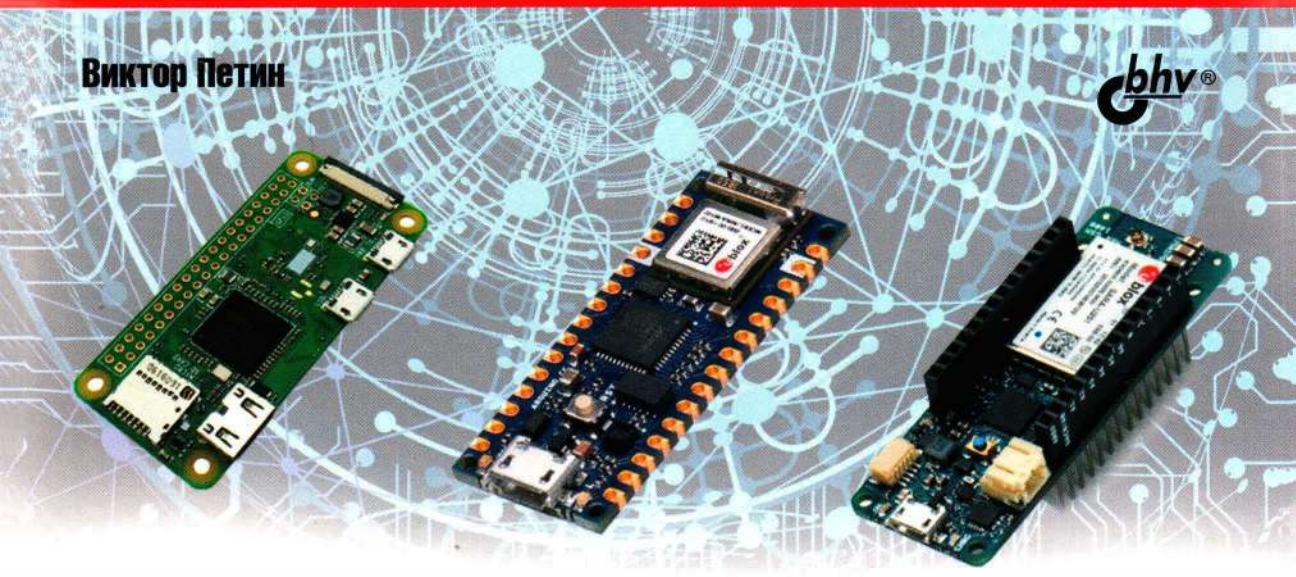
Электроника

Виктор Петин

bhv®



Новые возможности Arduino, ESP, Raspberry Pi

в проектах IoT

- исходные коды скетчей Arduino и ESP из книги
- исходные коды библиотек Arduino
- исходные коды python-скриптов для проектов с Raspberry Pi

Новые возможности, новые проекты.
Не опоздай!

Виктор Петин

Новые возможности Arduino, ESP, Raspberry Pi

в проектах IoT

Санкт-Петербург
«БХВ-Петербург»
2022

УДК 004.4
ББК 32.973.26-018.2
П29

Петин В. А.

П29 Новые возможности Arduino, ESP, Raspberry Pi в проектах IoT. — СПб.: БХВ-Петербург, 2022. — 320 с.: ил. — (Электроника)

ISBN 978-5-9775-6755-8

Рассмотрено создание простых устройств в рамках концепции Интернета вещей (IoT) на базе традиционных (Arduino Uno) и новых плат Arduino (MKR, Nano 33), плат ESP и микрокомпьютера Raspberry Pi. Приведены примеры подключения плат с помощью технологий Ethernet, WiFi, GPRS, BLE, LoRa к сети Интернет и другим устройствам. Описаны протоколы HTTP и MQTT. Рассмотрен обмен данными с облачными платформами Arduino IoT Cloud, Narodmon, ThingSpeak, Blynk и открытой LoRaWAN-сетью The Things Network (TTN).

Большая часть книги посвящена созданию практических проектов: собственный MQTT-сервер, табло на матрице для отображения биржевых котировок в реальном времени, GPS-трекер и онлайн-сервис поиска стоянок с использованием Яндекс.Карт, сканер штрих-кода с отправкой результатов в облако, IoT-принтер для печати курсов валют, бесконтактный измеритель температуры с отправкой данных в облако, предсказатель погоды на основе данных, поступающих в сервис ThingSpeak, проекты с элементами машинного обучения на платформе TinyML и др.

На сайте издательства размещен архив с исходными кодами программ и библиотек.

Для интересующихся современной электроникой

УДК 004.4
ББК 32.973.26-018.2

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Дизайн обложки	*Карины Соловьевой*

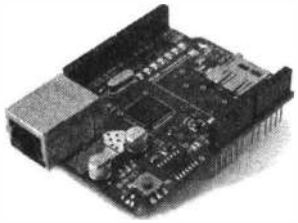
Подписано в печать 06.07.21.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 25,8.
Тираж 1200 экз. Заказ № 1725.
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готового оригинал-макета
ООО "Принт-М", 142300, М.О., г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-5-9775-6755-8

Оглавление

ГЛАВА 1



Вместо введения: что такое Интернет вещей?

Интернет вещей — это следующий уровень развития устройств, которые могут объединяться в сеть через Интернет с помощью беспроводных технологий. Они обмениваются данными в режиме реального времени как напрямую, так и через удаленные онлайн-серверы.

Эти устройства способны работать в автоматическом режиме, но пользователь может управлять ими, в том числе дистанционно. Вот самое простое объяснение того, что такое Интернет вещей, — это сеть, в которой общаются между собой не пользователи, а устройства.

Если раньше к Интернету подключались компьютеры, ноутбуки, смартфоны и планшеты, то теперь к нему можно подсоединить практически любое устройство: смарт-часы, «умные» бытовые приборы, дата-центры и даже «умную» одежду — вплоть до кроссовок и носков.

Впервые выражение «Интернет вещей» (Internet of Things, или IoT) применил американский исследователь Кевин Эштон. Выступая перед руководством компании Procter&Gamble, он рассказал о том, как изменятся логистические схемы внутри компании после массового внедрения радиочастотных меток. Но прошло всего несколько лет, и «Интернет вещей» из бизнес-концепции превратился в повседневную реальность, доступную каждому.

В конце 2000-х годов появилась организация IPSO Alliance. Она нацелена на разработку и внедрение решений, связанных с Интернетом вещей. В 2011 году исследовательская компания Gartner включила IoT в список наиболее перспективных развивающихся технологий. А в 2012 году об IoT заговорил весь мир.

Интернет вещей — это не только множество различных приборов и датчиков, объединенных между собой проводными и беспроводными каналами связи и подключенных к сети Интернет, а тесная интеграция реального и виртуального миров, в среде которой общение осуществляется между людьми и устройствами.

Решения на базе Интернета вещей становятся сейчас все более востребованными именно потому, что дают поставщикам «умных» решений возможность получать дополнительную прибыль, — «умное» поведение может дать существенной прирост «полезности», потребительской стоимости устройства или системы. Так, вен-

тилятор, который сам выключается по достижении нужной температуры, экономит владельцу электроэнергию и поэтому может стоить для него дороже. А вентилятор, который еще и видит, когда в помещении есть люди, а когда нет, — ценен еще больше.

Но как техника может стать «умной»? Во-первых, за счет, собственно, своей конструкции — эта конструкция может быть такой, что поведение системы будет выглядеть разумным.

Во-вторых, за счет «интеллектуализации» — оснащения системы устройствами сбора информации, ее обработки и принятия решений. Такой подход позволяет обеспечить достаточно сложное и «разумное» поведение гораздо более простыми способами, чем за счет создания соответствующей конструкции.

Наконец, третий путь — поведение системы становится «разумным» вследствие того, что она взаимодействует с другими системами. Так, для экономии энергии системе отопления требуется краткосрочный прогноз погоды. Этот прогноз можно получить, установив соответствующие датчики и систему обработки информации с них, способную прогнозировать погоду (мини-метеостанцию), а можно просто запросить погоду в Интернете. И в том и в другом случае поведение системы отопления будет выглядеть разумным.

Важно, что в последнем примере, с точки зрения заказчика, система ведет себя практически одинаково — соответственно, заказчик готов заплатить за эту функциональность одну и ту же цену. Однако для поставщика такой системы организация подключения ее к Интернету будет стоить значительно дешевле, чем разработка и создание интеллектуальной метеостанции.

Благодаря интеллекту и коннективности у оборудования появляется новый набор функций. Их можно разделить на четыре группы:

- мониторинг;
- управление;
- оптимизация;
- автономность.

Каждая функция, важная и сама по себе, оказывается своего рода ступенькой для следующего уровня. Так, функция мониторинга служит основой для управления, оптимизации и автономности техники. Компания может выбирать такой набор функций, чтобы ее продукция была максимально полезной для потребителя, — и тем самым укреплять свою конкурентную позицию.

Возьмем, к примеру, автоматическую теплицу, которая самостоятельно осуществляет полив, поддержание нужной температуры, уровня освещенности и пр. Такая теплица окажется востребованной теми, кто не хочет тратить много времени на уход за растениями, а также может не иметь для этого возможности в периоды длительного отсутствия: командировок, отпуска и т. п.

Какую проблему клиента решит функция *мониторинга*? Прежде всего — устранит беспокойство насчет того, все ли в порядке с растениями во время его отсутствия:

есть ли в системе вода, не выключалось ли электричество, может ли система вентиляции обеспечить нужную температуру, если в помещении стало слишком жарко, и др. Клиент наверняка заплатит больше, если предоставить ему возможность в любой момент знать, каковы условия в его теплице. Таким образом, продажная стоимость теплицы с функцией удаленного мониторинга параметров может возрасти существенно, в то время как ее реализация для производителя будет достаточно простой. В результате применение технологии Интернета вещей позволит производителю получить дополнительную прибыль.

Еще выше потребительская стоимость будет у той же теплицы, если добавить функцию *управления*, — чтобы пользователь мог удаленно не только получать информацию об условиях в теплице, но и менять их по своему усмотрению.

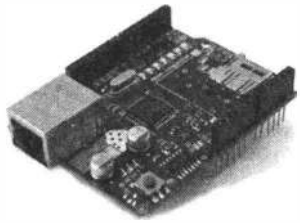
Наверняка в теплице подогрев включается автоматически, если температура падает ниже заданного предела, но, возможно, не стоит его включать, если знать, что по прогнозу погоды совсем скоро ожидается потепление? Таким образом, функция *оптимизации* за счет использования дополнительной информации позволить сэкономить деньги на содержание теплицы и получить урожай с меньшими затратами.

Наконец, средствами Интернета вещей несложно начать следить за количеством расходуемых материалов — к примеру, удобрений, — и автоматизировать их заказ либо контролировать состояние элементов, требующих замены или обслуживания: насосов, вентиляторов, нагревающих элементов, организовав таким образом самодиагностику и самообслуживание теплицы вплоть до полной ее *автономности*.

