вставьте SD-карту обратно в Yun. Откройте эскиз Arduino с помощью Arduino IDE и убедитесь, что вы выбрали плату Arduino Yun. Теперь загрузите эскиз на плату.

Теперь попробуйте провести рукой перед датчиком движения; датчик должен стать красным, и вы также должны заметить, что камера сразу становится активной (на камере Logitech C270 загорится маленький зеленый светодиод).

Вы также можете проверить свою учетную запись Dropbox в папке «Applications» (Приложения). Это должна быть новая созданная папка, содержащая снимки, сделанные шпионской камерой:



Поздравляем, вы создали свой первый проект шпионской камеры! Обратите внимание, что к этой папке, конечно, можно получить доступ из любой точки мира, поэтому, даже если вы находитесь на другом конце города, вы можете следить за тем, что происходит в комнате, где находится камера.

Прямая трансляция со шпионской камеры

Сейчас мы собираемся закончить эту главу более коротким проектом: использование камеры для потоковой передачи видео в реальном времени в веб-браузере. Этот поток будет доступен с любого устройства, подключенного к той же сети Wi-Fi, что и Yun.

Чтобы начать работу с этим проектом, войдите в свой Yun с помощью следующей команды (изменив имя платы на имя вашего Yun):

```
ssh root@arduinoyun.local
```

Затем введите следующее:

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 25" -o
"output_http.so -p 8080 -w /www/webcam" &
```

Это запустит трансляцию с вашего Yun. Теперь вы можете просто перейти по URL-адресу вашего Yun и добавить: 8080 в конце, например, http: //arduinoyun.local: 8080.

Вы попадете в потоковый интерфейс:

Теперь вы можете транслировать это видео в прямом эфире на свой мобильный телефон или любое другое устройство в той же сети. Это идеальный проект, чтобы, например, шпионить за комнатой, когда вы находитесь на улице.

Резюме

В этом проекте мы создали проект шпионской камеры, которая может отправлять изображения в Облако при обнаружении движения. Мы также увидели, что он может делать и другие вещи, например, транслировать живое видео в веб-браузере.

Конечно, есть много способов улучшить этот проект. Вы можете, например, установить несколько из этих шпионских камер в здании и заставить их делать снимки, которые вы определяете в коде Arduino (при создании имени файла).

В следующей главе мы собираемся создать еще одно захватывающее приложение для секретных агентов: проект, который позволит вам отслеживать секретные данные из любой точки мира!

7

Мониторинг секретных данных из любого места

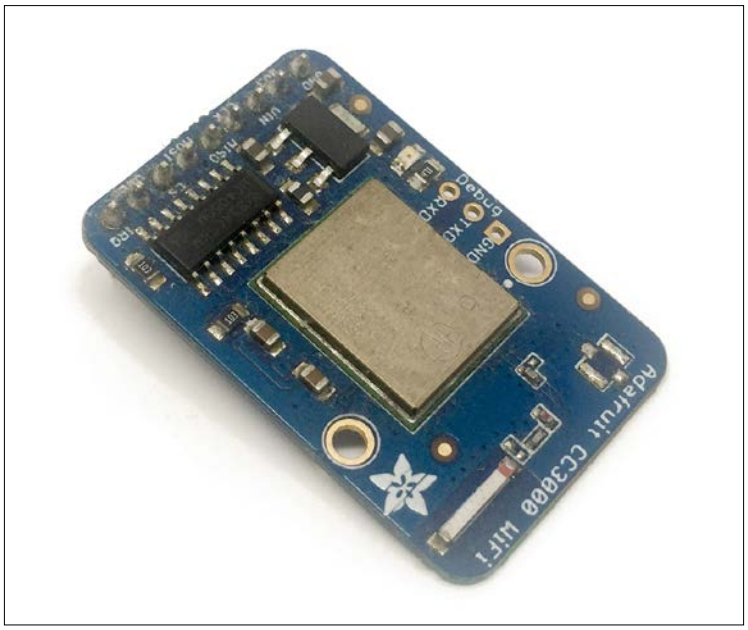
В этой главе мы собираемся создать проект, который будет непрерывно записывать данные с датчиков и отправлять эти данные по Wi-Fi, чтобы они были доступны из любого веб-браузера. Это отлично подходит для секретного агента, который хочет контролировать комнату удаленно, без своего участия. Вы, конечно, сможете адаптировать проект со своими собственными датчиками, в зависимости от того, что вы хотите записать.

Для этого в этой главе мы собираемся предпринять следующие шаги:

- Мы будем использовать Arduino вместе с чипом CC3000 Wi-Fi, который довольно удобен для обеспечения возможности подключения к Wi-Fi для проектов Arduino.
- Мы отправим данные датчиков в онлайн-сервис под названием dweet.io, а затем отобразим результат на панели управления с помощью Freeboard.io.
- Наконец, мы также увидим, как настроить автоматические оповещения на основе записанных данных.

Компоненты и программы

Сначала посмотрим, какие компоненты необходимы для этого проекта.
Мы, конечно, будем использовать Arduino Uno как мозг проекта. Для подключения к Wi-Fi мы будем использовать коммутационную плату CC3000 от Adafruit.



Мы также будем использовать несколько датчиков, чтобы проиллюстрировать работу проекта: датчик DHT11 для температуры и влажности, фотоэлемент для уровня освещенности и датчик движения..
Fнаконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

- Arduino Uno)
- Коммутационная плата CC3000
- Датчик DHT11 с резистором 4,7 кОм
- Фотоэлемент
- Резистор 10 кОм
- PIR Датчик движения
- Макетная плата
- Перемычки

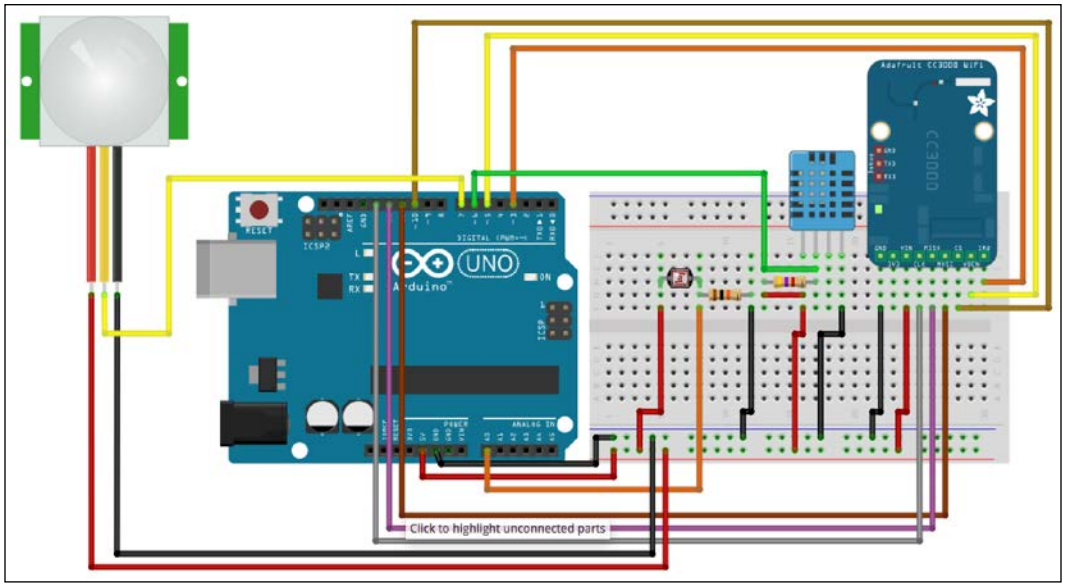
Что касается программного обеспечения, вам понадобится последняя версия Arduino IDE. Также вам понадобятся следующие библиотеки:

- Adafruit CC3000 library
- Adafruit DHT library

Чтобы установить эти библиотеки, просто используйте менеджер библиотек Arduino.

Конфигурация оборудования

Теперь давайте соберем различные компоненты этого проекта. Эта схема поможет вам:



Сначала подключите питание. Подключите Arduino Uno 5V к красной шине питания на макетной плате, а контакт GND к синей шине питания. Кроме того, разместите все основные компоненты на макетной плате.

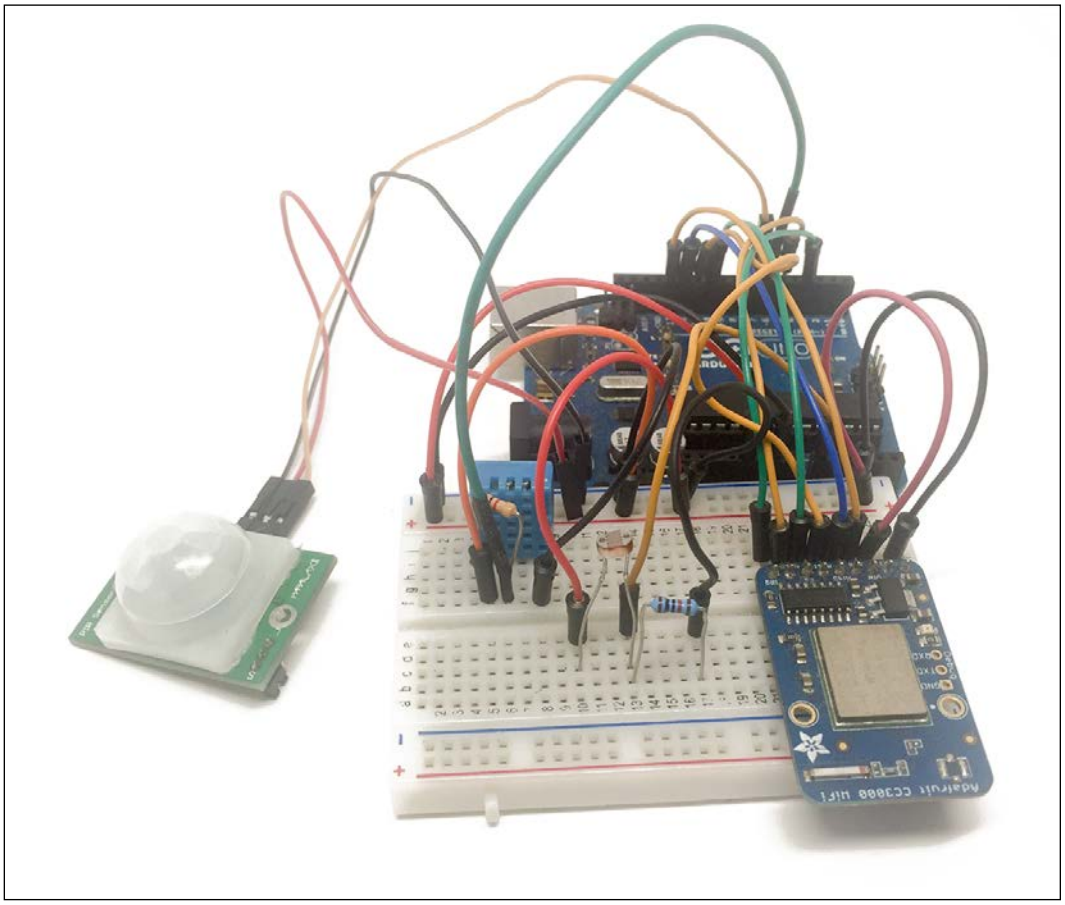
После этого для датчика DHT11 следуйте инструкциям на схеме, чтобы подключить датчик к плате Arduino. Убедитесь, что вы не забыли резистор 4,7 кОм между VCC и сигнальными контактами.

Теперь мы собираемся подключить фотоэлемент. Начните с размещения фотоэлемента на макете последовательно с резистором 10 кОм. После этого подключите другой конец фотоэлемента к красной шине питания, а другой вывод резистора - к синей шине питания. Наконец, подключите контакт между фотоэлементом и резистором к контакту A0 Arduino Uno.

Наконец, для датчика движения подключите контакт VCC к красной шине питания, контакт GND к синей шине питания и, наконец, выходной контакт датчика к контакту 7 Arduino.

Теперь мы собираемся подключить коммутационную плату CC3000. Подключите контакты, как показано на схеме: IRQ к контакту номер 3 платы Arduino, VBAT к контакту 5 и CS к контакту 10. После этого подключите эти контакты к плате Arduino: MOSI, MISO и CLK перейдите к контактам 11. , 12 и 13 соответственно. Наконец, подключите питание к коммутационному разъему CC3000: подключите 5 В к выводу Vin коммутационной платы, а GND к GND.

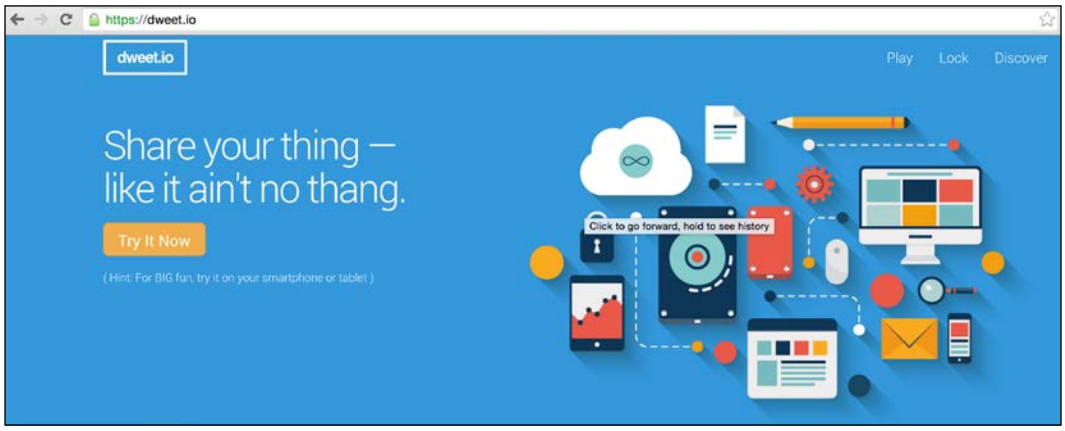
Это изображение полностью собранного проекта:



Поздравляем, ваш проект полностью собран! Можно переходить к следующему этапу: отправке данных в облако.

Отправка данных на dweet.io

Первым шагом в этом проекте действительно является отправка данных в Интернет, чтобы они хранились в сети. Для этого мы воспользуемся сервисом dweet.io. Вы можете проверить это на https://dweet.io/. Это главная страница приветствия:



Это полный код Arduino для этого проекта:

```
// Библиотеки
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "DHT.h"
#include <avr/wdt.h>

// Определение выводов микросхемы CC3000
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  5
#define ADAFRUIT_CC3000_CS    10

// DHT датчик
#define DHTPIN 6
#define DHTTYPE DHT11

// Задание параметров CC3000
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS, ADAFRUIT_
CC3000_IRQ, ADAFRUIT_CC3000_VBAT,
```

```
                                                SPI_CLOCK_DIV2); // you can
change this clock speed

// параметры DHT
DHT dht(DHTPIN, DHTTYPE);

// WLAN параметры
#define WLAN_SSID       "yourWiFiSSID"
#define WLAN_PASS       "yourWiFiPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2