В четырех последующих главах этой книги мы познакомимся с принципами работы, а также аппаратной и программной составляющими устройств для Интернета вещей на базе популярных плат Arduino, ESP и микрокомпьютера Raspberry Pi. Чтобы Arduino, ESP и Raspberry Pi стали полноценными устройствами для Интернета вещей, их надо оснастить необходимыми датчиками и исполнительными устройствами и предоставить им доступ к сети Интернет. Соответственно, мы рассмотрим работу Arduino, ESP и Raspberry Pi с такими датчиками и устройствами, а также организацию доступа их в Интернет с дальнейшей отправкой данных в известные облачные сервисы и получением их оттуда. А последняя, шестая глава книги посвящена примерам создания различных полезных проектов Интернета вещей: от табло для отображения биржевых котировок до предсказателя погоды, использующего данные облачного сервиса ThingSpeak.

Книга сопровождается электронным архивом, содержащим исходный код большинства рассмотренных примеров и проектов, используемые в проектах необходимые библиотеки, а также техническую документацию на задействованные в проектах устройства и датчики (см. *приложение*). Этот электронный архив можно скачать с FTP-сервера издательства «БХВ» по ссылке **ftp://ftp.bhv.ru/9785977567558.zip**, а также со страницы книги на сайте **https://bhv.ru**.

ГЛАВА 2



Аппаратные платформы для создания устройств Интернета вещей

В этой главе мы рассмотрим аппаратные средства, которые можно использовать для создания устройств IoT, — доступные и популярные платформы Arduino, ESP и Raspberry Pi.

2.1. Arduino Uno — традиционная плата для моделирования

Arduino — самая популярная платформа любительской и образовательной электроники и робототехники. Скорость проектирования и разработки на Arduino намного выше, чем это можно сделать на основе других микроконтроллеров, а обусловлено это простой, но в то же время хорошо проработанной архитектурой этой платформы. Невысокая цена, легкость программирования, доступная новичкам, большое количество плат расширения, а также программных наработок в виде библиотек позволяют сосредоточиться не на написании громоздкого кода, а на творчестве и фантазии.

Arduino Uno (рис. 2.1) — флагманская платформа семейства Arduino. Плата Arduino Uno выполнена на микроконтроллере ATmega328P с тактовой частотой 16 МГц. На плате предусмотрены 20 портов входа/выхода для подключения внешних устройств — например, плат расширения или датчиков.

В настоящее время внимание сообщества Arduino в большей степени направлено на Интернет вещей. А для Интернета вещей необходим доступ микроконтроллера к сетям на основе Wi-Fi, Bluetooth или Ethernet. Сама плата Arduino Uno такими возможностями не обладает. Но можно использовать эту плату в связке с другими платами или *шилдами* (платами расширения Arduino). Кроме того, разработчики создают и другие платы на основе и в форм-факторе Arduino Uno, добавляя в них дополнительные возможности, позволяющие использовать плату в качестве устройства Интернета вещей. Примером такой разработки, пригодной для создания проектов, требующих наличия в плате Arduino возможностей Wi-Fi, может служить

плата Arduino+WiFi от компании RobotDyn (рис. 2.2), в которой интегрированы плата Arduino Uno R3 и модуль Wi-Fi ESP8266. Оба эти компонента платы могут работать вместе или каждый в отдельности.

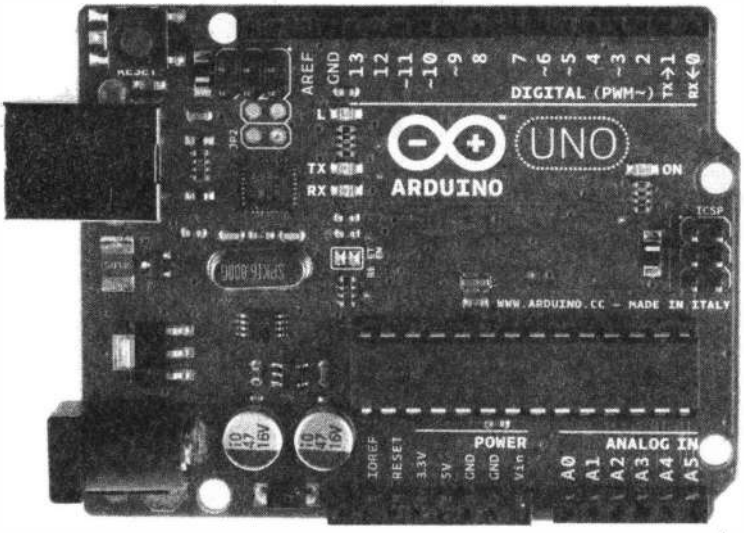


Рис. 2.1. Плата Arduino Uno

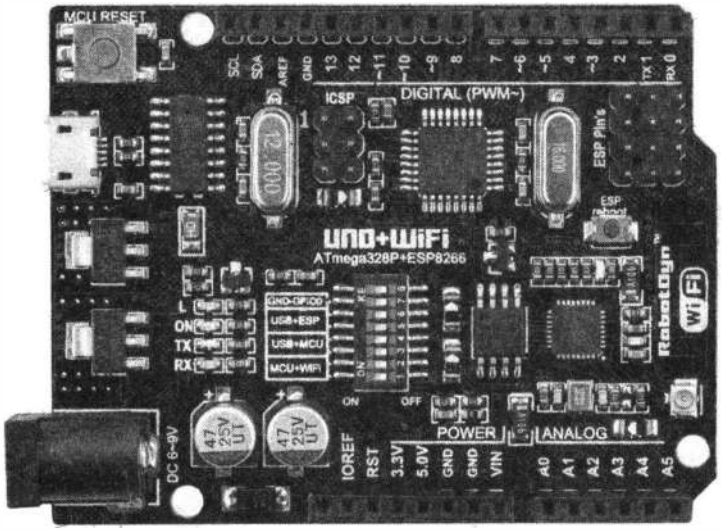


Рис. 2.2. Плата Arduino+WiFi от компании RobotDyn

Компания Things Network предоставляет совместимые с экосистемой Arduino аппаратные средства, которые позволяют легко развертывать сети LoRaWAN (см. *главу 5*). В настоящее время компания продвигает совместимую с Arduino плату The Things Uno (рис. 2.3).

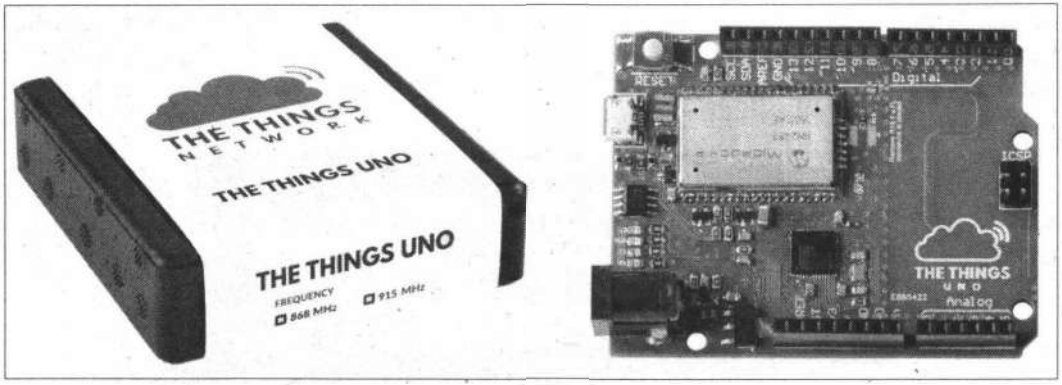


Рис. 2.3. Плата The Things Uno

Упомянутые здесь решения не единственные. И они показывают, что даже такую простую плату, как Arduino Uno, вполне можно использовать при создании проектов Интернета вещей.

2.2. Семейство плат Arduino MKR

Классический форм-фактор Arduino лег в основу всей экосистемы Arduino, но он постепенно отмирает. Uno-подобные платы, которые стали стандартом де-факто в мире любителей радиоэлектроники, медленно уходят на пенсию. Учитывая рост интереса сообщества Arduino к Интернету вещей, производители современных плат перешли к использованию форм-фактора MKR. Серия плат MKR (рис. 2.4) предос-

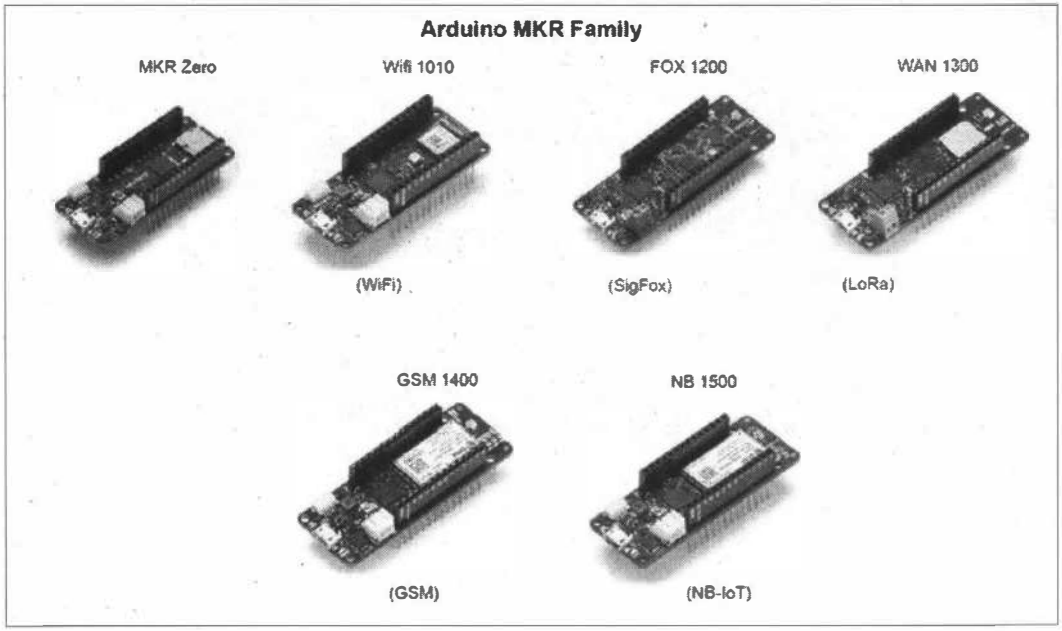


Рис. 2.4. Платы Arduino серии MKR

тавляет различные варианты подключения к сети и управления питанием, что побуждает людей использовать их в качестве единого стандартного формата для проектов IoT. Платы MKR могут повысить уровень стандартизации процесса проектирования, облегчая жизнь разработчика, собирающегося продавать свой конечный продукт на рынке.

Как и для плат классического форм-фактора Arduino, для плат MKR также выпускается множество шилдов (плат расширения), добавляющих к ним дополнительную функциональность. Некоторые из них показаны на рис. 2.5.