// Dweet параметры
#define thing_name  "mySecretThing"

// Переменные для отправки
int temperature;
int humidity;
int light;
int motion;

uint32_t ip;

void setup(void)
{
  // Инициализировать
  Serial.begin(115200);

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

  // подключиться к сети Wi-Fi
  Serial.print(F("Connecting to WiFi network ..."));
  cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY);
  Serial.println(F("done!"));

  /* Wait for DHCP to complete */
  Serial.println(F("Request DHCP"));
  while (!cc3000.checkDHCP())
  {
    delay(100);
```

```
  }

  //Запустить сторожевой таймер
  wdt_enable(WDTO_8S);
}

void loop(void)
{

  // Измерение от DHT
  float t = dht.readTemperature();
  float h = dht.readHumidity();
  temperature = (int)t;
  humidity = (int)h;

  // Измерение уровня освещенности
  float sensor_reading = analogRead(A0);
  light = (int)(sensor_reading/1024*100);

  // Получить показания датчика движения
  motion = digitalRead(7);
  Serial.println(F("Measurements done"));

  // Сбросить сторожевой таймер
  wdt_reset();

  // Получить IP
  uint32_t ip = 0;
  Serial.print(F("www.dweet.io -> "));
  while  (ip  ==  0)  {
    if  (!  cc3000.getHostByName("www.dweet.io", &ip))  {
      Serial.println(F("Couldn't resolve!"));
    }
    delay(500);
  }
  cc3000.printIPdotsRev(ip);
  Serial.println(F(""));

  // Сбросить сторожевой таймер
  wdt_reset();

  // Проверить подключение к Wi-Fi
  Serial.print(F("Checking WiFi connection ..."));
  if(!cc3000.checkConnected()){while(1){}}
```

```
Serial.println(F("done."));
wdt_reset();

// Послать запрос
Adafruit_CC3000_Client client = cc3000.connectTCP(ip, 80);
if (client.connected()) {
  Serial.print(F("Sending request... "));

  client.fastrprint(F("GET /dweet/for/"));
  client.print(thing_name);
  client.fastrprint(F("?temperature="));
  client.print(temperature);
  client.fastrprint(F("&humidity="));
  client.print(humidity);
  client.fastrprint(F("&light="));
  client.print(light);
  client.fastrprint(F("&motion="));
  client.print(motion);
  client.fastrprintln(F(" HTTP/1.1"));

  client.fastrprintln(F("Host: dweet.io"));
  client.fastrprintln(F("Connection: close"));
  client.fastrprintln(F(""));

  Serial.println(F("done."));
} else {
  Serial.println(F("Connection failed"));
  return;
}

// Сбросить сторожевой таймер
wdt_reset();

Serial.println(F("Reading answer..."));
while (client.connected()) {
  while (client.available()) {
    char c = client.read();
    Serial.print(c);
  }
}
Serial.println(F(""));

// Reset watchdog
```

```
  wdt_reset();

  // Закрыть соединение и отключить
  client.close();
  Serial.println(F("Closing connection"));
  Serial.println(F(""));

  // Сбросить сторожевой таймер и отключить
  wdt_reset();

}
```

Теперь давайте посмотрим на самые важные части кода. Во-первых, нам нужно включить необходимые библиотеки, такие как библиотека CC3000 и библиотека DHT:

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include "DHT.h"
#include <avr/wdt.h>
```

Затем мы определяем, к какому выводу подключен датчик DHT11:

```
#define DHTPIN 6
#define DHTTYPE DHT11
```

Вам также необходимо ввести свое имя Wi-Fi и пароль:

```
#define WLAN_SSID       "yourWiFiSSID"
#define WLAN_PASS       "yourWiFiPassword"
#define WLAN_SECURITY   WLAN_SEC_WPA2
```

Затем вы можете определить имя виртуального объекта,который будет хранить данные в сети:

```
#define thing_name  "mySecretThing"
```

В функции скетча setup () мы инициализируем микросхему CC3000:

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

Так же подключаемся к сети Wi-Fi:

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

Наконец, мы инициализируем сторожевой таймер до 8 секунд. Это автоматически сбросит Arduino, если мы не обновим его до этой задержки. Это нужно для предотвращения зависания проекта:

```
wdt_enable(WDTO_8S);
```

В функции скетча loop () мы сначала измеряем данные с датчика DHT:

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

После этого измеряем уровень внешней освещенности:

```
float sensor_reading = analogRead(A0);
light = (int)(sensor_reading/1024*100);
```

И, наконец, получаем статус датчика движения:

```
motion = digitalRead(7);
```

Затем мы пытаемся получить IP-адрес сайта dweet.io:

```
while  (ip  ==  0)  {
  if  (!  cc3000.getHostByName("www.dweet.io", &ip))  {
    Serial.println(F("Couldn't resolve!"));
  }
  delay(500);
}
```

Затем мы подключаем проект к этому IP-адресу:

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

Теперь мы можем отправлять данные в формате, заданном dweet.io:

```
client.fastrprint(F("GET /dweet/for/"));
client.print(thing_name);
client.fastrprint(F("?temperature="));
client.print(temperature);
client.fastrprint(F("&humidity="));
client.print(humidity);
client.fastrprint(F("&light="));
client.print(light);
client.fastrprint(F("&motion="));
client.print(motion);
client.fastrprintln(F(" HTTP/1.1"));
```

После этого читаем ответ с сервера:

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

И закрываем соединение:

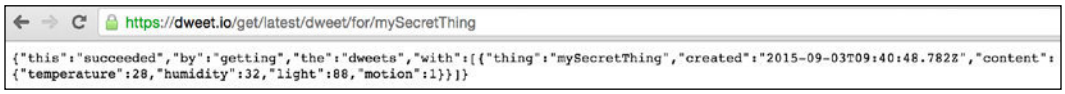
```
client.close();
```

Пришло время протестировать проект! Убедитесь, что вы получили весь код, скопировали его в среду IDE и изменили данные Wi-Fi и название объекта. Затем загрузите код на плату и откройте монитор последовательного порта:

Вы должны увидеть, что проект отправляет измерения на dweet.io, а затем получает ответ. Самая важная часть - это та, которая указывает, что данные были записаны:

```
{"this":"succeeded","by":"dweeting","the":"dweet","with":{"thing":"myS
ecretThing","created":"2015-09-03T09:38:07.051Z","content":{"temperatu
re":28,"humidity":32,"light":87,"motion":0}}}
```

Вы также можете проверить онлайн, чтобы убедиться, что данные были записаны:



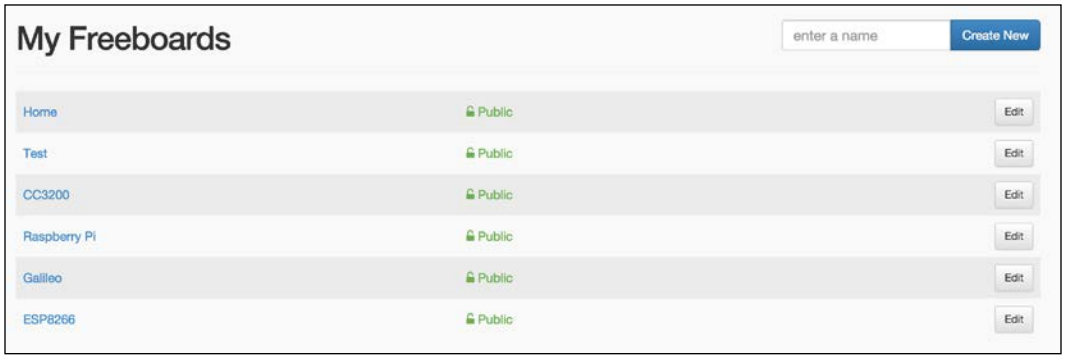
Теперь, когда мы уверены, что данные записываются, мы можем перейти к следующему шагу: шпионить за этими данными удаленно из любого веб-браузера!

Удаленный мониторинг устройства

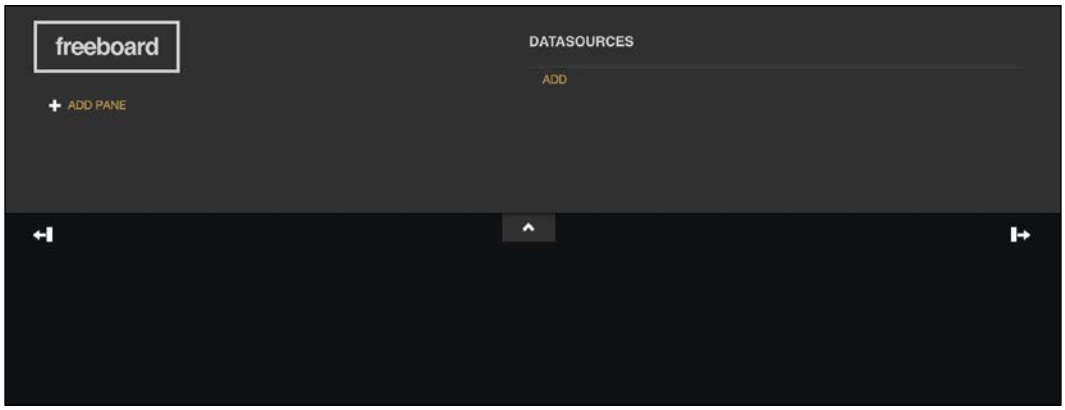
Теперь мы посмотрим, как получить доступ к данным, хранящимся на dweet.io, и отобразить их графически. Для этого мы воспользуемся сайтом под названием freeboard.io. Ты можешь идти там по следующему URL-адресу http://freeboard.io/.

Это главный экран приветствия, на котором вам нужно создать учетную запись:

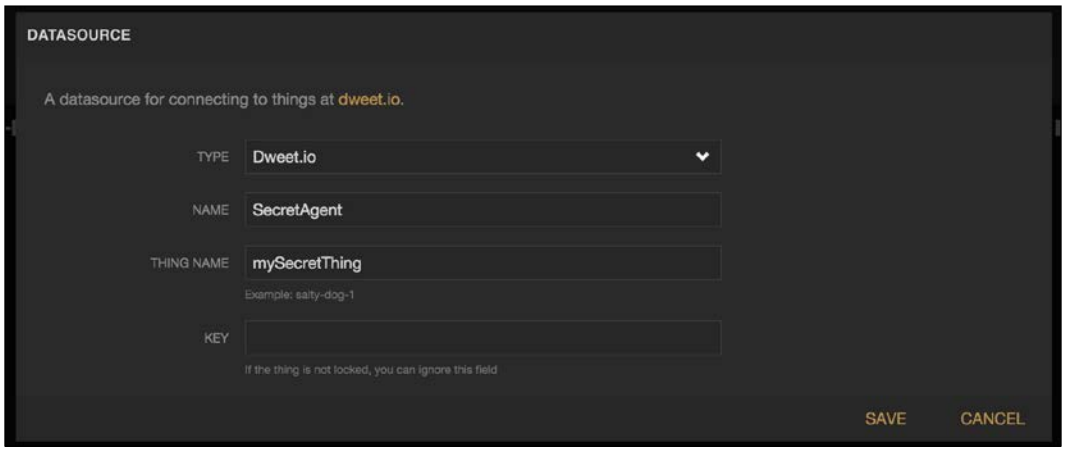
Как только у вас будет учетная запись, вы можете создать новую плату:



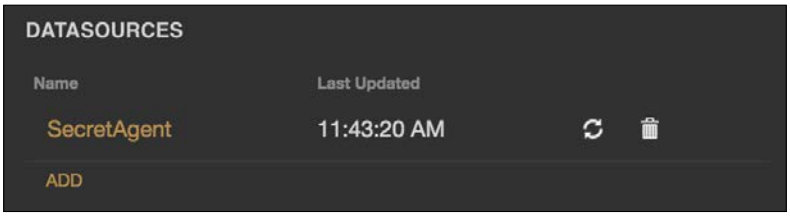
Как только это будет сделано, вы должны быть перенаправлены на аналогичную страницу с пустой доской:



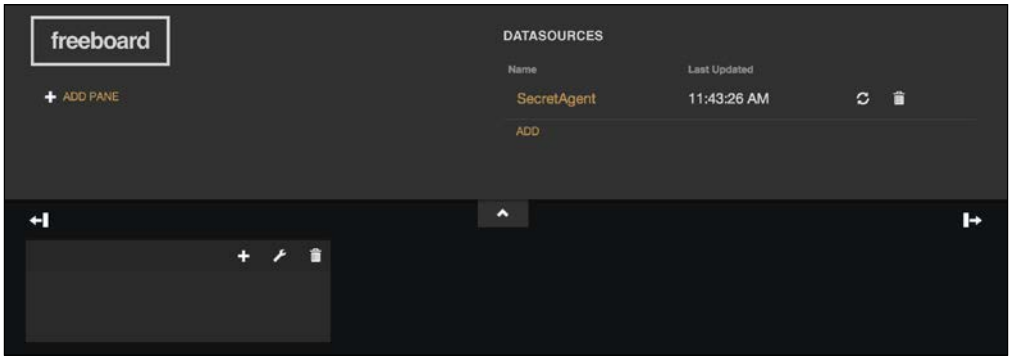
Во-первых, нам нужно установить источник данных, что означает, что нам нужно сообщить Freeboard, что нужно получать данные из того объекта, в котором мы храним данные. Добавьте новый источник данных и заполните поля, как показано на следующем снимке экрана, с именем объекта, который конечно хранит ваши данные:



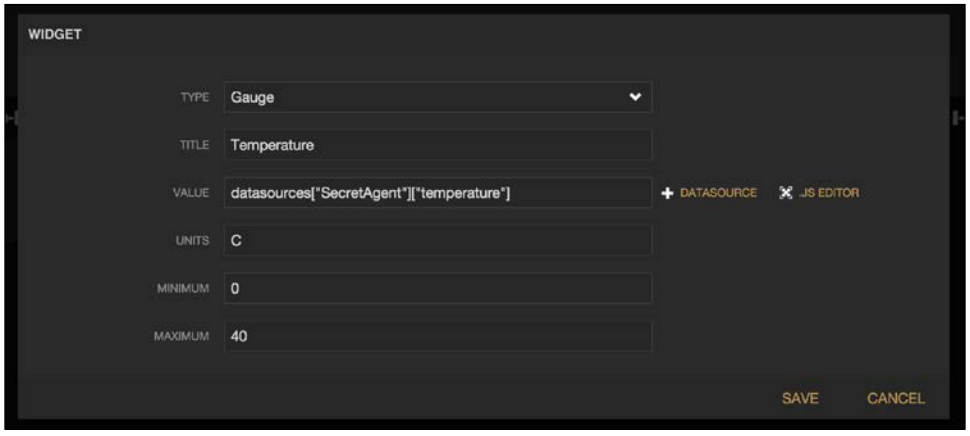
После этого вы увидите источник данных, появившийся в верхней части доски, с датой последнего обновления:



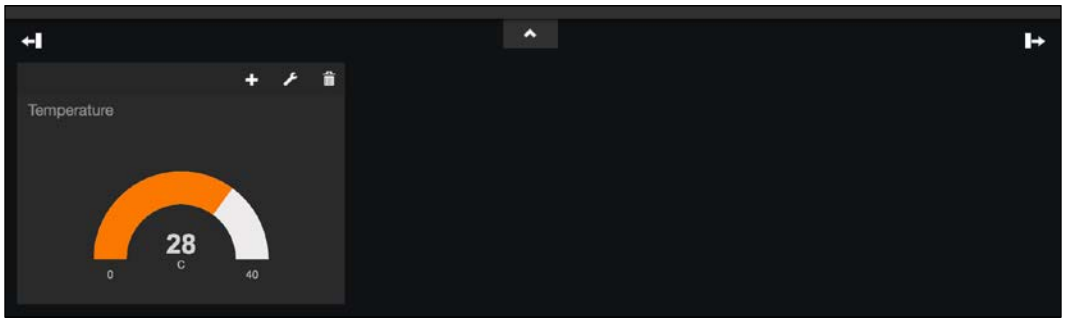
Пришло время добавить некоторые графические элементы на нашу панель управления. Сначала мы добавим единицу для температуры. Нажмите на новую панель, которая создаст новый блок внутри панели:



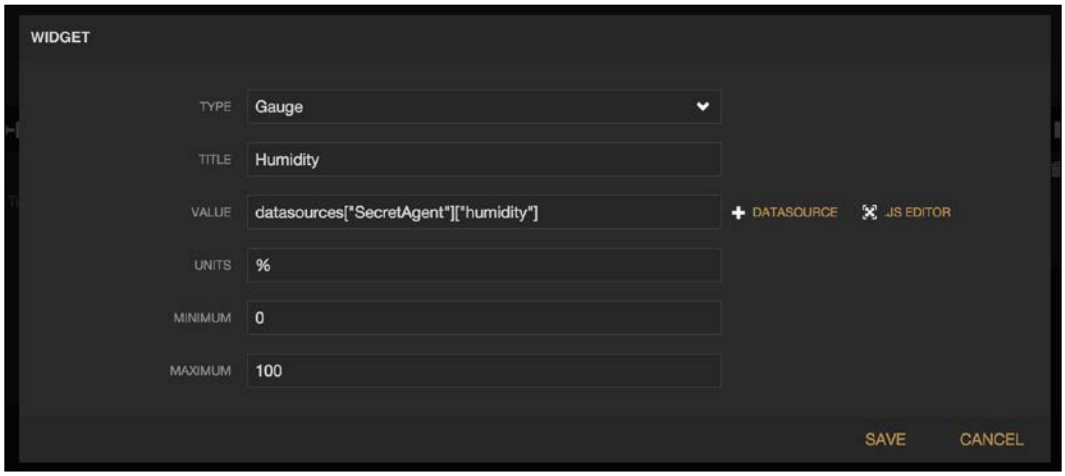
Затем щелкните маленький знак +, чтобы создать новый виджет. Здесь мы собираемся использовать виджет датчика температуры. Заполните форму создания виджета, как показано на следующем скриншоте:



Вы должны сразу увидеть указатель температуры на вашей плате:

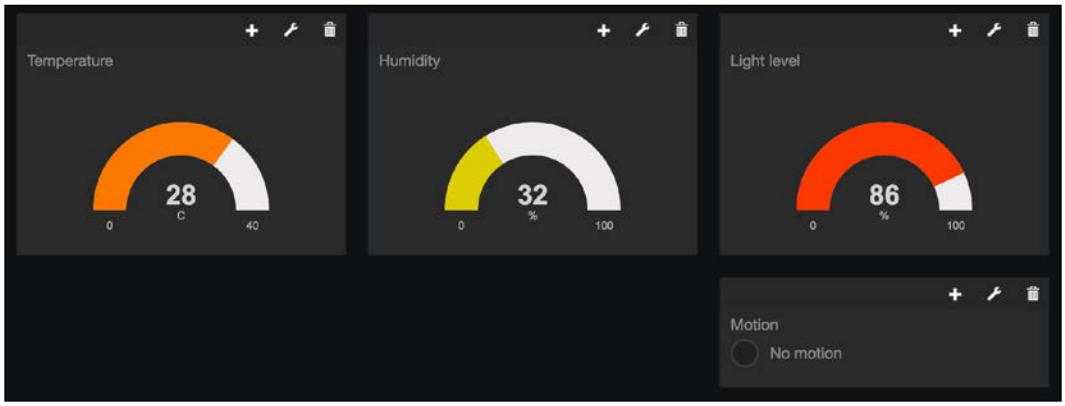


Теперь сделаем то же самое с влажностью:

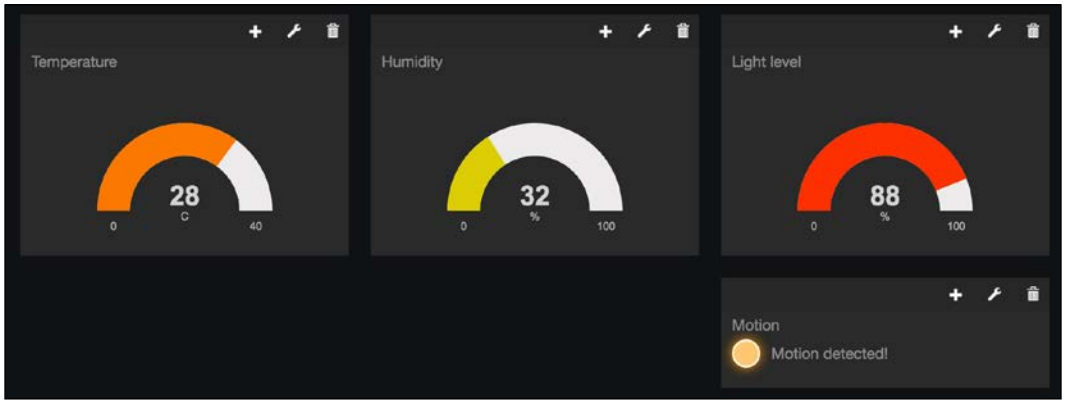


Вы также можете сделать то же самое для уровня внешней освещенности. Теперь у вас есть все данные с этих датчиков, обновляемые практически в реальном времени на вашей плате:

Последнее, что нам нужно поставить, это датчик движения. Поскольку это датчик включения / выключения, я использовал для этого виджет индикатора:



Теперь попробуйте провести рукой перед датчиком. Вы должны сразу увидеть, что индикатор меняет свой цвет:



Поздравляем, теперь у вас есть панель управления, к которой вы можете получить доступ в любое время, чтобы следить за этими данными!

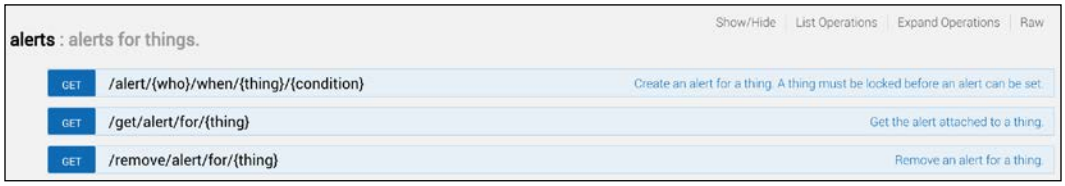
Создание автоматических оповещений по электронной почте

С нашим проектом мы можем сделать еще кое-что. Шпионить за данными - это хорошо, но мы не всегда за компьютером. Например, мы хотели бы получать уведомление по электронной почте, если проект обнаруживает движение.

Dweet.io предлагает эту услугу по очень низкой цене (менее 1 доллара в месяц). Для этого вам необходимо заблокировать свое устройство. По сути, это ключ, который можно получить на https://dweet.io/locks.

Это экран, на котором вы можете купить этот ключ:

Как только у нас будет ключ, мы сможем установить оповещение. Сайт dweet.io очень хорошо это объясняет:



Чтобы настроить оповещение, просто перейдите по следующему URL-адресу, изменив различные параметры желаемыми параметрами https://dweet.io/alert/youremail@yourdomain.com/when/yourThing/dweet.motion==1? Key = yourKey .
Как только это будет сделано, вы автоматически получите предупреждение всякий раз, когда ваш проект обнаружит движение, даже если вы на самом деле не просматриваете данные!

Резюме

В этой главе мы создали устройство, которое можно просто поместить в комнате, чтобы шпионить за чем либо из любой точки мира. Он непрерывно записывает данные с подключенных к нему датчиков и отправляет эти данные непосредственно в облако. Конечно, вы можете сделать многое, чтобы улучшить этот проект. Вы можете, например, поэкспериментировать со своими датчиками, в зависимости от данных, за которыми вы хотите следить. Вы также можете попробовать заменить чип CC3000 Wi-Fi на шилд GSM / GPRS, который мы использовали в предыдущей главе, чтобы иметь модуль, который вы можете где-то разместить, не подключая его к сети Wi-Fi.

В следующей главе мы углубимся в более сложную тему: собственно создание GPS-трекера на базе Arduino!

8

Создание GPS-трекера с Arduino

Теперь мы собираемся создать очень распространенный инструмент, который должен быть у любого секретного агента: GPS-трекер. Мы снова будем использовать платформу Arduino, чтобы создать собственный GPS-трекер. Фактически мы создадим два приложения, используя одно и тоже оборудование:

- Первым будет устройство определения местоположения, которое отправляет свою позицию через SMS.
- Другой проект - это GPS-трекер, который вы сможете найти в реальном времени на карте.

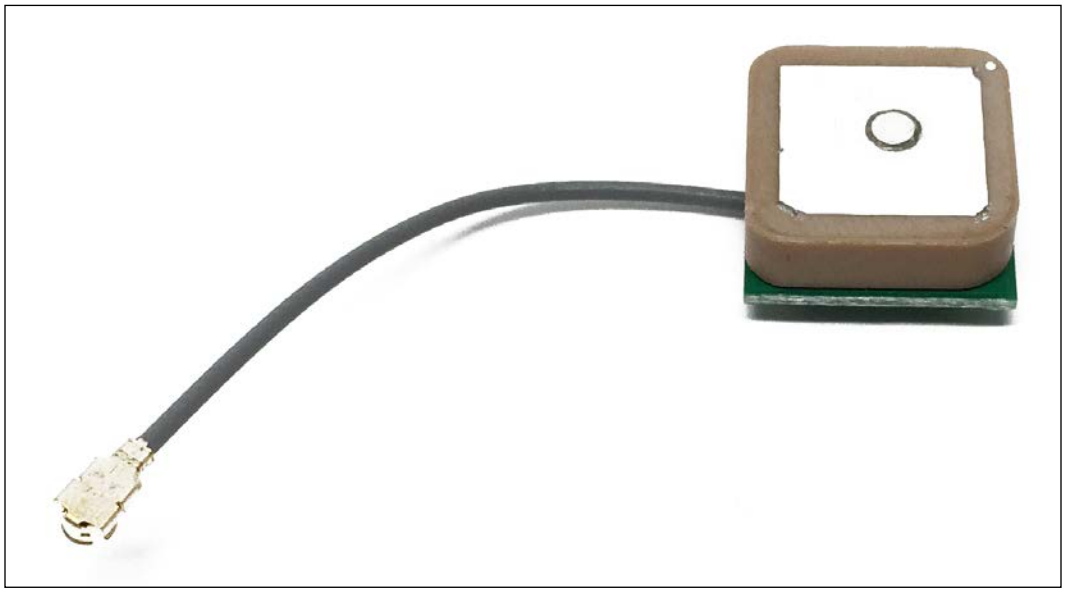
Начнем!

Компоненты и программы

Сначала давайте посмотрим, какие компоненты необходимы для этого проекта. Как правило, мы собираемся использовать плату Arduino Uno в качестве центральной части проекта.

Для частей GPRS и GPS мы собираемся снова использовать модуль Adafruit FONA 808, который мы уже использовали в главе 5 «Открытие замка с помощью SMS». Мы также используем антенну GSM для связи GSM / GPRS.

Однако, поскольку здесь мы хотим использовать встроенный GPS, нам такжепонадобится дополнительная антенна GPS. Я использовал стандартную пассивную GPS-антенну uFL отAdafruit:



Затем вам понадобится аккумулятор для питания шилда FONA и встроенного модуля GPS, поскольку плата Arduino Uno не позволяет запитать шилд FONA,который потребляет до 2 А за раз!. Для этого я использовал LiPo аккумулятор на 3,7 В вместе с зарядным устройством microUSB.

Очень важная часть проекта - это SIM-карта, которую нужно разместить внутри шилда FONA. Вам понадобится обычная SIM-карта (не micro или nano), которая активирована, не заблокирована PIN-кодом и способна получать текстовые сообщения. Вы можете получить его у любого из местных операторов мобильной связи. Кроме того, в этой главе вам также необходимо иметь активный тарифный план с объемом кредита не менее 1 МБ на счете.

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

- Arduino Uno
- Adafruit Fona 808 breakout
- GSM uFL antenna
- SIM-карта GSM
- 3.7V LiPo battery
- LiPo Зарядное устройство
- Passive GPS antenna

- Макетная плата
- Перемычки

Что касается программного обеспечения, вам понадобится только последняя версия Arduino IDE и библиотека Adafruit FONA. Вы можете установить эту библиотеку с помощью диспетчера библиотек Arduino IDE.

Конфигурация оборудования

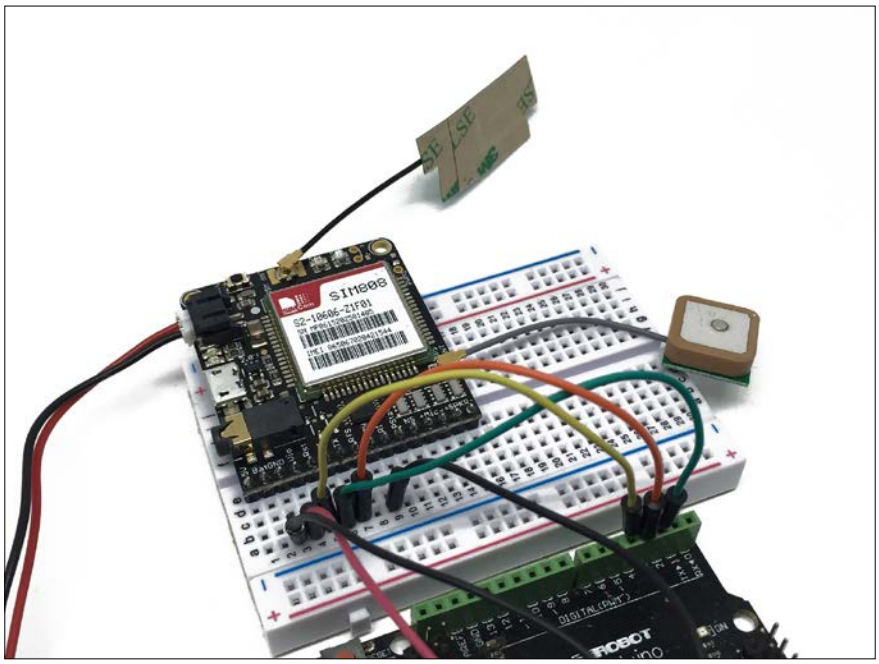
Пришло время собрать аппаратную часть этого проекта.