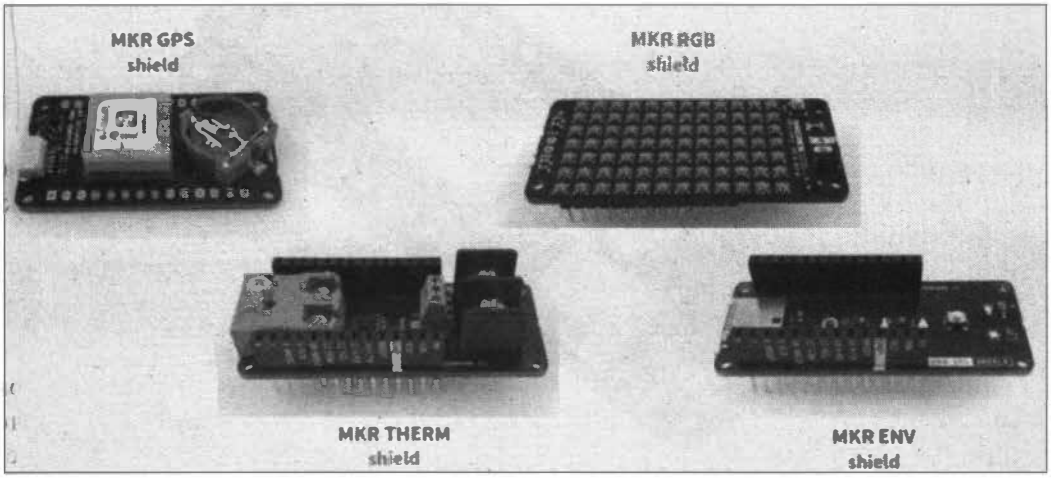


Рис. 2.5. Шилды для плат Arduino серии MKR

2.2.1. Arduino MKR1000 — с поддержкой Wi-Fi

Плата Arduino MKR1000 WiFi (рис. 2.6) предназначена для тех, кому нужно практичное, компактное и недорогое решение, сочетающее в себе функционал Wi-Fi и невысокие требования к знаниям в сетевых технологиях.

Плата основана на однокристальной системе (SoC) Atmel ATSAMW25, которая является частью семейства SmartConnect беспроводных устройств Atmel, созданных специально для проектов в области Интернета вещей. Чип ATSAMW25 состоит из трех основных блоков:

- 32-битного ARM-микроконтроллера SAMD21 на ядре Cortex-M0+ с низким энергопотреблением;
- чипа Wi-Fi WINC1500 с диапазоном 2,5 ГГц, IEEE, 802.11 b/g/n и низким энергопотреблением;
- крипто-чипа ECC508 (для защиты передаваемых данных).

В систему ATSAMW25 также встроена одна антенна, поддерживающая один канал данных и выполненная в форме печатной платы. Кроме того, дизайн платы MKR1000 предусматривает цепь, позволяющую заряжать плату от литий-ионной батареи или заряжать эту батарею при подаче на плату напряжения от внешнего 5-вольтового источника питания. Переключение с одного источника питания на

другой выполняется автоматически. Также для питания платы можно использовать напряжение 5 вольт от USB-порта.

WiFi-модуль MKR1000 поддерживает сертификат SHA-256.

Плата Arduino MKR1000 WiFi представляет собой USB-устройство, способное быть USB-хостом и работать в режиме full-speed.

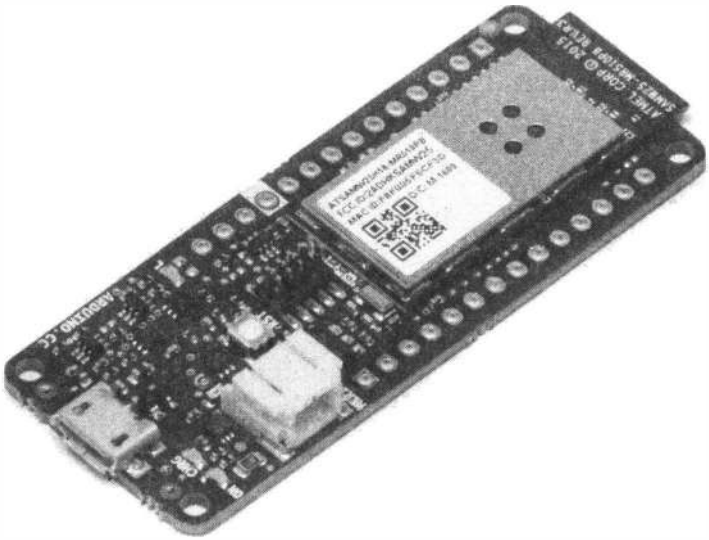


Рис. 2.6. Плата Arduino MKR1000 WiFi

Внимание!

В отличие от большинства плат Arduino/Genuino, MKR1000 работает на напряжении 3,3 вольта. То есть максимальное напряжение, к которому терпимы I/O контакты MKR1000, — это 3,3 вольта, и если его превысить, можно повредить плату. Хотя MKR1000 может коммуницировать с 5-вольтовыми цифровыми устройствами, но для такой двунаправленной коммуникации нужно правильно настроить переключение вольтовой логики.

Назначение контактов платы Arduino MKR1000 WiFi показано на рис. 2.7.

Технические характеристики платы Arduino MKR1000 WiFi представлены в табл. 2.1.

***Таблица 2.1.** Технические характеристики платы Arduino MKR1000 WiFi*

Микроконтроллер	ARM MCU SAMD21 Cortex-M0+, 32 бита, низкое энергопотребление
Wi-Fi	чип WINC1500 с диапазоном 2,5 ГГц, IEEE, 802.11 b/g/n и низким энергопотреблением
Питание платы (USB/VIN), В	5
Подключение батареи	Li-Ion 3,7 В, минимальная емкость 700 мАч
Рабочее напряжение, В	3,3
Цифровые I/O контакты	8

***Таблица 2.1** (окончание)*

Цифровые I/O контакты с поддержкой ШИМ	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 или 18, A4 или 19)
UART	1
SPI	1
I^2C	1
Входные аналоговые контакты	7 (АЦП 8/10/12 бит)
Выходные аналоговые контакты	1 (ЦАП 10 бит)
Внешние прерывания	8 (0, 1, 4, 5, 6, 7, 8, A1 или 16, A2 или 17)
Максимальная сила тока на один I/O контакт, мА	7
Flash-память, Кбайт	256
SRAM, Кбайт	32
EEPROM	нет
Тактовая частота	32,768 КГц (RTC), 48 МГц
Встроенный светодиод (LED_BUILTIN)	Контакт 6
Размер, мм	61,5×25
Вес, г	32

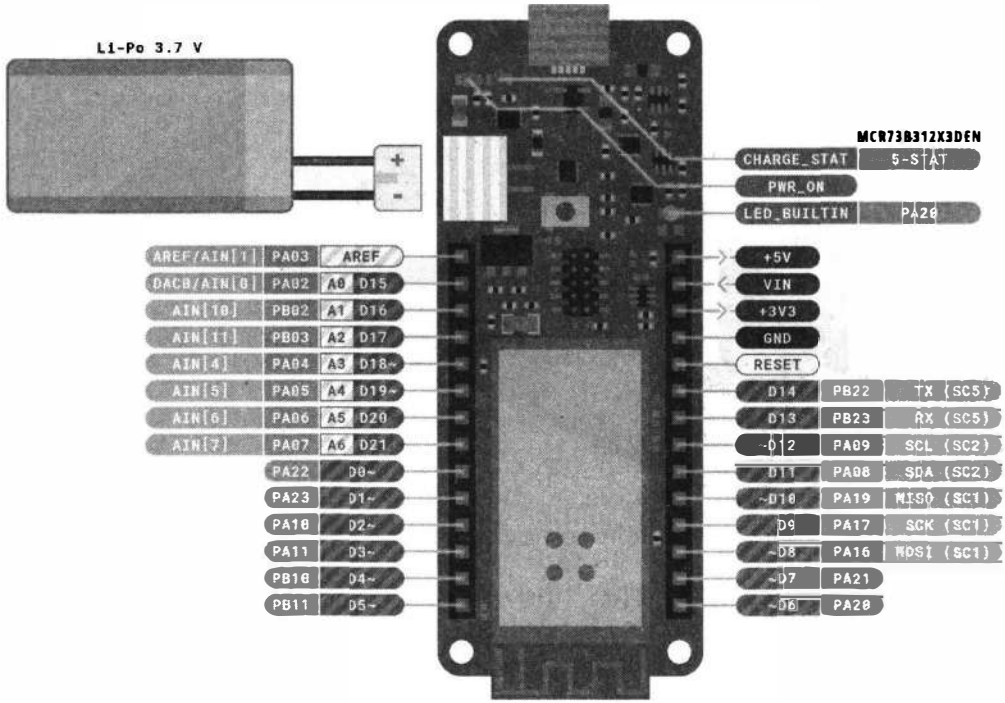


Рис. 2.7. Назначение контактов платы Arduino MKR1000 WiFi

Если вы хотите подключить к MKR1000 батарею, то вам понадобится батарея, имеющая 2-контактный мама-коннектор типа JST PHR2. Li-Ion батареи заряжены до 4,2 В при помощи тока, который составляет, как правило, половину от номинальной емкости. В плату Arduino MKR1000 WiFi встроен специальный чип, который уже заряжен на 350 мА. Это значит, что минимальная емкость Li-Ion батареи должна быть 700 мАч. Батарейные элементы с меньшей емкостью могут перегреться, наполниться газами и взорваться, тем самым вызвав возгорание устройства. Батареи с большей емкостью будут дольше заряжаться, но не сломаются и не перегреются. Чип запрограммирован на 4-часовую зарядку, после чего он автоматически переходит в режим ожидания.

Светодиод CHARGE_STAT управляется зарядным чипом, который отслеживает ток, идущий от батареи при зарядке. Как правило, он загорается, когда плата питается от USB или VIN, а чип начинает заряжать Li-Ion батарею, подключенную к коннектору JST.

Встроенный светодиод на плате Arduino MKR1000 WiFi подключен к цифровому контакту D6, а не к D13, как на других платах Arduino.

2.2.2. Arduino MKR1010 — с поддержкой Wi-Fi и Bluetooth

Плата Arduino MKR Wi-Fi 1010 (рис. 2.8) — это усовершенствованная плата MKR1000, оснащенная модулем u-blox ESP32.

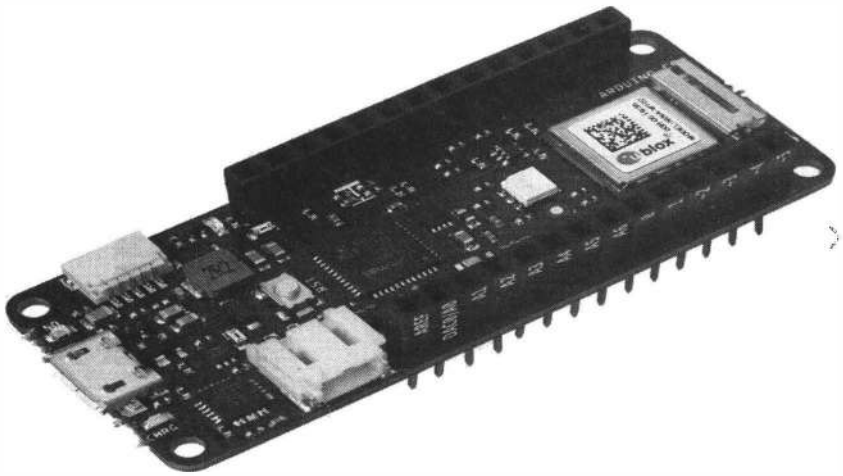


Рис. 2.8. Плата Arduino MKR Wi-Fi 1010

Мозгом платы Arduino MKR Wi-Fi 1010 является 32-разрядный микроконтроллер фирмы Microchip (Atmel) — ATSAMD21G18 с вычислительном ядром ARM Cortex M0. За беспроводную связь отвечает модуль u-blox NINA-W102 со встроенным чипом ESP32 для обмена данными в диапазоне 2,4 ГГц по Wi-Fi и Bluetooth. Регулировка выходной мощности обеспечивает оптимальное соотношение между даль-

ностью связи, скоростью передачи данных и энергопотреблением. Коммуникационный чипсет платы Arduino MKR Wi-Fi 1010 может быть клиентом и хост-устройством Bluetooth и BLE.