Сначала подключите источник питания к макетной плате: подключите контакт 5V от платы Arduino к красной линии питания на макетной плате, а контакт GND к синей линии питания.

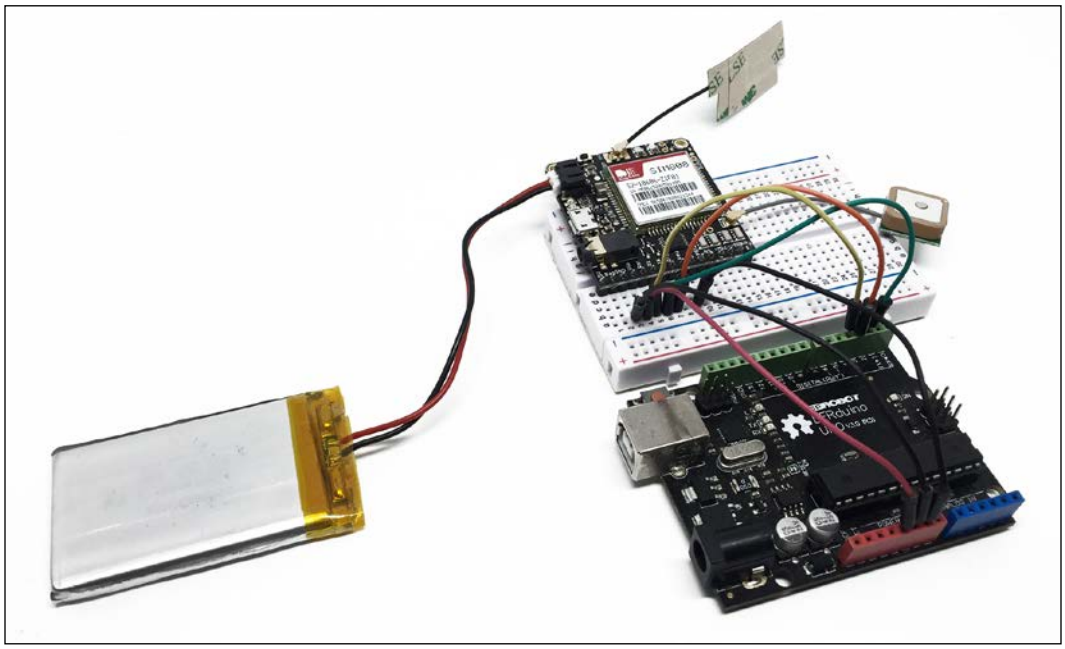
Затем поместите экран FONA на макетную плату. Подключите контакт VIO к красной линии питания, а контакты GND и Key - к синей линии питания.

После этого подключите контакт RST к контакту 4 Arduino, TX к контакту 3 Arduino, а RX к контакту Arduino 2. Также подключите LiPo аккумулятор 3,7 В, антенну GPS и антенну GSM к экрану FONA.

Это крупный план шилда после сборки проекта:



А это обзор всего проекта в сборе:



Тестирование функций локации

Прежде чем мы углубимся в два захватывающих проекта этой главы, мы сначала проведем простой тест с использованием FONA shield и посмотрим, действительно ли он может найти наш проект на карте. Скетч фактически проверит, правильно ли работает местоположение GPS.

Если нет, не беспокойтесь: скетч и другие проекты этой главы будут автоматически использовать местоположение GPRS. Он менее точен, но работает неплохо. Так будет, например, если вы тестируете этот проект внутри.

Это полный код для этой части:

```
// Библиотеки
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>

// Пины
#define FONA_RX 2
#define FONA_TX 3
```

```
#define FONA_RST 4

// Буфер
char replybuffer[255];

// определение параметров
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);

void setup() {

  // Плата инициализации
  while (!Serial);
  Serial.begin(115200);
  Serial.println(F("FONA location test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));

  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while(1);
  }
  Serial.println(F("FONA is OK"));

  // Распечатать номер IMEI SIM-карты.
  char imei[15] = {0}; // MUST use a 16 character buffer for IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("SIM card IMEI: "); Serial.println(imei);
  }

  // Настройка GPRS APN (имя пользователя / пароль по желанию)
  fona.setGPRSNetworkSettings(F("your_APN"));
  //fona.setGPRSNetworkSettings(F("your_APN"), F("your_username"),
F("your_password"));

  // Включить GPS
  if (!fona.enableGPS(true)) {
    Serial.println(F("Failed to turn on GPS"));
  }

  // Включить GPRS
```

```
  fona.enableGPRS(true);

  // Выберите локализацию GPS или GPRS
  boolean GPSloc;
  int8_t stat;

  // Проверить исправление GPS
  stat = fona.GPSstatus();
  if (stat < 0) {
    GPSloc = false;
  }
  if (stat == 0 || stat == 1) {
    GPSloc = false;
  }
  if (stat == 2 || stat == 3) {
    GPSloc = false;
  }

  // Распечатать, какой метод локализации используется
  Serial.print("Localisation method: ");
  if (GPSloc) {Serial.println("GPS");}
  else {Serial.println("GPRS");}

  // Позиция печати
  if (GPSloc) {
    char gpsdata[80];
    fona.getGPS(0, gpsdata, 80);
    Serial.println(F("Reply in format: mode, longitude, latitude,
altitude, utctime(yyyymmddHHMMSS), ttff, satellites, speed, course"));
    Serial.println(gpsdata);
  }
  else {
    uint16_t returncode;
    if (!fona.getGSMLoc(&returncode, replybuffer, 250))
      Serial.println(F("Failed!"));
    if (returncode == 0) {
      Serial.println(replybuffer);
    } else {
      Serial.print(F("Fail code #")); Serial.println(returncode);
    }
  }
}

void loop() {
  // Nothing here
}
```

Теперь мы рассмотрим важные части этого наброска.

Он начинается с импорта необходимых библиотек:

```
#include "Adafruit_FONA.h"
#include <SoftwareSerial.h>
```

Затем мы определяем пины, к которым подключен FONA:

```
#define FONA_RX 2
#define FONA_TX 3
#define FONA_RST 4
```

параметры объекта SoftwareSerial и FONA breakout:

```
SoftwareSerial fonaSS = SoftwareSerial(FONA_TX, FONA_RX);
SoftwareSerial *fonaSerial = &fonaSS;
Adafruit_FONA fona = Adafruit_FONA(FONA_RST);
```

Затем мы инициализируем FONA:

```
while (!Serial);
Serial.begin(115200);
Serial.println(F("FONA location test"));
Serial.println(F("Initializing....(May take 3 seconds)"));

fonaSerial->begin(4800);
if (! fona.begin(*fonaSerial)) {
  Serial.println(F("Couldn't find FONA"));
  while(1);
}
Serial.println(F("FONA is OK"));
```

После этого вам нужно разместить данные GPRS. Это полностью зависит от вашего оператора связи. Мне просто нужно было ввести APN, то есть Интернет, но, возможно, вам также придется указать имя пользователя и пароль. Свяжитесь с вашим оператором связи, чтобы получить точную информацию, затем закомментируйте / раскомментируйте нужную строку и заполните данные:

```
fona.setGPRSNetworkSettings(F("your_APN"));
//fona.setGPRSNetworkSettings(F("your_APN"), F("your_username"),
F("your_password"));
```

Затем мы можем включить GPS:

```
if (!fona.enableGPS(true)) {
  Serial.println(F("Failed to turn on GPS"));
}
```

Так же включаем соединение GPRS:

```
fona.enableGPRS(true);
```

Когда это будет сделано, мы получаем состояние GPS и проверяем, можем ли мы найти спутник GPS и получить фиксацию. Если да, мы будем использовать GPS для определения местоположения. Если нет, мы переключимся на местоположение GPRS:

```
stat = fona.GPSstatus();
if (stat < 0) {
  GPSloc = false;
}
if (stat == 0 || stat == 1) {
  GPSloc = false;
}
if (stat == 2 || stat == 3) {
  GPSloc = false;
}
```

Как только мы узнаем, какой метод определения местоположения использовать, мы можем фактически получить местоположение в зависимости от выбранного метода:

```
if (GPSloc) {
  location = getLocationGPS();
  latitude = getLatitudeGPS(location);
  longitude = getLongitudeGPS(location);
  latitudeNumeric = convertDegMinToDecDeg(latitude.toFloat());
  longitudeNumeric = convertDegMinToDecDeg(longitude.toFloat());
}
else {
  location = getLocationGPRS();
  latitude = getLatitudeGPRS(location);
  longitude = getLongitudeGPRS(location);
  latitudeNumeric = latitude.toFloat();
  longitudeNumeric = longitude.toFloat();
}
Serial.print("Latitude, longitude: ");
Serial.print(latitudeNumeric, 6);
Serial.print(",");
Serial.println(longitudeNumeric, 6);
```

Как видите, здесь мы вызываем несколько вспомогательных функций, чтобы получить широту и долготу в качестве плавающих переменных. Теперь мы рассмотрим все эти функции подробно.
Вот подробности функции определения местоположения с помощью GPS:

```
String getLocationGPS() {

  // Буфер
  char gpsdata[80];

  // Получить данные
  fona.getGPS(0, gpsdata, 80);
  return String(gpsdata);
}
```

Эта функция действительно проста, поскольку она просто получает данные от GPS FONA и возвращает их в виде `String`.

Функция получения местоположения с помощью GPRS аналогична:

```
String getLocationGPRS() {

  // Буфер для ответа и код возврата
  char replybuffer[255];
  uint16_t returncode;

  // Получить и вернуть местоположение
  if (!fona.getGSMLoc(&returncode, replybuffer, 250))
    return String("Failed!");
  if (returncode == 0) {
    return String(replybuffer);
  } else {
    return String(returncode);
  }

}
```

Затем нам действительно нужно извлечь данные из возвращенного объекта String и получить широту и долготу нашего модуля GPS. Вот функция долготы с использованием местоположения GPRS:

```
String getLongitudeGPRS(String data) {

  // Find commas
  int commaIndex = data.indexOf(',');
  int secondCommaIndex = data.indexOf(',', commaIndex+1);

  return data.substring(0, commaIndex);
}
```

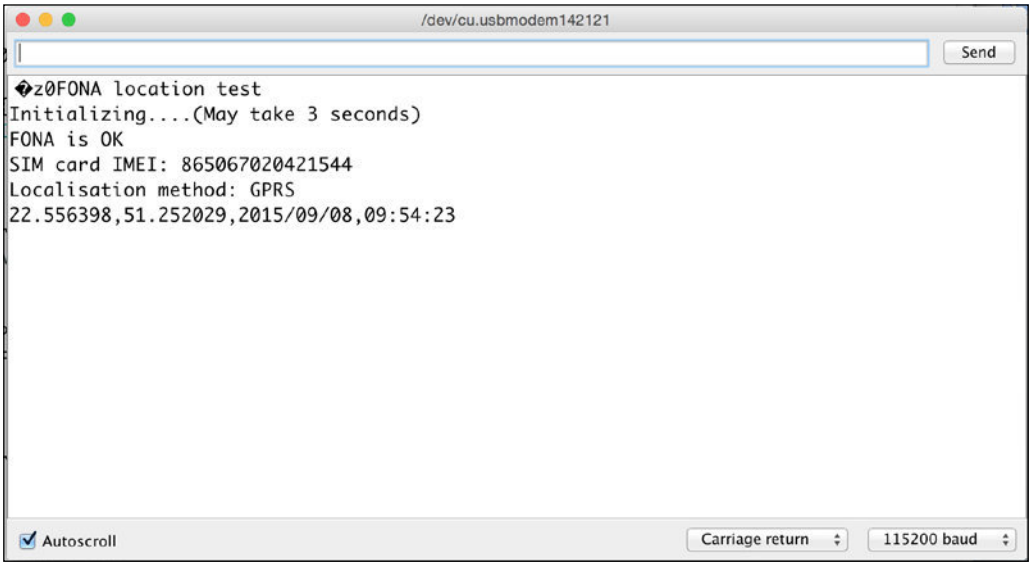
Это функция для получения широты с использованием местоположения GPRS:

```
String getLatitudeGPRS(String data) {

  // Find commas
  int commaIndex = data.indexOf(',');
  int secondCommaIndex = data.indexOf(',', commaIndex+1);

  return data.substring(commaIndex + 1, secondCommaIndex);
}
```

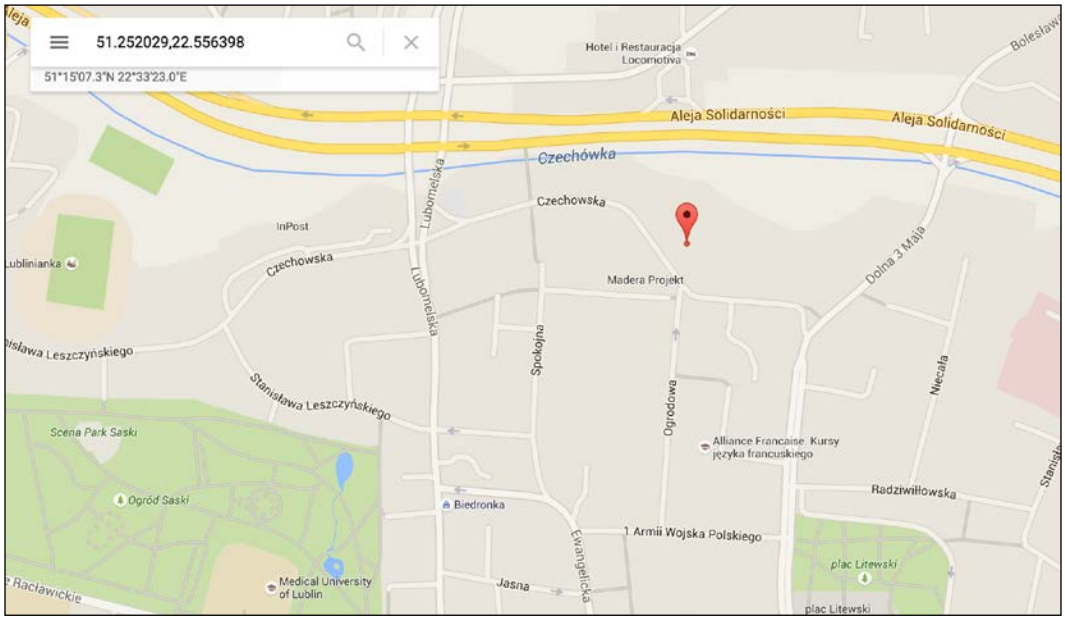
Для местоположения по GPS все аналогично. Однако нам нужна одна дополнительная функция для модуля GPS. Действительно, широта и долгота указываются в градусах-минутах-секундах, и нам нужно преобразовать их в числовой формат. Это делается с помощью следующей функции:

```
double convertDegMinToDecDeg (float degMin) {
  double min = 0.0;
  double decDeg = 0.0;

  // Получите минуты, fmod () требует удвоения
  min = fmod((double)degMin, 100.0);

  //восстановить координаты в десятичных
  градусахdegMin = (int) ( degMin / 100 );
  decDeg = degMin + ( min / 60 );

  return decDeg;
}
```

Пришло время проверить скетч! Просто поместите весь код в Arduino IDE, обновите настройки GPRS и загрузите его на плату. Затем откройте монитор последовательного порта, и вы должны увидеть следующее:

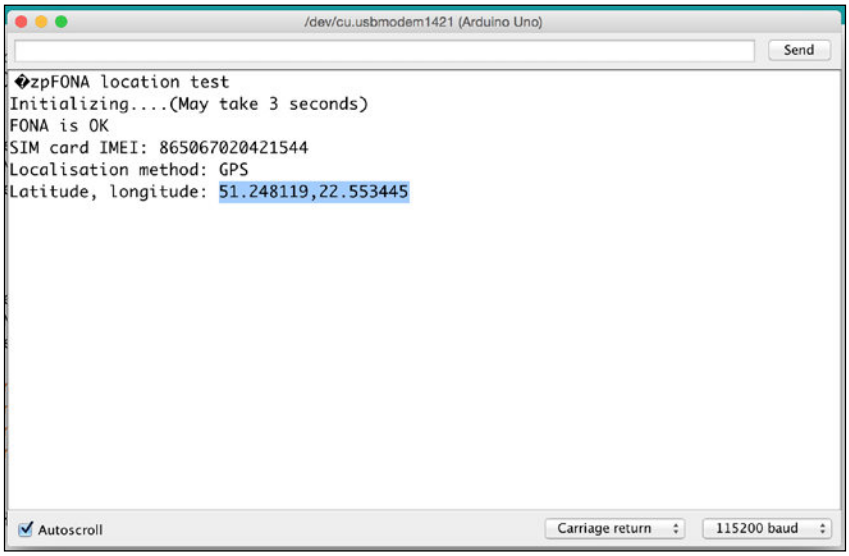


Как видите, я тестировал это в помещении, поэтому в скетче использовалось местоположение GPRS. Затем я просто скопировал широту и долготу и вставил их в Google Maps:

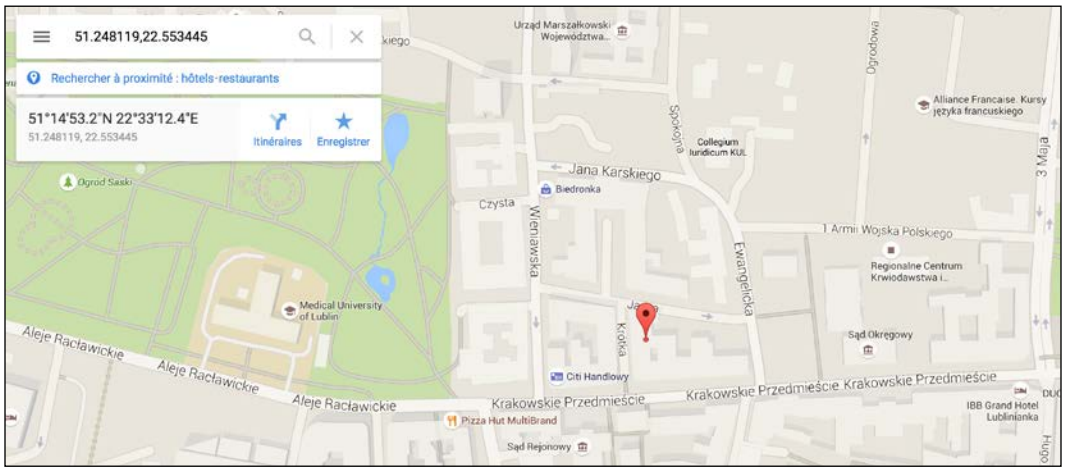


Я сразу увидел точку рядом с моим местоположением, которое находилось в 100-150 метрах от моего реального местоположения. Неплохо, если вам нужно отслеживать, например, находится что-то внутри или за пределами города.

Затем я снова попытался выйти на улицу, и между антенной GPS и небом ничего не было. Затем скетч автоматически выбрал опцию определения местоположения по GPS:



Затем я также скопировал и вставил широту и долготу GPS в Google Maps:



На этот раз местоположение было действительно точным и точно показывало, где я был в тот момент.

Отправка местоположения GPS по SMS

Теперь, когда у нас есть часть местоположения, работающая правильно, мы можем начать создавать наши проекты GPS-отслеживания для секретных агентов. Первый проект просто использует местоположение и отправляет его по SMS на выбранный вами номер телефона.

Поскольку скетч очень похож на предыдущий, я только покажу, какие части изменились по сравнению с скетчом проверки местоположения. Сначала нам нужно определить номер телефона, на который вы хотите отправить данные отслеживания:

```
char * sendToNumber = "123456789";
```

После получения местоположения, как и в предыдущем разделе, мы можем теперь создать сообщение, которое мы отправим через SMS:

```
char messageToSend[140];
String message = "Current GPS location: " + latitude + "," +
longitude;
message.toCharArray(messageToSend, message.length());
```

Используя пример FONA, очень легко отправить SMS на номер, который мы определили ранее:

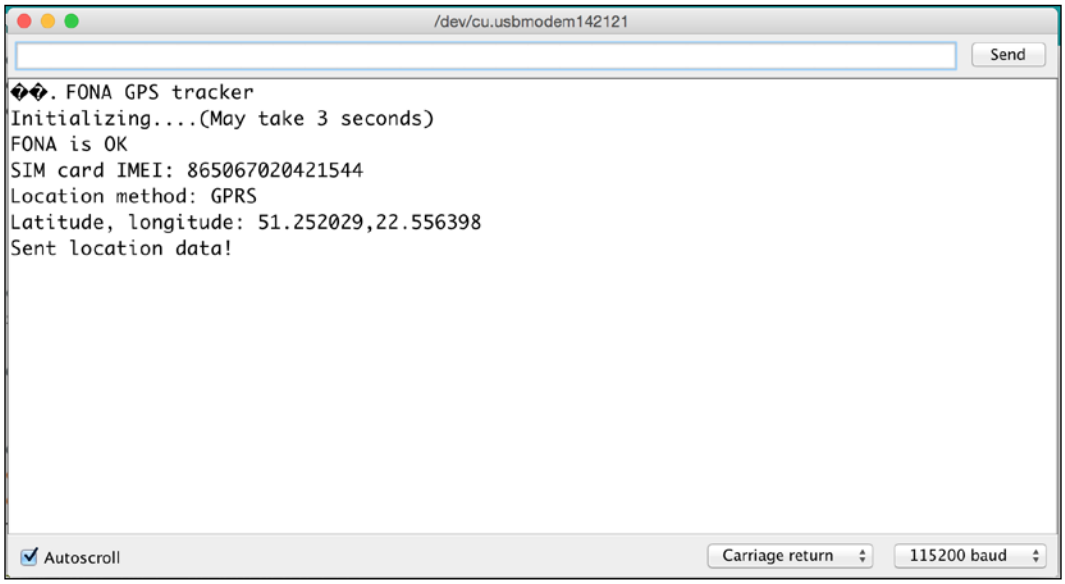
```
if (!fona.sendSMS(sendToNumber, messageToSend)) {
  Serial.println(F("Failed to send SMS"));
} else {
  Serial.println(F("Sent location data!"));
}
```

Наконец, мы ждем перед каждым сообщением. Я использовал 10 секунд для целей тестирования, но вы можете ввести свою задержку в миллисекундах:

```
Serial.println(F("Waiting until next message..."));
delay(10000);
```

Пришло время протестировать проект. Вы можете получить весь код из репозитория книги на GitHub https://github.com/marcoschwartz/arduino-secret-agents.

Затем убедитесь, что вы изменили номер телефона и настройки GPRS внутри кода и загрузили его на плату. Вот что вы должны увидеть:



Если вы видите последнюю строку, это означает, что SMS действительно отправлено. После этого я быстро получил сообщение:

Затем последовала серия сообщений с указанием текущего местоположения проекта:

Если вы видите это сообщение на своем телефоне, поздравляем, вы создали свой первый GPS-трекер с использованием Arduino!

Создание трекера местоположения GPS

Пришло время построить последний проект этой главы: настоящий GPS-трекер. Для этого проекта мы получим местоположение точно так же, как и раньше, используя GPS, если он доступен, и местоположение GPRS в противном случае.

Однако здесь мы собираемся использовать возможности GPRS шилда для отправки данных о широте и долготе на dweet.io, который мы использовали ранее. Затем мы внесем эти данные на Google Maps, чтобы вы могли следить за своим устройство в реальном времени из любой точки мира.

Поскольку скетч очень похож на те, что были в предыдущих разделах, я подробно расскажу только о самых его важных частях.

Вам нужно определить имя для объекта, который будет содержать данные о местоположении GPS:

```
String dweetThing = "mysecretgpstracker";
```

Затем, получив текущее местоположение, мы готовим данные для отправки на dweet.io:

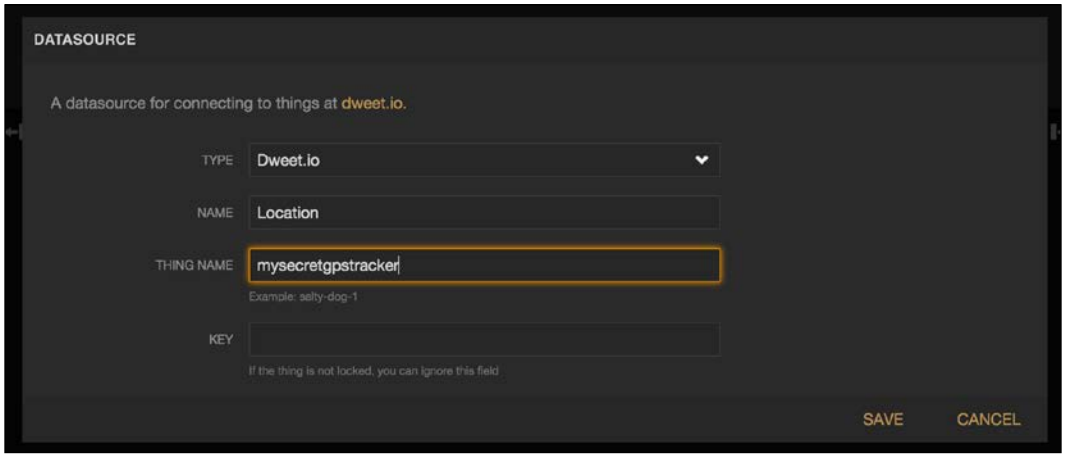
```
uint16_t statuscode;
int16_t length;
char url[80];
String request = "www.dweet.io/dweet/for/" + dweetThing + "?latitude="
+ latitude + "&longitude=" + longitude;
request.toCharArray(url, request.length());
```

После этого мы фактически отправляем данные на dweet.io:

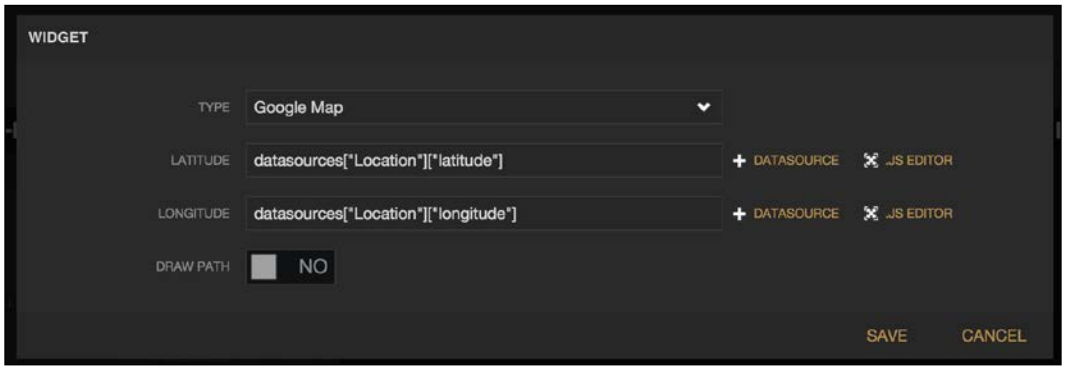
```
if (!fona.HTTP_GET_start(url, &statuscode, (uint16_t *)&length)) {
    Serial.println("Failed!");
  }
  while (length > 0) {
    while (fona.available()) {
      char c = fona.read();

      // Serial.write работает слишком медленно, мы будем писать
напрямую в Serialregister!
      #if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
        loop_until_bit_is_set(UCSR0A, UDRE0); /* Wait until data
register empty. */
        UDR0 = c;
      #else
        Serial.write(c);
      #endif
      length--;
    }
  }
  fona.HTTP_GET_end();
```

Теперь, перед тестированием проекта, мы собираемся подготовить нашу панель инструментов, на которой будет размещен виджет Google Maps. Как и в предыдущей главе, мы воспользуемся для этой цели freeboard.io. Если у вас еще нет учетной записи, перейдите на http://freeboard.io/.