Для защиты передаваемых данных используется крипто-чип ECC508.

Разъем micro-USB предназначен для прошивки платформы Arduino M0 с помощью компьютера. На плате расположен JST PH-разъем (2 pin) для подключения внешних Li-Pol и Li-Ion аккумуляторов.

При одновременном питании платформы от USB и аккумулятора батарея заряжается через контролер заряда BQ24195L до 4,2 В, светодиод индикации питания ON горит, светодиод индикации заряда батареи CHRG горит.

Линейный понижающий регулятор напряжения AP7215-33 с выходом 3,3 В и максимальным выходным током 600 мА обеспечивает питание микроконтроллера.

На платформе предусмотрен JST SH-разъем (5 pin) для подключения дополнительных модулей по интерфейсу I²C.

Контакты 2(SCK/BCLK), 3(WS/LRCLK/FS) и A6(SD/SDATA/SDIN/SDOUT) используются для передачи и приема цифрового звука с другими аудиоустройствами.

> ***Внимание!***
>
> В отличие от большинства плат Arduino/Genuino, MKR1010 работает на напряжении 3,3 вольта. То есть максимальное напряжение, к которому терпимы I/O контакты MKR1010 — это 3,3 вольта, и если его превысить, можно повредить плату. Хотя MKR1010 может коммуницировать с 5-вольтовыми цифровыми устройствами, но для такой двунаправленной коммуникации нужно правильно настроить переключение вольтовой логики.

Назначение контактов платы Arduino MKR Wi-Fi 1010 показано на рис. 2.9.

Технические характеристики платы Arduino MKR WiFi 1010 представлены в табл. 2.2.

***Таблица 2.2.** Технические характеристики платы Arduino MKR Wi-Fi 1010*

Микроконтроллер	ARM MCU SAMD21 Cortex-M0+, 32 бита, низкое энергопотребление
Wi-Fi и Bluetooth	u-blox NINA-W102
Питание платы (USB/VIN), В	5–6
Подключение батареи	Li-ion 3,7 В, минимальная емкость 1024 мАч
Рабочее напряжение, В	3,3
Цифровые I/O контакты	8
Цифровые I/O контакты с поддержкой ШИМ	13 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, A3 или 18, A4 или 19)
UART	1
SPI	1
I²C	1

Таблица 2.2 (окончание)

Входные аналоговые контакты	7 (АЦП 8/10/12 бит)
Выходные аналоговые контакты	1 (ЦАП 10 бит)
Внешние прерывания	8 (0, 1, 4, 5, 6, 7, 8, A1 или 16, A2 или 17)
Максимальная сила тока на один I/O контакт, мА	7
Flash-память, Кбайт	256
SRAM, Кбайт	32
EEPROM	нет
Тактовая частота	32,768 КГц (RTC), 48 МГц
Встроенный светодиод (LED_BUILTIN)	Контакт 6
USB	Полноскоростное USB-устройство и встроенный хост
Размер, мм	61,5×25
Вес, г	32

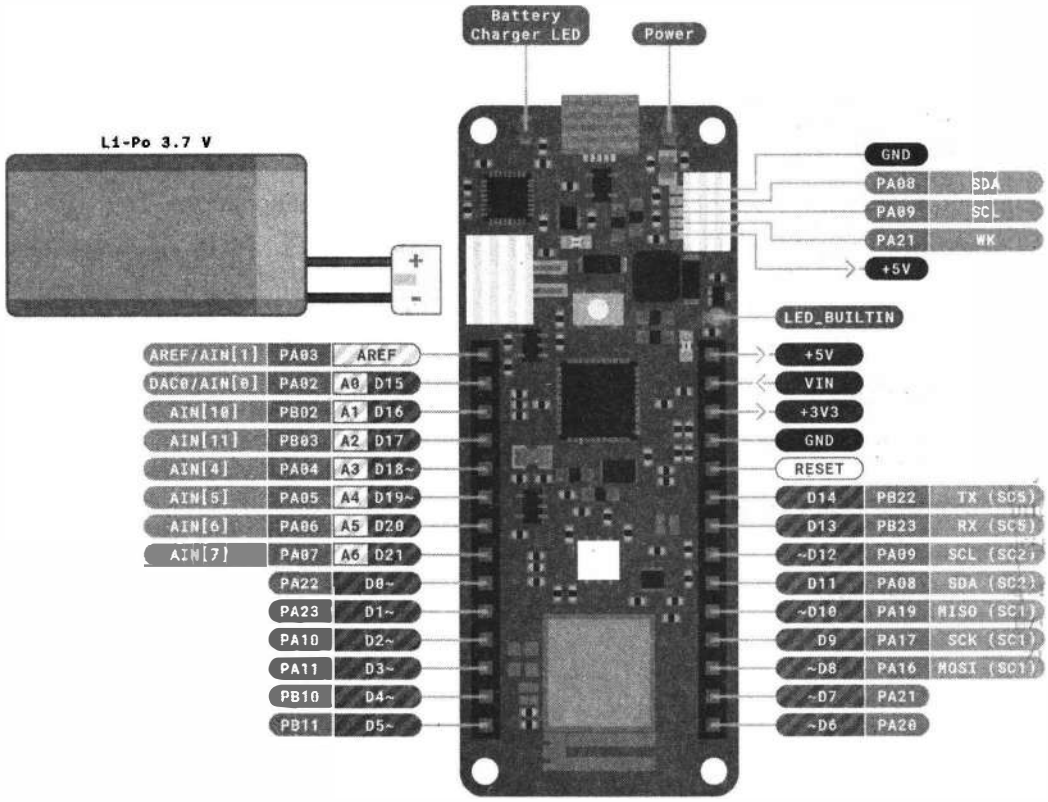


Рис. 2.9. Назначение контактов платы Arduino MKR Wi-Fi 1010

2.2.3. Arduino MKR GSM 1400 — с поддержкой GSM-связи

Плата Arduino MKR GSM 1400 (рис. 2.10) была разработана, чтобы предложить практическое и экономически эффективное решение для конструкторов-любителей с минимальным опытом работы в сети, желающих добавить GSM-связь к своим проектам.

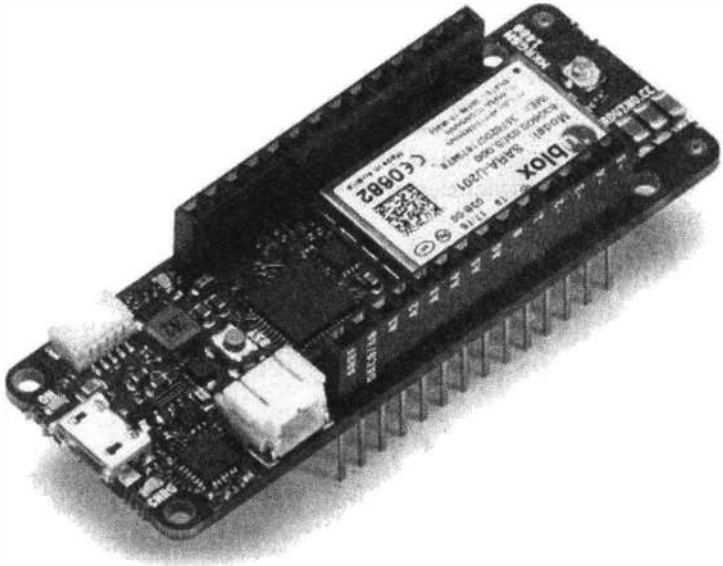


Рис. 2.10. Плата Arduino MKR GSM 1400

Мозгом платы Arduino MKR GSM 1400 является 32-разрядный микроконтроллер фирмы Microchip (Atmel) — ATSAMD21G18 с вычислительном ядром ARM Cortex M0. Модуль u-blox SARA-U201 обеспечивает сотовую связь 3G/2G с поддержкой UMTS/HSPA и GSM/GPRS. Поддержка стандарта 3G обеспечивает входящую скорость передачи данных до 7,2 Мбит/с и исходящую до 5,76 Мбит/с. Для пользования сотовой связью понадобится SIM-карта формата Nano SIM, которая устанавливается с обратной стороны платы. В зоне слабого приема необходимо использовать дополнительную антенну усиления GSM-сигнала, которая подключается через разъем U.FL.

Разъем micro-USB предназначен для прошивки платформы Arduino M0 с помощью компьютера. На плате расположен JST PH-разъем (2 pin) для подключения внешних Li-Pol и Li-Ion аккумуляторов.

При одновременном питании платформы от USB и аккумулятора батарея заряжается через контроллер заряда BQ24195L до 4,2 В, светодиод индикации питания ON горит, светодиод индикации заряда батареи CHRG горит.

Линейный понижающий регулятор напряжения AP7215-33 с выходом 3,3 В и максимальным выходным током 600 мА обеспечивает питание микроконтроллера.

Во время сотовых передач максимальный ток, потребляемый платой, будет превышать 500 мА. Это сверх того, что может быть получено с помощью стандартного USB-порта. Поэтому при питании платы от USB обязательно нужно подключить к плате Li-Po батарею емкостью не менее 1500 мАч — ток, обеспечиваемый USB-портом, будет дополнен батареей. При питании платы с помощью VIN требуется источник питания 5 В не менее 2 А.

На платформе предусмотрен JST SH-разъем (5 pin) для подключения дополнительных модулей по интерфейсу I²C.

> ***Внимание!***
>
> **В отличие от большинства плат Arduino/Genuino, MKR GSM 1400 работает на напряжении 3,3 вольта. То есть максимальное напряжение, к которому терпимы I/O контакты MKR GSM 1400, — это 3,3 вольта, и если его превысить, можно повредить плату. Хотя MKR GSM 1400 может коммуницировать с 5-вольтовыми цифровыми устройствами, но для такой двунаправленной коммуникации нужно правильно настроить переключение вольтовой логики.**

Назначение контактов платы Arduino MKR GSM 1400 показано на рис. 2.11.