Создайте новую панель мониторинга, а также новый источник данных. Вставьте название вашего устройства в поле THING NAME.:

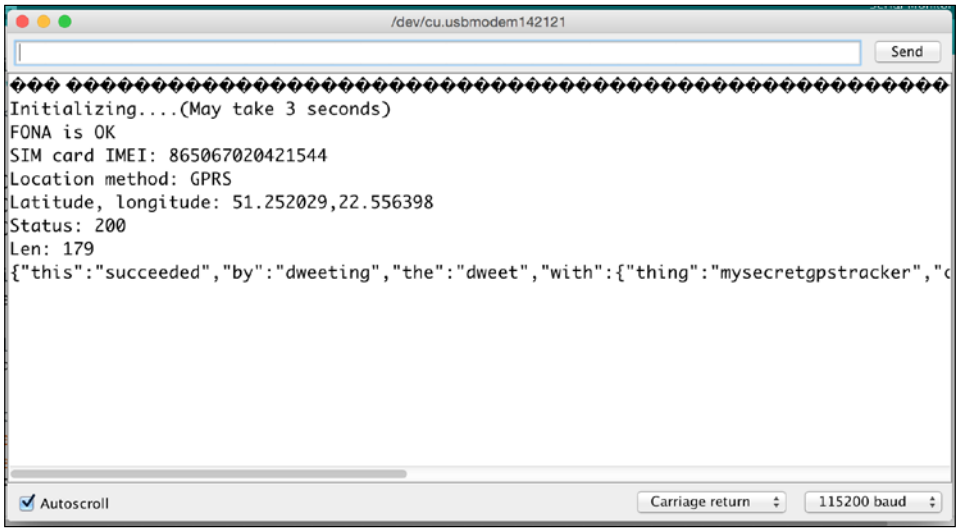


Затем создайте новую панель с виджетом Google Maps. Свяжите этот виджет с широтой и долготой источника данных о вашем местоположении:



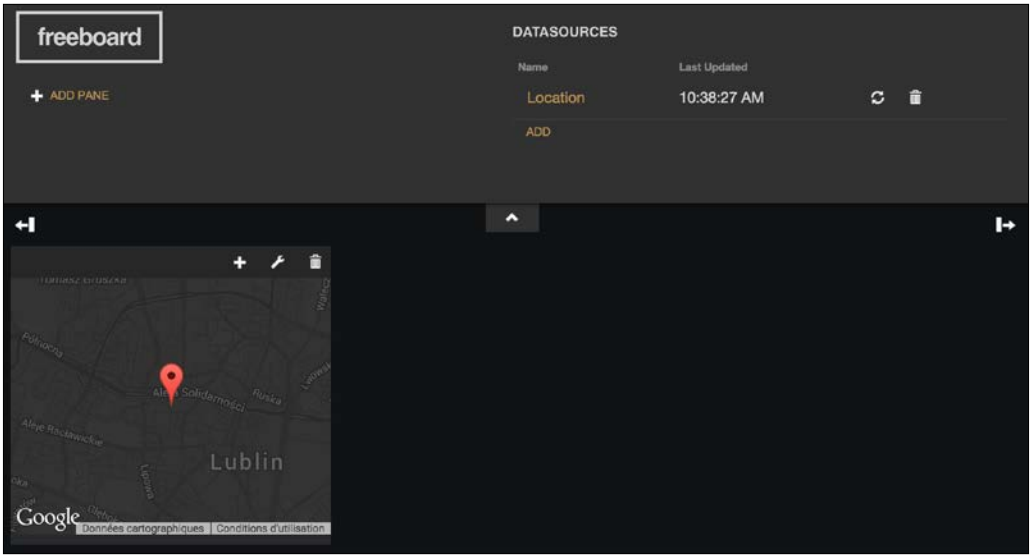
Пришло время протестировать проект. Убедитесь, что вы взяли весь код, например, из папки с кодами. Кроме того, не забудьте изменить название вещи, а также настройки GPRS.

Затем загрузите скетч на плату и откройте монитор последовательного порта. Вот что вы должны увидеть:



Самая важная строка - последняя, которая подтверждает, что данные были отправлены на dweet.io и были там сохранены.

Теперь просто вернитесь к только что созданной приборной панели, и теперь вы можете увидеть, что местоположение на карте было обновлено:



Обратите внимание, что эта карта также обновляется в реальном времени по мере поступления новых измерений с платы. Вы также можете изменить задержку между двумя обновлениями положения трекера, изменив функцию delay () в скетче. Поздравляем, вы только что создали собственное устройство GPS-слежения!

Резюме

В этой главе мы создали устройство, которое позволяет нам отслеживать GPS-местоположение любого объекта, к которому оно прикреплено. Мы создали два проекта на основе одного и того же оборудования: один, который отправляет на ваш телефон SMS-сообщения с указанием местоположения устройства, а другой - где вы можете увидеть текущее местоположение устройства на карте.

Конечно, есть много способов улучшить этот проект. Вы можете, например, добавить в проект датчики и отслеживать их также с панели инструментов Freeboard, как мы это делали в предыдущем проекте. Еще одна полезная вещь - сделать проект автономным, и для этого вы можете просто запитать плату Arduino напрямую от шилда FONA, так как он имеет выход 5 В

В заключительной главе книги мы собираемся создать еще одно классное приложение для секретных агентов: небольшого мобильного робота-наблюдателя.

9

Создание робота-шпиона Arduino

В этом последнем проекте книги мы собираемся построить небольшого робота-шпиона для наблюдения на базе Arduino. Секретный агент сможет удаленно управлять роботом с веб-страницы и видеть то, что видит робот, в режиме реального времени с помощью камеры.

Для этого мы будем использовать все, что вы узнали из этой книги:

- Как использовать Wi-Fi с Arduino
- Как создавать интерфейсы управления
- Как использовать камеру с Arduino Yun

Компоненты и программы

Сначала давайте посмотрим, какие компоненты необходимы для этого проекта. Ядром проекта, конечно же, будет сам робот. Для шасси робота на рынке имеется широкий выбор. Для этого проекта я выбрал шасси DFRobot MiniQ 2wheels, которое представляет собой небольшое шасси робота, на которое вы можете легко установить платы Arduino.

Затем для этого проекта вам понадобятся две платы Arduino. Первый будет Arduino Yun, который мы будем использовать для подключения USB-камеры, как мы это делали в главе 6 - Создание облачной шпионской камеры. Для самой камеры я снова использовал камеру C720 от Logitech.

Другая вещь, которая вам понадобится, - это Arduino Uno, которая позаботится о приводе двигателей робота через моторный шилд. Здесь нам нужно использовать Arduino Uno, потому что Arduino Yun несовместим с большинством моторных шилдов DFRobot.

Для управления моторами использовал моторный шилд на 1A от DFRobot. Для дистанционного управления роботом мы также будем использовать коммутационную плату CC3000, которую мы использовали в предыдущей главе. Чтобы собрать плату CC3000 с остальной частью робота, мы также будем использовать шилд прототипирования DFRobot.

Вам также понадобится аккумуляторная батарея 7,2 В для питания робота, когда он не подключен к вашему компьютеру. Я также использовал аккумуляторную батарею DFRobot с адаптером постоянного тока.

Наконец, вот список всех компонентов, которые мы будем использовать в этом проекте:

- DFRobot MiniQ 2-х колесное шасси
- DFRobot Моторный шилд
- DFRobot Прототип шилд
- CC3000 breakout плата
- Arduino Uno
- Arduino Yun
- USB камера
- Макетная плата
- Перемычки
- Аккумулятор 7,2 В с разъемом питания

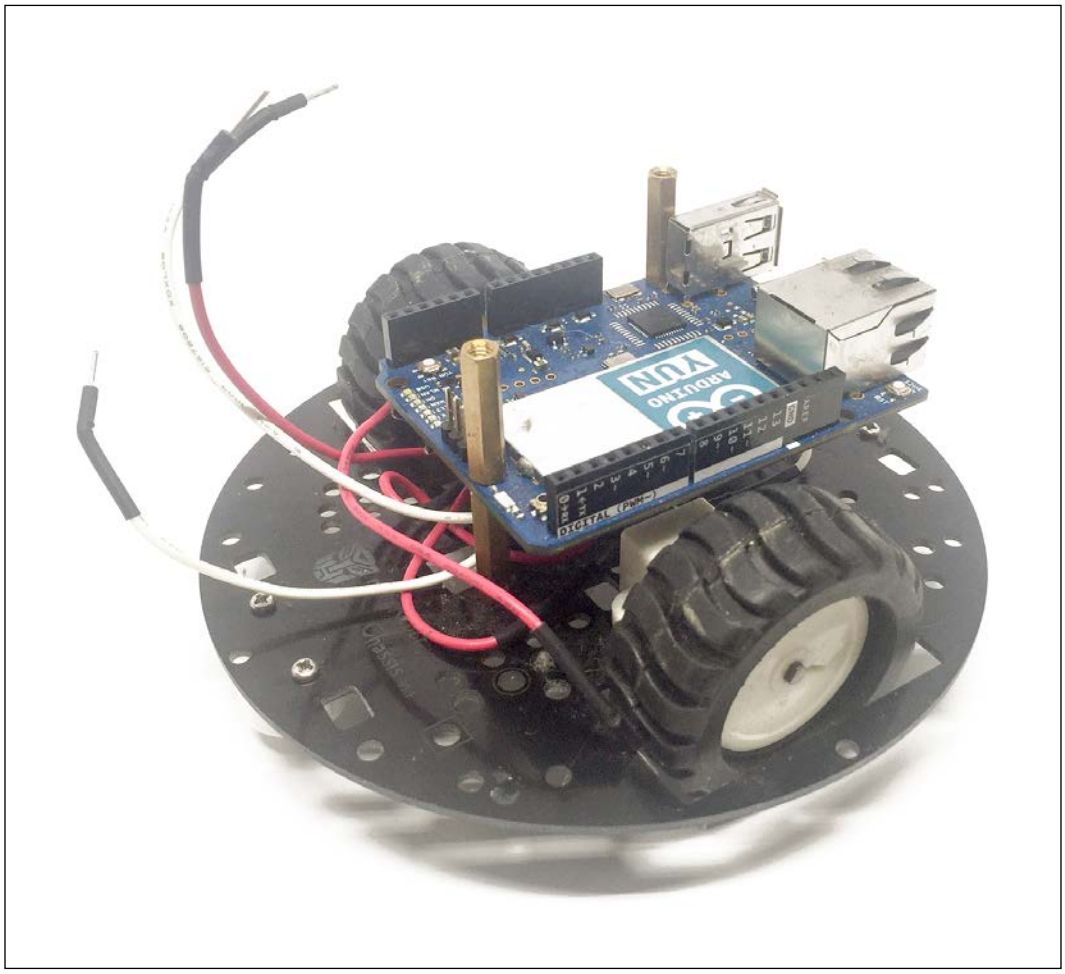
Что касается программного обеспечения, вам понадобится последняя версия Arduino IDE. Вам также потребуется установить следующие библиотеки с помощью диспетчера библиотек Arduino:

- Adafruit CC3000 library
- aREST

Конфигурация оборудования

Начнем с самой сложной части этого проекта: сборки самого робота. Конечно, шаги будут зависеть от шасси робота, которое вы используете, но цель состоит в том, чтобы дать вам основные шаги в этом разделе.

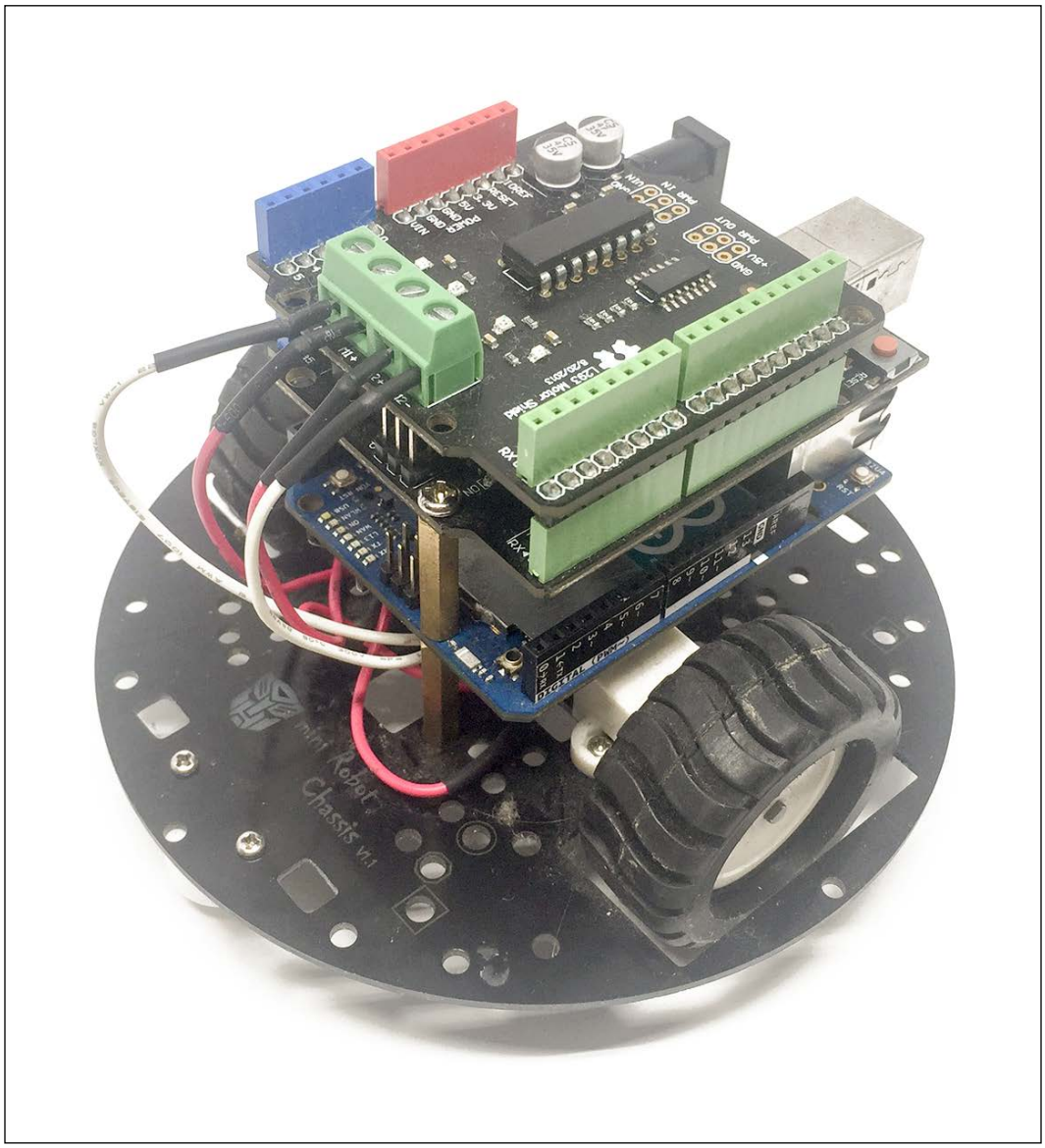
Начнем с фактического монтажа платы Arduino Yun на шасси робота с помощью винтов и разъемов, поставляемых с самим шасси:



Как только это будет сделано, мы монтируем плату Arduino Uno поверх Yun, используя латунные стойки.

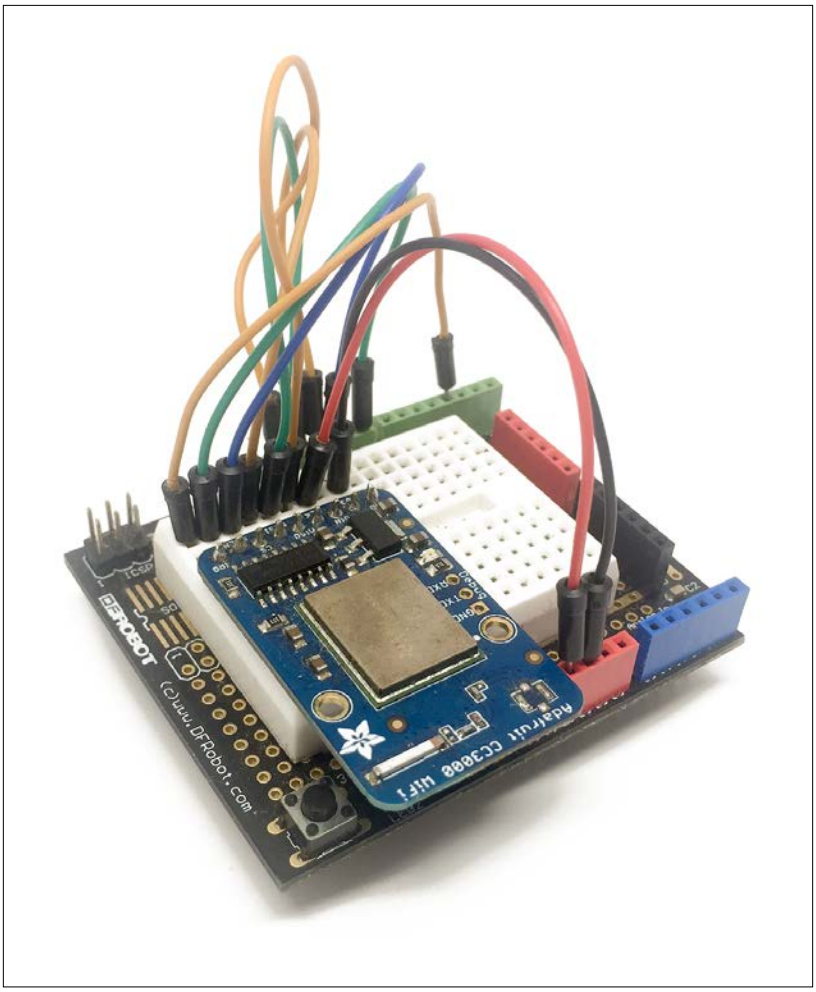
Затем мы устанавливаем моторный шилд поверх платы Arduino Uno. Затем мы вставляем кабели от двигателей в специальные разъемы на шилде двигателя и закрепляем их винтами. Убедитесь, что вы используете одинаковую полярность для обоих двигателей.

Вот результат на данный момент:

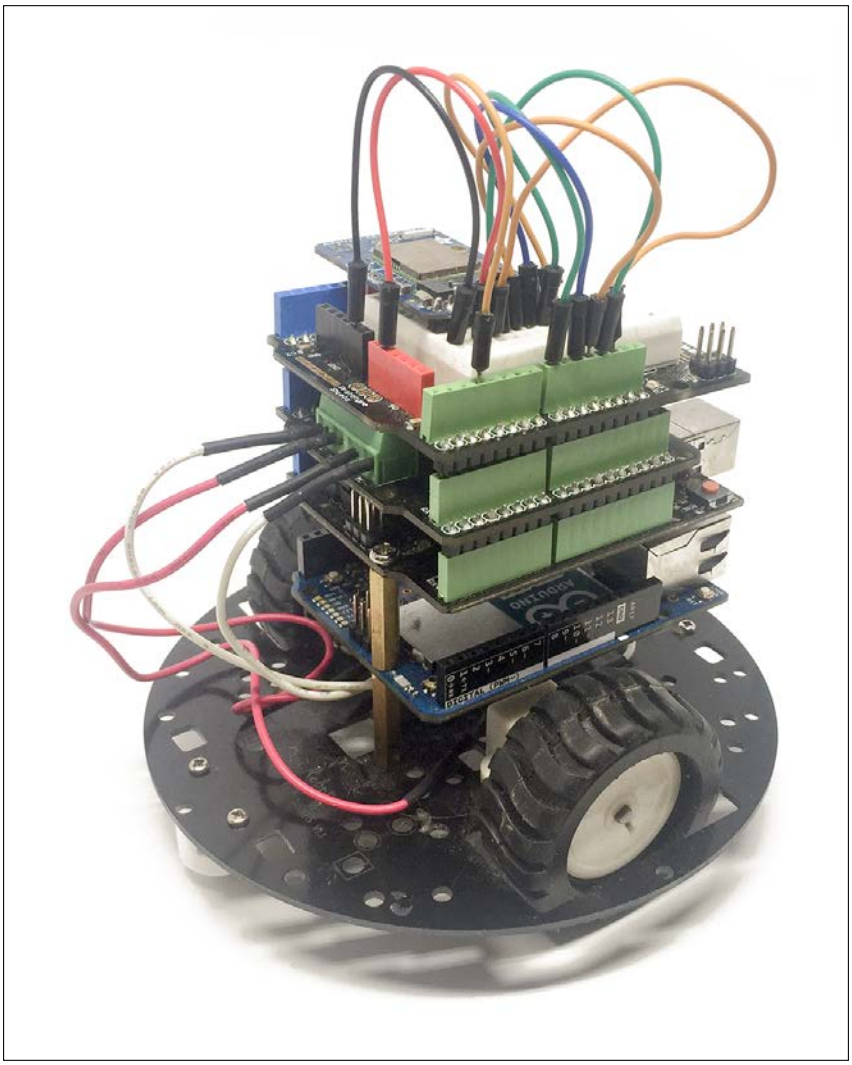


Теперь мы собираемся собрать коммутационную плату CC3000 на шилд прототипа, который затем будет установлен поверх робота. Чтобы узнать, как подключить плату CC3000board, обратитесь к главе 7 - Мониторинг секретных данных из любого места. Единственная разница в том, что вывод IRQ подключен к выводу 3, а вывод VBAT подключен к выводу 8.

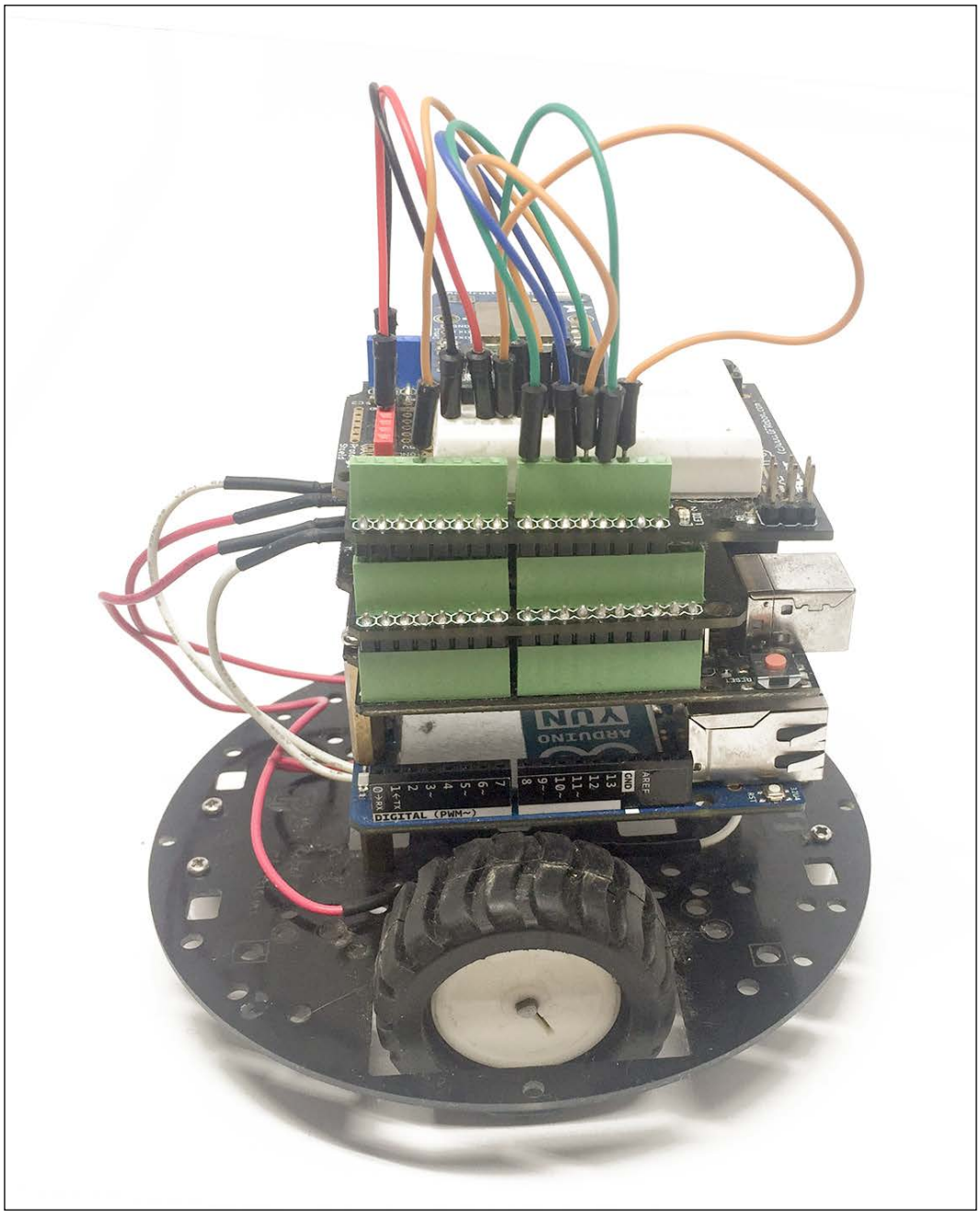
Это собранная плата CC3000 с макетной платой для прототипирования::



Теперь просто установите этот шилд на робота.:

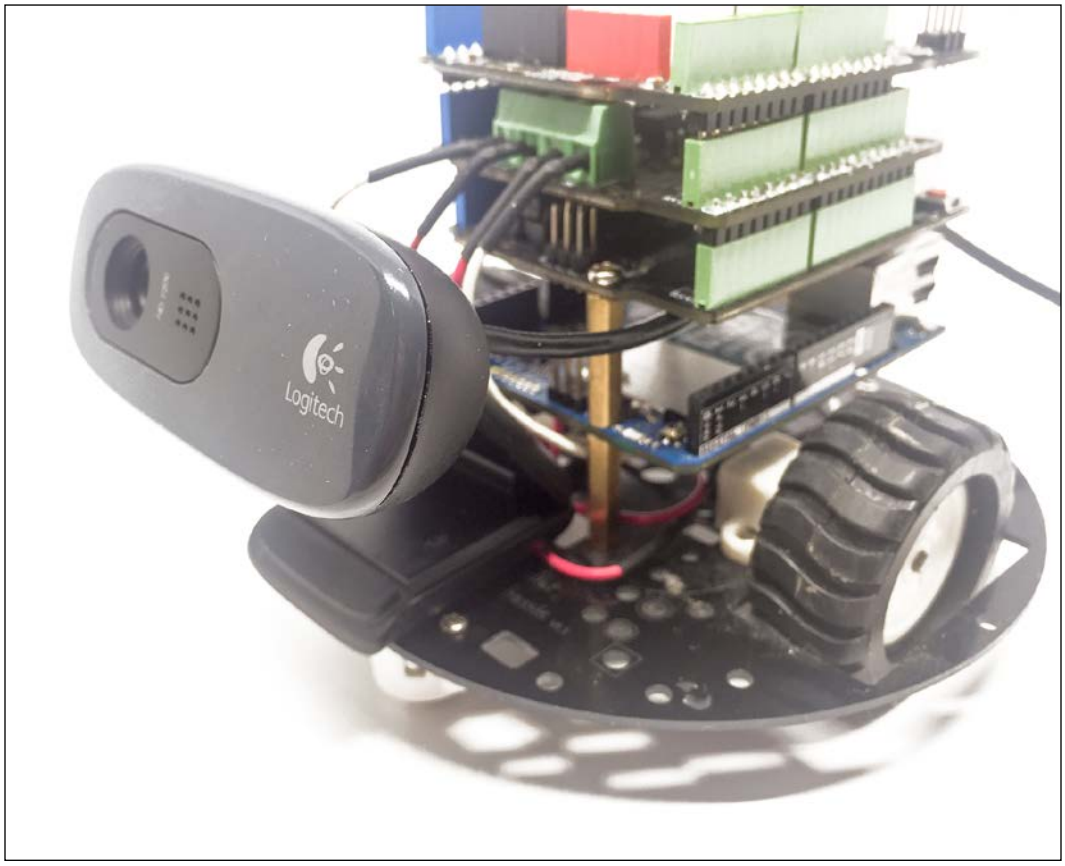


Вот вид сбоку на проект на этом этапе:

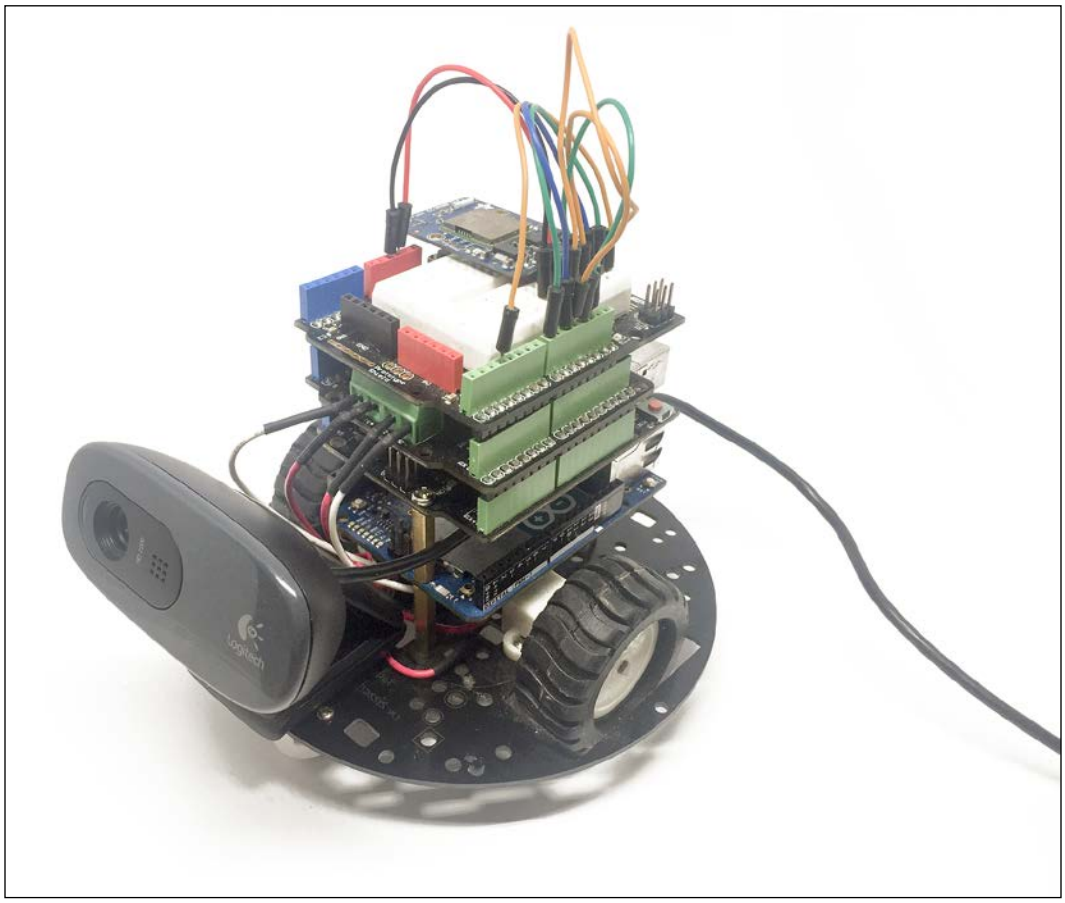


Наконец, подключите камеру к USB-порту Arduino Yun и поместите ее перед роботом. Я закрепил его винтами, но это будет зависеть от шасси вашего робота.