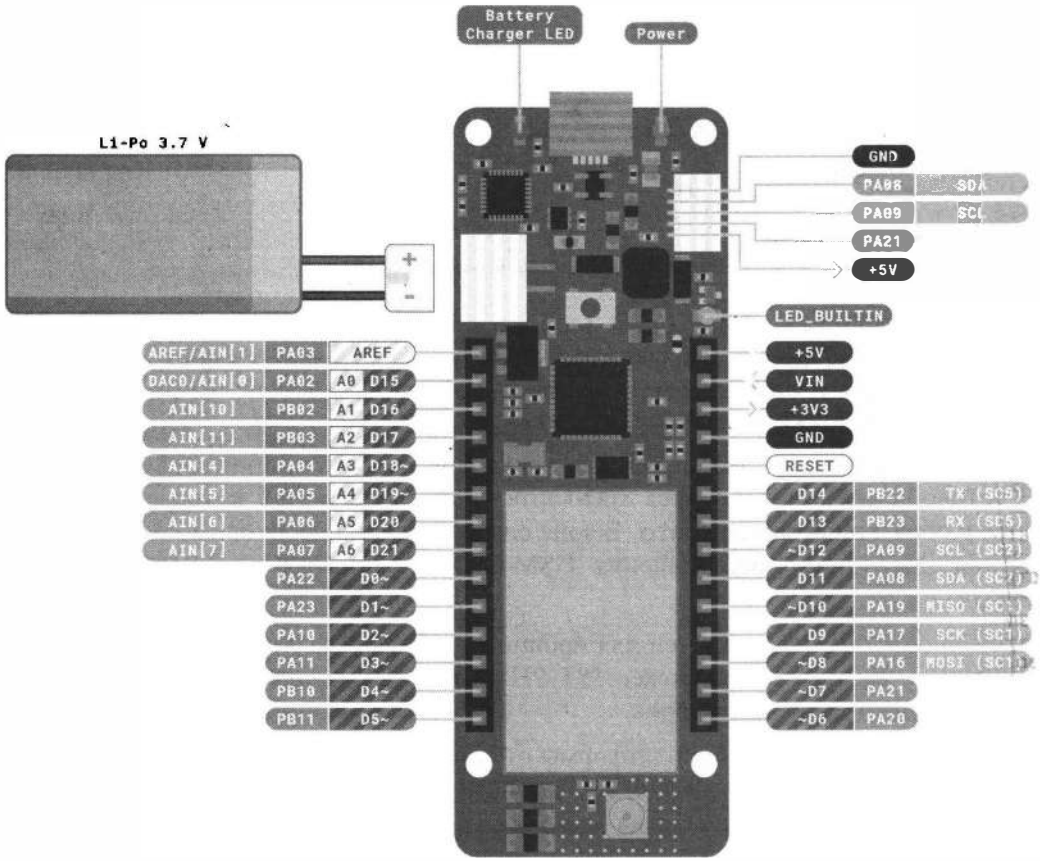


Рис. 2.11. Назначение контактов платы Arduino MKR GSM 1400

Технические характеристики платы Arduino MKR GSM 1400 представлены в табл. 2.3.

Таблица 2.3. *Технические характеристики платы Arduino MKR GSM 1400*

Микроконтроллер	ARM MCU SAMD21 Cortex-M0+, 32 бита, низкое энергопотребление
Модуль сотовой связи	u-blox SARA-U201
Размер SIM-карты	Nano SIM
Стандарты связи	3G UMTS/HSPA и 2G GSM/GPRS
Несущие частоты, МГц	850/900/1800/1900/2100
Максимальная входящая скорость, Мбит/с	7,2
Максимальная исходящая скорость, Мбит/с	5,76
Питание платы (USB/VIN), В	5–6
Подключение батареи	Li-ion 3,7 В, минимальная емкость 1500 мАч
Рабочее напряжение, В	3,3
Цифровые I/O контакты	8
Цифровые I/O контакты с поддержкой ШИМ	13 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, A3 или 18, A4 или 19)
UART	1
SPI	1
I^2C	1
Входные аналоговые контакты	7 (АЦП 8/10/12 бит)
Выходные аналоговые контакты	1 (ЦАП 10 бит)
Внешние прерывания	8 (0, 1, 4, 5, 6, 7, 8, A1 или 16, A2 или 17)
Максимальная сила тока на один I/O контакт, мА	7
Flash-память, Кбайт	256
SRAM, Кбайт	32
EEPROM	нет
Тактовая частота	32,768 КГц (RTC), 48 МГц
Встроенный светодиод (LED_BUILTIN)	Контакт 6
Размер, мм	68×25
Вес, г	32

2.3. Семейство плат Arduino Nano 33

В 2019 году Arduino запустило серию плат Nano нового поколения. Платы имеют тот же размер, что и оригинальные платы Nano, но оснащены новыми процессорами и обладают более низким энергопотреблением. В семейство входят платы

Arduino Nano Every, Arduino Nano 33 IoT, Arduino Nano 33 BLE и Arduino Nano 33 BLE Sense (рис. 2.12).

Оригинальные платы Arduino Nano были меньше других плат Arduino, но не дешевле. С характеристиками, аналогичными платам Uno, платы Nano представляли собой просто классические платы Uno с классическим форм-фактором. И хотя новые платы занимают в схеме такое же место, что и оригинальные платы Nano, они тем не менее от оригинальных плат немного отличаются, поскольку занимаемое место и форм-фактор — это не одно и то же.

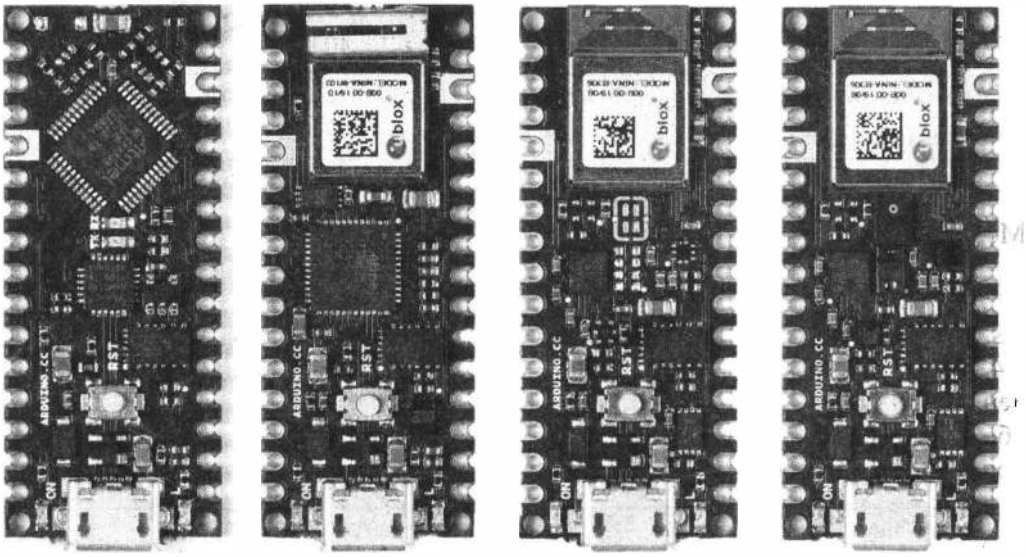


Рис. 2.12. Новое семейство Arduino Nano (слева направо): Nano Every, Nano 33 IoT, Nano 33 BLE и Nano 33 BLE Sense

Существенная разница здесь заключается в том, что, в отличие от оригинальных плат Nano, новые платы Nano имеют форм-фактор с краевыми полуотверстиями, что позволяет припаять их непосредственно к другой печатной плате. Так что платы нового семейства Nano на самом деле предназначены не столько для самостоятельной работы, сколько для поверхностного монтажа в виде модулей.

Далее мы рассмотрим платы Arduino Nano 33, предназначенные для создания устройств IoT.

2.3.1. Arduino Nano 33 IoT — с поддержкой Wi-Fi и Bluetooth BLE

Плата Arduino Nano 33 IoT (рис. 2.13) является простым и дешевым решением для внедрения существующих, а также создания новых, устройств в системы IoT. Эта небольшая, надежная и мощная плата имеет подключение Wi-Fi и Bluetooth, что в сочетании с архитектурой с низким энергопотреблением делает ее практичным и экономически эффективным решением для ваших проектов.

Рис. 2.13. Плата Arduino Nano 33 IoT

Мозгом платы Arduino Nano 33 IoT является 32-разрядный микроконтроллер фирмы Microchip (Atmel) — ATSAMD21G18 с вычислительном ядром ARM Cortex M0. За беспроводную связь отвечает модуль u-blox NINA-W102 со встроенным чипом ESP32. Радиомодуль обеспечивает поддержку Wi-Fi 802.11b/g/n в диапазоне ISM 2,4 ГГц и Bluetooth BLE 4.2 (Bluetooth BR/EDR и Bluetooth с низким энергопотреблением). Крипточип ATECC608A хранит криптографические ключи в аппаратном обеспечении, гарантируя очень высокий уровень безопасности для этого класса продуктов.

6-осевая система инерциальной ориентации (IMU) — блок инерциальных датчиков LSM6DSL iNEMO фирмы STMicroelectronics с 3D акселерометром и 3D гироскопом и управлением через шину I^2C.

Импульсный понижающий регулятор напряжения MPM3610 обеспечивает питание микроконтроллера и другой логики платы при подключении ее через пин VIN. Диапазон входного напряжения от 5 до 21 В. Выходное напряжение 3,3 В с максимальным выходным током 1,2 А.

Плата может быть использована либо в макете (при монтаже штыревых разъемов), либо в качестве SMT-модуля, будучи непосредственно припаянной через зубчатые площадки.

> ***Внимание!***
>
> В отличие от большинства плат Arduino/Genuino, Arduino Nano 33 IoT работает на напряжении 3,3 вольта. То есть максимальное напряжение, к которому терпимы I/O контакты Arduino Nano 33 IoT, — это 3,3 вольта, и если его превысить, можно повредить плату. Хотя Arduino Nano 33 IoT может коммуницировать с 5-вольтовыми цифровыми устройствами, но для такой двунаправленной коммуникации нужно правильно настроить переключение вольтовой логики.

Назначение контактов платы Arduino Nano 33 IoT показано на рис. 2.14.

Технические характеристики платы Arduino Arduino Nano 33 IoT представлены в табл. 2.4.

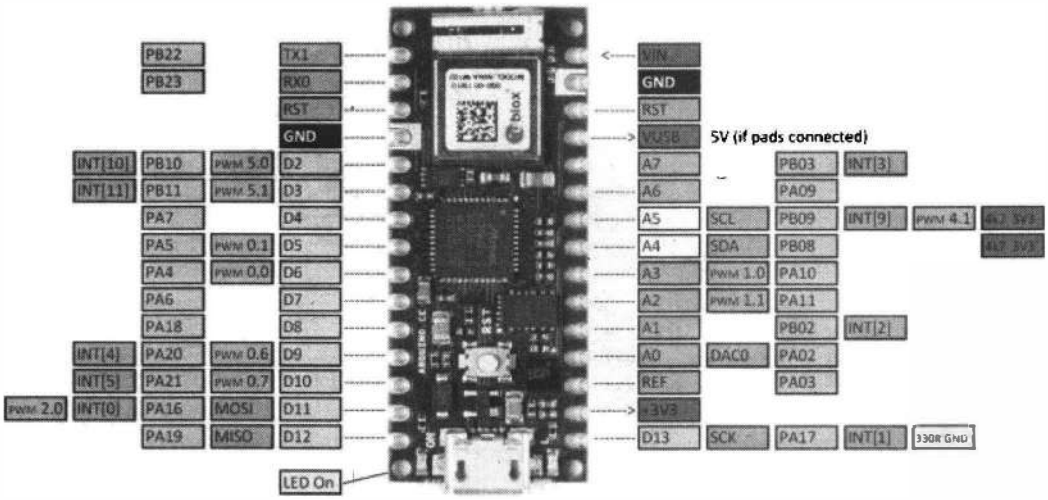


Рис. 2.14. Назначение контактов платы Arduino Nano 33 IoT

Таблица 2.4. *Технические характеристики платы Arduino Nano 33 IoT*

Микроконтроллер	ARM MCU SAMD21 Cortex-M0+, 32 бита, низкое энергопотребление
Модуль сотовой связи	u-blox NINA-W102
Питание платы VIN, В	5–21
Рабочее напряжение, В	3,3
Цифровые I/O контакты	14
Цифровые I/O контакты с поддержкой ШИМ	11 (2, 3, 5, 6, 9, 10, 11, 12, 16 или A2, 17 или A3, 19 или A5)
UART	1
SPI	1
I^2C	1
Входные аналоговые контакты	8 (АЦП 8/10/12 бит)
Выходные аналоговые контакты	1 (ЦАП 10 бит)
Внешние прерывания	Все цифровые контакты
Максимальная сила тока на один I/O контакт, мА	7
Flash-память, Кбайт	256
SRAM, Кбайт	32
Тактовая частота	32,768 КГц (RTC), 48 МГц
IMU	LSM6DS3
Встроенный светодиод (LED_BUILTIN)	Контакт 13
Размер, мм	45×18
Вес, г	5

2.3.2. Arduino Nano 33 BLE, Arduino Nano 33 BLE Sence — для создания носимых устройств с минимальным электропотреблением

Arduino Nano 33 BLE и Arduino Nano 33 BLE Sense (рис. 2.15) — компактные платформы для разработки на чипе u-blox NINA-B306 с микроконтроллером Nordic nRF52840 и беспроводным модулем Bluetooth BLE. Основной процессор включает в себя другие удивительные функции — такие, как сопряжение Bluetooth через NFC и режимы сверхнизкого энергопотребления.

На плате также распаян IMU-модуль на 9 степеней свободы, который содержит трехосевые сенсоры: акселерометр, гироскоп и магнитометр, что позволяет создать на Arduino Nano BLE собственный фитнес-браслет, умные часы или другой мобильный проект с беспроводной связью по Bluetooth.

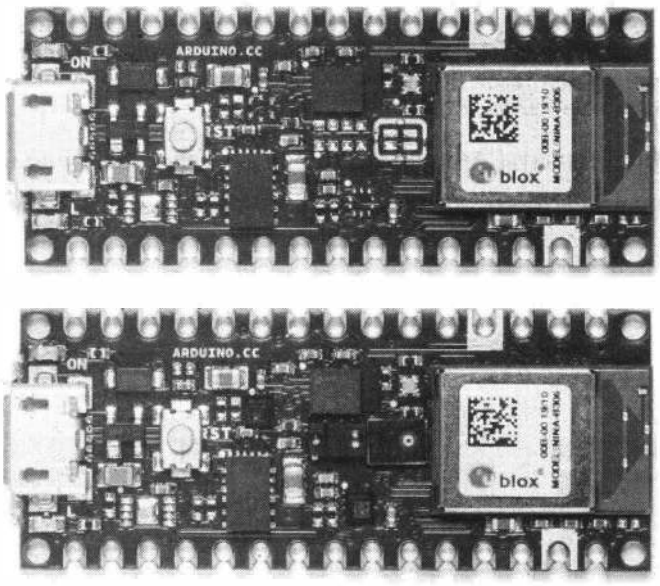


Рис. 2.15. Платы Arduino Nano 33 BLE (верхняя) и Arduino Nano 33 BLE Sence

Импульсный понижающий регулятор напряжения MPM3610 обеспечивает питание микроконтроллера и другой логики платы при подключении ее через пин VIN. Диапазон входного напряжения от 5 до 18 В. Выходное напряжение 3,3 В с максимальным выходным током 1,2 А.

Отличие модуля Arduino Nano 33 BLE Sense от Arduino Nano 33 BLE заключается в том, что он — в дополнение к 9-осевой IMU — поставляется с гораздо большим набором датчиков: датчиками давления, влажности, температуры и света, а также датчиком жестов и встроенным микрофоном:

- датчик HTS221 определяет температуру и относительную влажность воздуха в окружающем пространстве и выдает их значения в 16-битном формате;

- датчик атмосферного давления LPS22HB служит альтиметром для носимого гаджета или барометром для метеостанции;
- датчик Avago APDS-9960 от Broadcom использует четыре фотодиода с ИК-излучателями для измерения расстояния и распознавания базовых жестов: взмаха руки влево или вправо, вверх-вниз и вперед-назад. Также он умеет распознавать цвета через интенсивность каналов RGB и уровень освещенности;
- встроенный цифровой микрофон MP34DT05 пригодится для распознавания коротких голосовых команд или записи звука.

Платы Arduino Nano 33 BLE и Arduino Nano 33 BLE Sense могут быть использованы либо в макете (при монтаже штыревых разъемов), либо в качестве SMT-модуля, будучи непосредственно припаянными через зубчатые площадки.

> ***Внимание!***
>
> **В отличие от большинства плат Arduino/Genuino, Arduino Nano 33 BLE работает на напряжении 3,3 вольта. То есть максимальное напряжение, к которому терпимы I/O контакты Arduino Nano 33 BLE, — это 3,3 вольта, и если его превысить, можно повредить плату. Хотя Arduino Nano 33 BLE может коммуницировать с 5-вольтовыми цифровыми устройствами, но для такой двунаправленной коммуникации нужно правильно настроить переключение вольтовой логики.**

Назначение контактов платы Arduino Nano 33 BLE Sence показано на рис. 2.16.

Технические характеристики платы Arduino Nano 33 BLE Sence представлены в табл. 2.5.

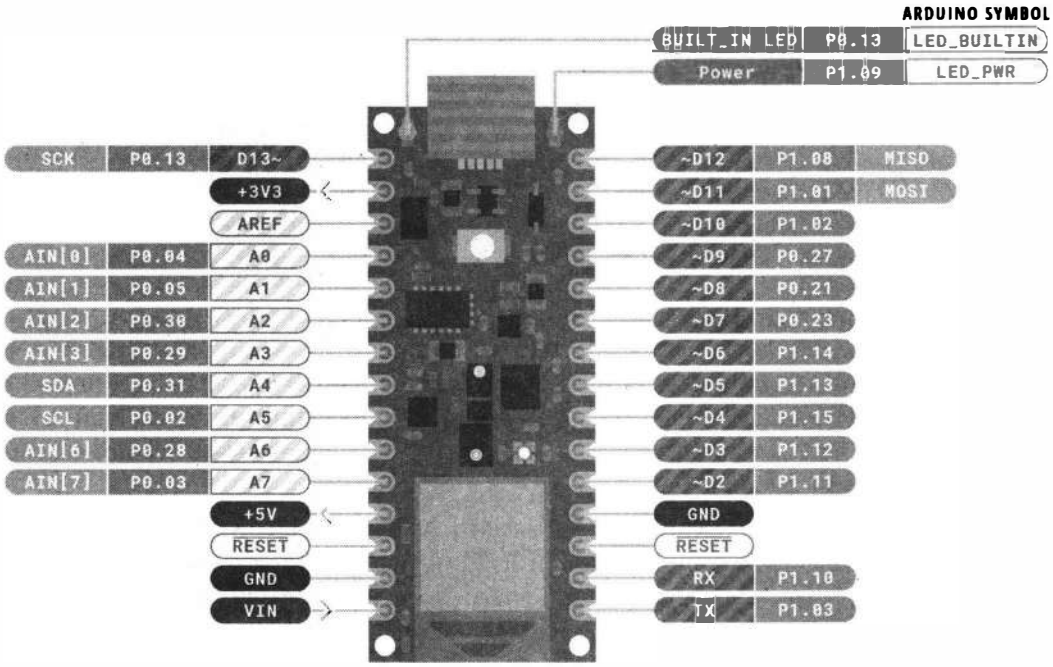


Рис. 2.16. Назначение контактов платы Arduino Nano 33 BLE Sence

Таблица 2.5. Технические характеристики платы Arduino Nano 33 BLE Sence

Микроконтроллер	nRF52840
Питание платы VIN, В	5–21
Рабочее напряжение, В	3,3
Цифровые I/O контакты	14
Цифровые I/O контакты с поддержкой ШИМ	Все цифровые контакты
UART	1
SPI	1
I2C	1
Входные аналоговые контакты	8 (АЦП 8/10/12 бит)
Выходные аналоговые контакты	–
Внешние прерывания	Все цифровые контакты
Максимальная сила тока на один I/O контакт, мА	15
Flash-память, Мбайт	1
SRAM, Кбайт	256 (nRF52840)
Тактовая частота	32,768 КГц (RTC), 48 МГц
IMU	LSM9DS1
Встроенный светодиод (LED_BUILTIN)	Контакт 13
Размер, мм	45×18
Вес, г	5

2.4. ESP32 — серия недорогих микроконтроллеров с интегрированными модулями Wi-Fi и Bluetooth

Модули Wi-Fi ESP8266 за время своего существования стали поистине народными и получили широкое распространение в любительской разработке устройств Интернета вещей. Но жизнь не стоит на месте, и компания-разработчик Espressif выпустила новый микроконтроллер — ESP32. ESP32 получил по сравнению с ESP8266 значительный прирост в производительности — его вычислительная мощность возросла в четыре раза. У ESP32 есть два ядра, каждое из которых работает на частоте 160 МГц (ESP8266 имеет 1 ядро, работающее на частоте 80 МГц). Контроллер несет на борту 520 Кбайт оперативной памяти и 448 Кбайт flash-памяти. Поддерживает не только Wi-Fi (802.11n с максимальной скоростью 150 Мбит в секунду), но и Bluetooth 4.2 BR/EDR и Low Energy.

Основным недостатком плат ESP8266 было очень малое количество контактов — в ESP32 этот недостаток устранен, выводов стало гораздо больше и они многофункциональные. Блок ввода/вывода имеет специальный мультиплексор, который

позволяет назначать различные функции на один вывод микроконтроллера. Значительно увеличено количество аналоговых входов (18 АЦП (12 бит) и 2 ЦАП (8 бит)), обеспечена поддержка PWM на всех контактах, 10 портов могут работать в режиме сенсорных кнопок. ESP32 имеет три порта UART, два I^2C, четыре SPI, два I^2S. Также имеется инфракрасный контроллер (прием/передача), шина CAN 2.0. Есть еще датчик температуры и датчик Холла. Для шифрования при передаче данных по Wi-Fi в ESP32 имеются криптографические модули AES и SHA. Блок-схема периферии ESP32 показана на рис. 2.17.

Для удобной работы с микроконтроллером ESP32 был выпущен модуль WROOM-32 (рис. 2.18). Назначение контактов модуля WROOM-32 показано на рис. 2.19.