Это увеличенное изображение полностью собранного робота:



Это обзор полностью собранного робота:



Если у вас аналогичный результат, поздравляем! Теперь вы готовы запрограммировать своего робота-наблюдателя. Пока не беспокойтесь о батарее, так как мы будем делать все с помощью USB-кабелей, которые подключат робота напрямую к вашему компьютеру.

Настройка управления двигателем

Теперь мы собираемся настроить различные части проекта, начиная с настройки Arduino Uno, которая будет управлять двигателями через Wi-Fi. Для этого мы воспользуемся библиотекой aREST Arduino, которая упрощает управление проектами Arduino через Wi-Fi.

Вот полный код этой части:

```
// Тест робота через aREST + WiFi
#define NUMBER_VARIABLES 1
#define NUMBER_FUNCTIONS 5

// Библиотеки
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include <aREST.h>
#include <avr/wdt.h>

// CC3000 пины
#define ADAFRUIT_CC3000_IRQ   3
#define ADAFRUIT_CC3000_VBAT  8
#define ADAFRUIT_CC3000_CS    10

// Robot speed
#define FULL_SPEED 100
#define TURN_SPEED 50

// Motor пины
int speed_motor1 = 6;
int speed_motor2 = 5;
int direction_motor1 = 7;
int direction_motor2 = 4;

// Пины датчика
int distance_sensor = A0;

//  Задание параметров CC3000
Adafruit_CC3000 cc3000 = Adafruit_CC3000(ADAFRUIT_CC3000_CS, ADAFRUIT_
CC3000_IRQ, ADAFRUIT_CC3000_VBAT);

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

// Порт для прослушивания входящих TCP-соединений
#define LISTEN_PORT           80

// Server instance
Adafruit_CC3000_Server restServer(LISTEN_PORT);

#define WLAN_SSID       "KrakowskiePrzedm51m.15(flat15)"
```

```
#define WLAN_PASS       "KrK51flat15_1944_15"
#define WLAN_SECURITY   WLAN_SEC_WPA2

void setup(void)
{
  // Start Serial
  Serial.begin(115200);

  // Дайте имя роботу
  rest.set_id("1");
  rest.set_name("robot");

  // Выставить функции
  rest.function("forward",forward);
  rest.function("backward",backward);
  rest.function("left",left);
  rest.function("right",right);
  rest.function("stop",stop);

  // Настройте CC3000 и подключитесь к беспроводной сети
  Serial.print(F("Initialising CC3000..."));
  if (!cc3000.begin())
  {
    while(1);
  }
  Serial.println(F("done"));

  Serial.print(F("Connecting to WiFi..."));
  if (!cc3000.connectToAP(WLAN_SSID, WLAN_PASS, WLAN_SECURITY)) {
    while(1);
  }
  Serial.println(F("done"));

  Serial.print(F("Getting DHCP..."));
  while (!cc3000.checkDHCP())
  {
    delay(100);
  }

  // Запустить сервер
  restServer.begin();
```

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

  displayConnectionDetails();

  wdt_enable(WDTO_8S);
}

void loop() {

  // Обработка вызовов REST
  Adafruit_CC3000_ClientRef client = restServer.available();
  rest.handle(client);
  wdt_reset();

  // Проверьте подключение
  if(!cc3000.checkConnected()){while(1){}}
  wdt_reset();
}

// Вперед
int forward(String command) {

  send_motor_command(speed_motor1,direction_motor1,100,1);
  send_motor_command(speed_motor2,direction_motor2,100,1);
  return 1;
}

// Назад
int backward(String command) {

  send_motor_command(speed_motor1,direction_motor1,100,0);
  send_motor_command(speed_motor2,direction_motor2,100,0);
  return 1;
}

// Влево
int left(String command) {

  send_motor_command(speed_motor1,direction_motor1,75,0);
  send_motor_command(speed_motor2,direction_motor2,75,1);
  return 1;
}

// Вправо
```

```
int right(String command) {

  send_motor_command(speed_motor1,direction_motor1,75,1);
  send_motor_command(speed_motor2,direction_motor2,75,0);
  return 1;
}

// Stop
int stop(String command) {

  send_motor_command(speed_motor1,direction_motor1,0,1);
  send_motor_command(speed_motor2,direction_motor2,0,1);
  return 1;
}

// Функция для управления данным двигателем робота
void send_motor_command(int speed_pin, int direction_pin, int pwm,
boolean dir)
{
  analogWrite(speed_pin,pwm); // Set PWM control, 0 for stop, and 255
for maximum speed
  digitalWrite(direction_pin,dir);
}

// Распечатать сведения о подключении микросхемы CC3000
bool displayConnectionDetails(void)
{
  uint32_t ipAddress, netmask, gateway, dhcpserv, dnsserv;

  if(!cc3000.getIPAddress(&ipAddress, &netmask, &gateway, &dhcpserv,
&dnsserv))
  {
    Serial.println(F("Unable to retrieve the IP Address!\r\n"));
    return false;
  }
  else
  {
    Serial.print(F("\nIP Addr: ")); cc3000.printIPdotsRev(ipAddress);
    Serial.print(F("\nNetmask: ")); cc3000.printIPdotsRev(netmask);
    Serial.print(F("\nGateway: ")); cc3000.printIPdotsRev(gateway);
    Serial.print(F("\nDHCPsrv: ")); cc3000.printIPdotsRev(dhcpserv);
    Serial.print(F("\nDNSserv: ")); cc3000.printIPdotsRev(dnsserv);
    Serial.println();
    return true;
  }
}
```

Поскольку этот код довольно длинный, мы рассмотрим здесь только самые важные части и начнем с включения всех необходимых библиотек:

```
#include <Adafruit_CC3000.h>
#include <SPI.h>
#include <aREST.h>
#include <avr/wdt.h>
```

Затем мы определяем пины, соответствующие моторному шилду:

```
int speed_motor1 = 6;
int speed_motor2 = 5;
int direction_motor1 = 7;
int direction_motor2 = 4;
```

После этого вам необходимо ввести собственное имя сети Wi-Fi и пароль:

```
#define WLAN_SSID       "your_wifi_ssid"
#define WLAN_PASS       "your_wifi_password"
#define WLAN_SECURITY   WLAN_SEC_WPA2s
```

Затем мы объявляем процедуру aREST, который мы будем использовать позже для доступа к моторным функциям через Wi-Fi:

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

В функции скетча setup () мы предоставляем различные функции для управления роботом, чтобы они были доступны через Wi-Fi:

```
rest.function("forward",forward);
rest.function("backward",backward);
rest.function("left",left);
rest.function("right",right);
rest.function("stop",stop);
```

После этого запускаем на плате Wi-Fi сервер:

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

В функции скетча loop () мы постоянно отслеживаем соединения и обрабатываем их с помощью экземпляра REST:

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

Теперь давайте посмотрим на одну из функций управления роботом, например, на ту, которая движет робота вперед:

```
int forward(String command) {

  send_motor_command(speed_motor1,direction_motor1,100,1);
  send_motor_command(speed_motor2,direction_motor2,100,1);
  return 1;
}
```

В функции скетча loop () мы постоянно отслеживаем соединения и обрабатываем их с помощью процедуры REST:

```
void send_motor_command(int speed_pin, int direction_pin, int pwm,
boolean dir)
{
  analogWrite(speed_pin,pwm);
  digitalWrite(direction_pin,dir);
}
```

Пришло время настроить эту часть проекта. Подключите Arduino Uno к компьютеру через USB и загрузите этот скетч на плату. Затем откройте последовательный монитор. Через некоторое время вы должны увидеть IP-адрес платы, напечатанный на Serial monitor.

Настройка прямой трансляции

Теперь мы можем перейти к следующей части: настройке потокового видео с камеры на Arduino Yun. Это то, что мы уже делали в главе 6 «Создание облачной шпионской камеры», поэтому здесь мы рассмотрим только самые важные части. Обратитесь к разделу «Настройка оборудования», если вам нужно узнать, как снова настроить Yun.

Сначала подключитесь к вашему Yun, используя следующую команду:

```
ssh root@arduinoyun.local
```

Проверить, работает ли потоковая передача, можно на следующей странице.

http: //arduinoyun.local: 8080.

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 640x480 -f 25" -o
"output_http.so -p 8080 -w /www/webcam" &
```

Проверить, работает ли потоковая передача, можно на следующей странице.
http: //arduinoyun.local: 8080.

Настройка интерфейса

Теперь мы можем перейти к последней части: настройке интерфейса, который позволит секретному агенту управлять роботом, а также видеть прямую трансляцию с камеры. Этот интерфейс будет состоять из HTML-страницы и файла JavaScript. Он будет основан на модуле aREST .js, который упрощает управление устройствами aREST с веб-страницы.

Это полная HTML-страница:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset=utf-8 />
  <title>Surveillance Robot</title>
  <link rel="stylesheet" type="text/css" href="https://maxcdn.
bootstrapcdn.com/bootstrap/3.3.4/css/bootstrap.min.css">
  <link rel="stylesheet" type="text/css" href="interface.css">
  <script type="text/javascript" src="https://code.jquery.com/jquery-
2.1.4.min.js"></script>
  <script type="text/javascript" src="https://cdn.rawgit.com/Foliotek/
AjaxQ/master/ajaxq.js"></script>
  <script type="text/javascript" src="https://cdn.rawgit.com/
marcoschwartz/aREST.js/master/aREST.js"></script>
  <script type="text/javascript" src="interface.js"></script>
</head>

<body>

<div class='container'>

<h1>Surveillance Robot</h1>


<div class='row'>

  <div class="col-md-2"></div>

  <div class="col-md-2">
    <button id='fw' class='btn btn-primary btn-block'
type="button">Forward</button>
  </div>

</div>

<div class='row'>

  <div class="col-md-2">
```

```
    <button id='left' class='btn btn-primary btn-block'
type="button">Left</button>
  </div>

  <div class="col-md-2">
    <button id='stop' class='btn btn-danger btn-block'
type="button">Stop</button>
  </div>

  <div class="col-md-2">
    <button id='right' class='btn btn-primary btn-block'
type="button">Right</button>
  </div>

</div>

<div class='row'>

  <div class="col-md-2"></div>

  <div class="col-md-2">
    <button id='bw' class='btn btn-primary btn-block'
type="button">Backward</button>
  </div>

</div>

<div class='row'>

  <img src="http://arduinoyun.local:8080/?action=stream" />

</div>

</div>

</body>
</html>
```

Наиболее важные части на этой странице - это кнопки для управления роботом и прямой эфир с камеры. Например, это определяет кнопку для перемещения робота вперед:

```
<div class="col-md-2">
    <button id='fw' class='btn btn-primary btn-block'
type="button">Forward</button>
  </div>
```

Этот тег <img> позволяет вставлять на страницу прямой эфир с камеры:

```
<img src="http://arduinoyun.local:8080/?action=stream" />
```

Теперь давайте посмотрим на файл JavaScript, который фактически отправит команду роботу.
Это полный файл:

```
$( document ).ready(function() {

    // Device
    var address = '192.168.1.105';
    var device = new Device(address);

    // Button
    $('#fw').click(function() {
      device.callFunction('forward', '');
    });

    $('#bw').click(function() {
      device.callFunction('backward', '');
    });

    $('#left').click(function() {
      device.callFunction('left', '');
    });

    $('#right').click(function() {
      device.callFunction('right', '');
    });

    $('#stop').click(function() {
      device.callFunction('stop', '');
    });

});
```

Вам нужно изменить только одно в этом скрипте: IP-адрес вашей платы, который вы получили ранее:

```
  var address = '192.168.1.105';
  var device = new Device(address);
```

Каждая из различных строк кода на странице управляет функцией робота. Например, этот фрагмент кода создает связь между кнопкой вперед и функцией вперед на роботе:

```
$('#fw').click(function() {
     device.callFunction('forward', '');
});
```

Тестирование робота-наблюдателя

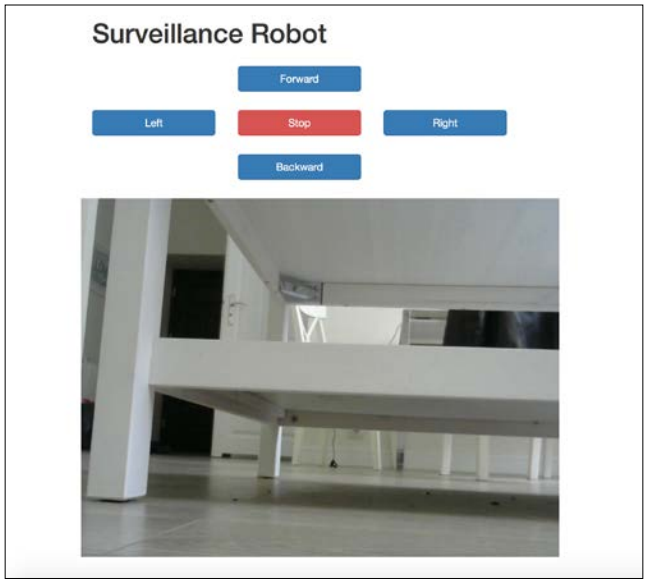
Теперь, когда у нас есть все элементы, пришло время протестировать нашего маленького робота-наблюдателя. Обратите внимание, что вы можете получить полный код с https://github.com/marcoschwartz/arduino-secret-agents.

Убедитесь, что вы настроили код и оборудование в соответствии с инструкциями из предыдущих разделов.

Теперь пора запитать робота от аккумулятора. Вставьте аккумулятор в корпус робота, а затем подключите его к входу разъема постоянного тока Arduino Uno.

Затем подключите вывод Arduino Uno 5V к выводу Vin на Yun. Также соедините контакты GND двух плат вместе. Это гарантирует, что Arduino Yun также будет запитан. Не забудьте после этого снова запустить прямой видеопоток.

Теперь поместите своего робота на землю и откройте страницу HTML. Вот что вы должны увидеть:



Теперь попробуйте разные кнопки, и вы увидите, что робот реагирует почти мгновенно. Поздравляем, вы только что создали своего собственного робота-шпиона на базе Arduino!

Резюме

В этой последней главе книги вы создали своего собственного маленького робота-наблюдателя, управляемого через Wi-Fi. С помощью этого робота вы теперь можете следить за тем, что происходит в комнате, а также перемещать робота.

Есть несколько способов улучшить этот проект. Например, вы можете добавить больше датчиков к роботу и отображать данные этих датчиков на той же странице, с которой вы управляете роботом.

Вы уже дошли до конца этой книги, и я надеюсь, что вам понравилось читать все проекты секретных агентов! Я настоятельно рекомендую вам снова попробовать все проекты из этой книги и создать их самостоятельно. Я также советую вам использовать все знания из этой книги и создавать свои собственные проекты секретных агентов.