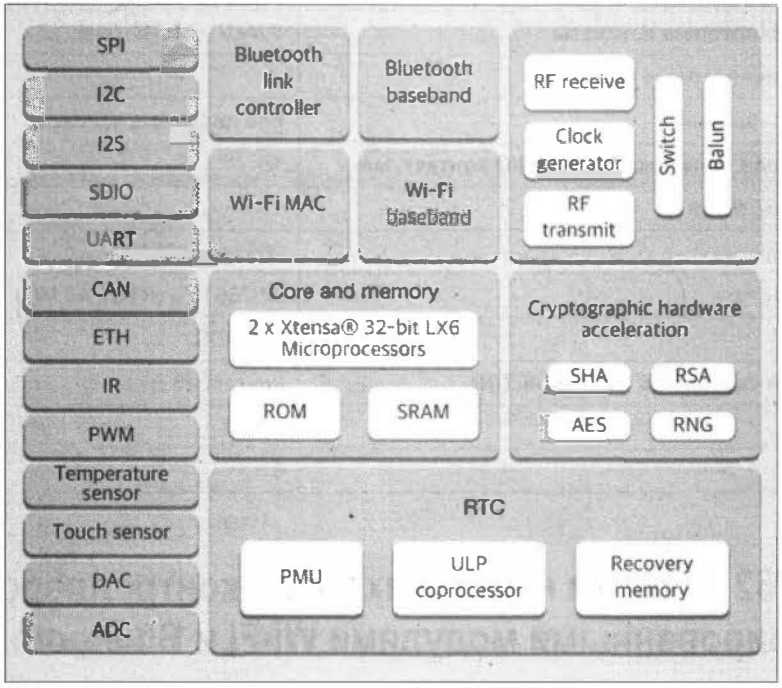


Рис. 2.17. Блок-схема периферии ESP32

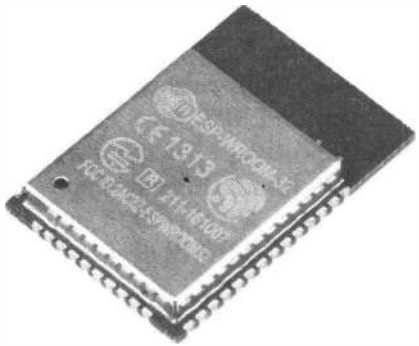


Рис. 2.18. Модуль WROOM-32 с микроконтроллером ESP32

И теперь появилось множество отладочных плат на этом модуле — например, плата ESP32 DEV Board (рис. 2.20). Назначение контактов платы ESP32 DEV Board показано на рис. 2.21.

Немаловажный вопрос в плане использования модулей для устройств Интернета вещей — энергопотребление. Максимальный ток потребления модуля ESP32 в режиме передачи Wi-Fi или Bluetooth — 160–260 мА, без включенных Wi-Fi или Bluetooth — 20 мА, в спящем режиме — 10 мкА.

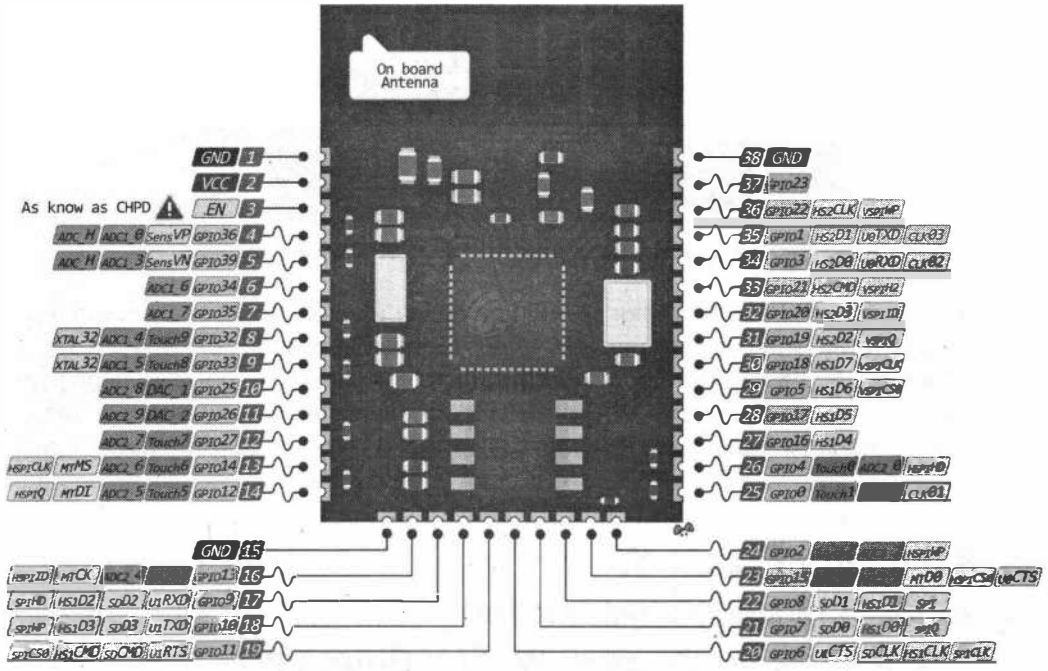


Рис. 2.19. Назначение контактов модуля WROOM-32

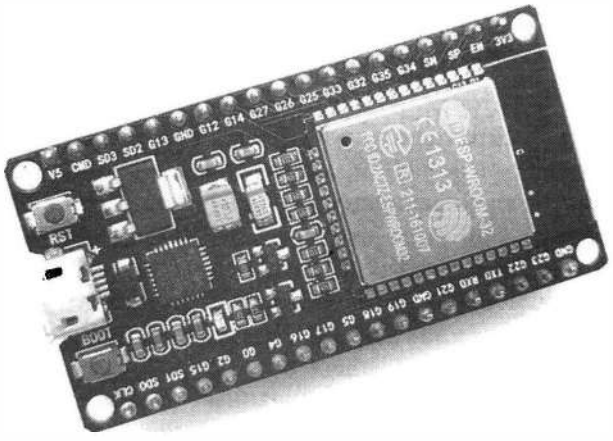


Рис. 2.20. Плата ESP32 DEV Board

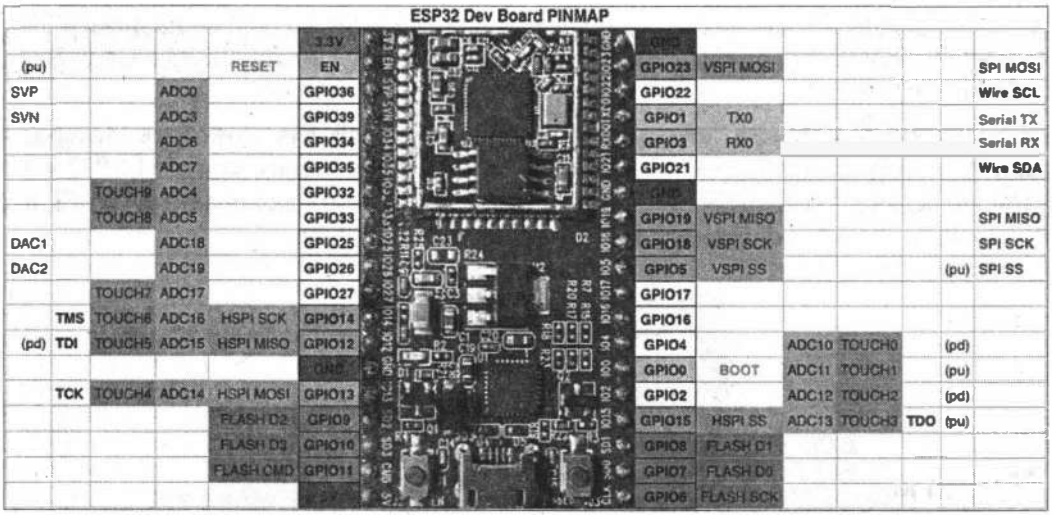


Рис. 2.21. Назначение контактов платы ESP32 DEV Board

ESP32 не заменит ESP8266 с точки зрения простоты и цены, но он является ценным членом семейства микроконтроллеров с поддержкой средств для IoT. Хотя он и стоит дороже ESP8266, но его высокая производительность, богатая периферия и возможности подключения по Wi-Fi и Bluetooth, позволят применять этот микроконтроллер в требовательных к вычислительным ресурсам приложениях Интернета вещей.

2.5. Raspberry Pi Zero W — полноценный микрокомпьютер с добавлением поддержки Wi-Fi и Bluetooth

Raspberry Pi Zero W (рис. 2.22) — это одноплатный компьютер на базе SoC Broadcom BCM2835, включающий центральный процессор ARM1176JZ-F с тактовой частотой 1,5 ГГц, графический процессор VideoCore IV с тактовой частотой 400 МГц и 512 Мбайт оперативной памяти SDRAM LPDDR2.

Raspberry Pi Zero W — вторая модель миниатюрного одноплатного компьютера линейки Raspberry Pi. По сравнению с Raspberry Pi Zero Version 1.3, Zero W (Wireless) не претерпел других изменений, кроме добавления чипа беспроводной связи Cypress CYW43438, поддерживающего сети Wi-Fi стандарта 802.11n и Bluetooth 4.1 (Bluetooth Classic и LE), при этом антенна является частью макетной платы. Этот чип используется и на Raspberry Pi 3 Model B. Другие характеристики Raspberry Pi Zero W остались без изменений: SoC, ОЗУ, разъем mini-HDMI, слот для microSD, два разъема micro-USB (один для питания, второй для подключения периферийных устройств USB), интерфейс CSI (Camera Serial Interface) для подключения камеры по интерфейсу MIPI, а также три нераспаянных разъема: 40-контактный разъем расширенного ввода/вывода (GPIO), композитный видеовыход (TV) и разъем (RUN) для сброса (рис. 2.23). Все электронные компоненты распаяны на верхней

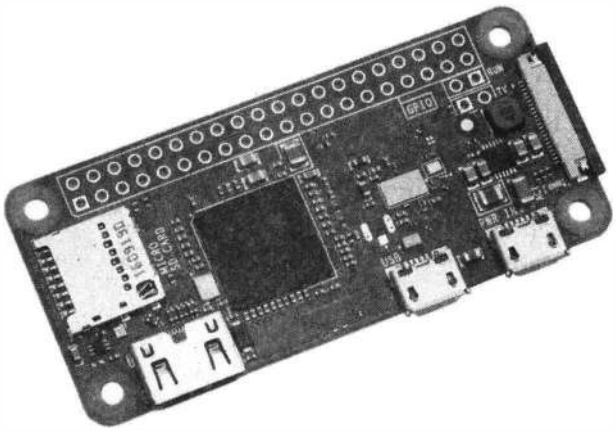


Рис. 2.22. Микрокомпьютер Raspberry Pi Zero W

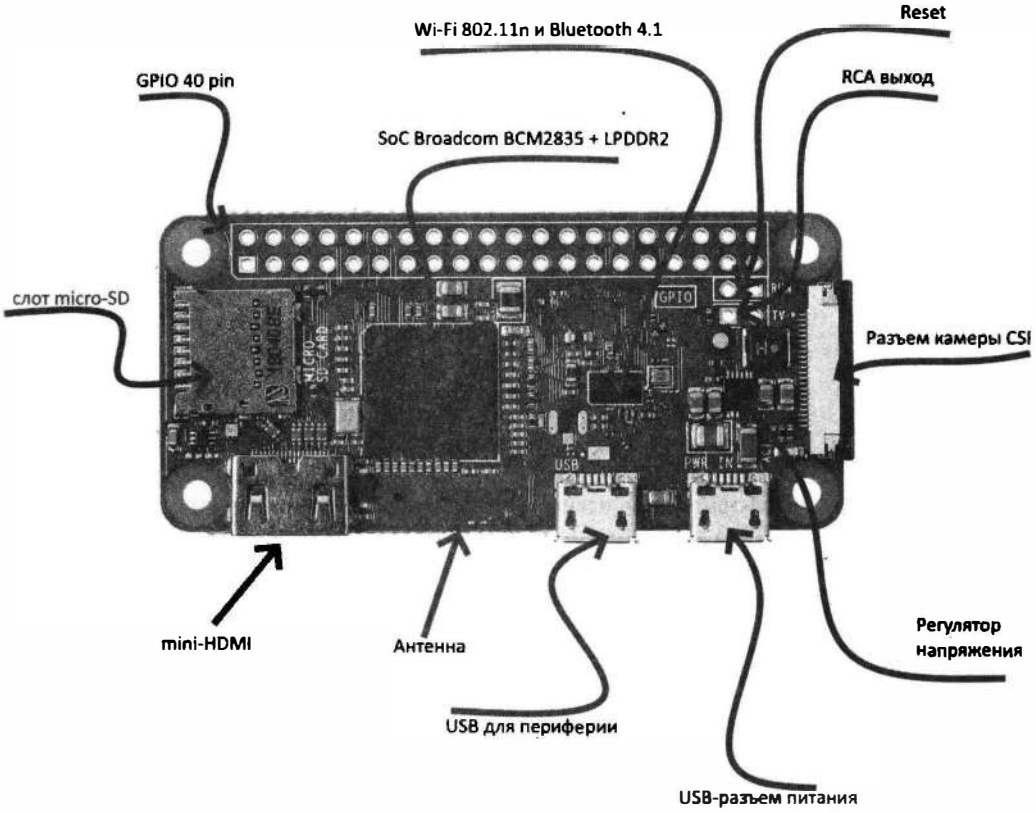


Рис. 2.23. Микрокомпьютер Raspberry Pi Zero W

части платы. Необычное решение — модули процессора и оперативной памяти напаяны друг на друга. Снизу располагается SoC Broadcom BCM2835, а прямо над ним сверху размещен модуль LPDDR2 памяти Elpida B4432BBPA-10-F емкостью 512 Мбайт.

Для подключения видеоустройств Raspberry Pi Zero имеет разъем mini-HDMI, который способен воспроизводить видеофайлы со стабильной частотой в 60 FPS и FullHD-разрешением. Устройство поддерживает подключение камер благодаря порту CSI. Питание Raspberry Pi Zero W осуществляется от 5 В адаптера через разъем micro-USB или пины питания. Рекомендуется использовать источник питания с силой тока 2 А. 40-пиновый GPIO-интерфейс Raspberry Pi Zero W (рис. 2.24) идентичен GPIO на Raspberry Pi 3. То есть все платы расширения, продаваемые для большой «малинки», можно использовать и на Zero W, не опасаясь каких-либо проблем с совместимостью.

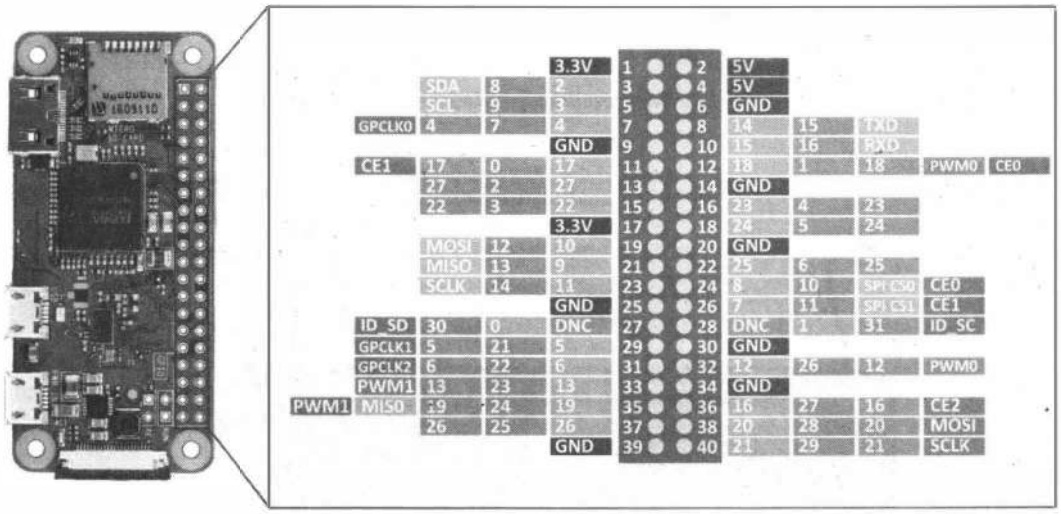


Рис. 2.24. GPIO-интерфейс Raspberry Pi Zero W

Внимание!

В отличие от платформ с логическим напряжением 5 В, напряжение логических уровней Raspberry Pi — 3,3 В. Выходы для логической единицы выдают 3,3 В, а в режиме входа ожидают принимать не более 3,3 В. Более высокое напряжение может повредить микрокомпьютер. Будьте внимательны при подключении периферии — убедитесь, что она может корректно функционировать в этом диапазоне напряжений.

Технические характеристики Raspberry Pi Zero W представлены в табл. 2.6.

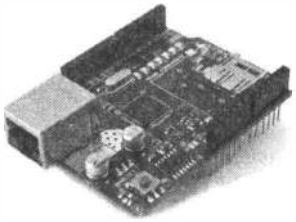
Таблица 2.6. *Технические характеристики Raspberry Pi Zero W*

Система на кристалле (SoC)	Broadcom BCM2835 (CPU, GPU, DSP и SDRAM)
Процессор	32-битный 1-ядерный ARMv6Z ARM1176JZF-S с тактовой частотой 1,5 ГГц, 16 Кбайт cache L1 и 128 Кбайт cache L2 (семейство ARM11)
Графический процессор	Двухъядерный GPU VideoCore IV с тактовой частотой 250 МГц поддерживает стандарты OpenGL ES 2.0, OpenVG, MPEG-2, VC-1 и способен кодировать, декодировать и выводить Full HD-видео (1080p, 30 FPS, H.264 High-Profil)

***Таблица 2.6** (окончание)*

ОЗУ	512 Майт SDRAM LPDDR2 400 МГц (совместно с GPU)
Хранилище	слот для карты памяти MicroSDHC
Дата выхода	февраль 2017
Цена, USD	10
Wi-Fi/Bluetooth	Wi-Fi 802.11n и Bluetooth 4.1 (Bluetooth Classic и LE), обеспечиваемые микросхемой Cypress CYW43438
Видеовход	1 × CSI для подключения камеры по интерфейсу MIPI
Видеовыход	1 × HDMI через разъем mini HDMI (1080p 60) 1 × композитное видео через два контакта на плате (помечены как TV)
Аудиовход	Через I²S
Аудиовыход	HDMI
USB-порты	1 порт Micro-USB 2.0 напрямую от BCM2835
Периферия	40 портов ввода/вывода общего назначения (GPIO), UART (Serial), I²S, I²C/TWI, SPI с селектором между двумя устройствами; пины питания: 3,3 В, 5 В и «земля»
Питание	5 В, 2 А через порт micro-USB или GPIO
Энергопотребление	100 мА (0,5 Вт) в среднем (в режиме ожидания), 350 мА (1,75 Вт) максимум в условиях стресса (монитор, клавиатура и мышь подключены)
Размеры, мм	67,6×30,0x5,0
Вес, г	9
ОС	Raspbian, Ubuntu, Debian, Fedora, Arch Linux, Gentoo, RISC OS, Android, Firefox OS, NetBSD, FreeBSD, Slackware, Tiny Core Linux

ГЛАВА 3



Организация связи для устройств Интернета вещей

Чтобы устройство стало устройством Интернета вещей, необходимо подключить его к сети Интернет. В этой главе мы рассмотрим варианты такого подключения.

3.1. Подключение к Интернету платы Arduino Uno

Подключение компьютеров к сети Интернет, как правило, осуществляется через роутер. Точно так же предоставить доступ в Интернет через роутер можно и плате Arduino Uno. Однако напрямую плату Arduino Uno подключить к роутеру нельзя. Для этого необходимы дополнительные платы: либо шилд (плата расширения) Ethernet Shield, либо какой-либо вариант модуля ESP8266. Подключение платы Arduino Uno к роутеру при этом осуществляется следующим образом:

- по сетевому кабелю через один из входов роутера — с помощью шилда Ethernet Shield;
- по сети Wi-Fi — с помощью модуля ESP8266.

3.1.1. Подключение к Интернету по сетевому кабелю

Для подключения Arduino Uno к Интернету по сетевому кабелю мы воспользуемся платой расширения Ethernet Shield (рис. 3.1). Эта плата, прежде всего, позволяет подключить плату Arduino к сети Ethernet — плата Arduino становится полноправным сетевым устройством и может общаться со всеми устройствами сети, к которой она подключена: компьютерами, сетевыми принтерами, роутерами и др. А через роутер она получает доступ к Интернету.

Итак, сначала соединяем плату Arduino с платой расширения Ethernet Shield. Обращаем при этом внимание на нумерацию цифровых выводов на платах — они должны совпадать. Затем подключаем плату расширения Ethernet Shield к сети, для чего один конец сетевого кабеля подсоединяем к сетевому входу Ethernet Shield, а второй его конец — к свободному сетевому входу роутера (рис. 3.2).

Однако простого электрического соединения устройств недостаточно. Необходимо создать программу для получения платой Arduino сетевого IP-адреса, обеспечивающего ее подключение к сети.

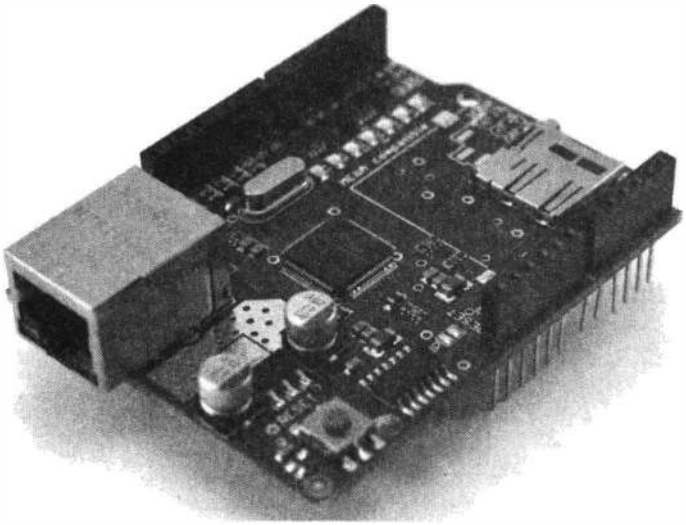


Рис. 3.1. Плата расширения Ethernet Shield W5100

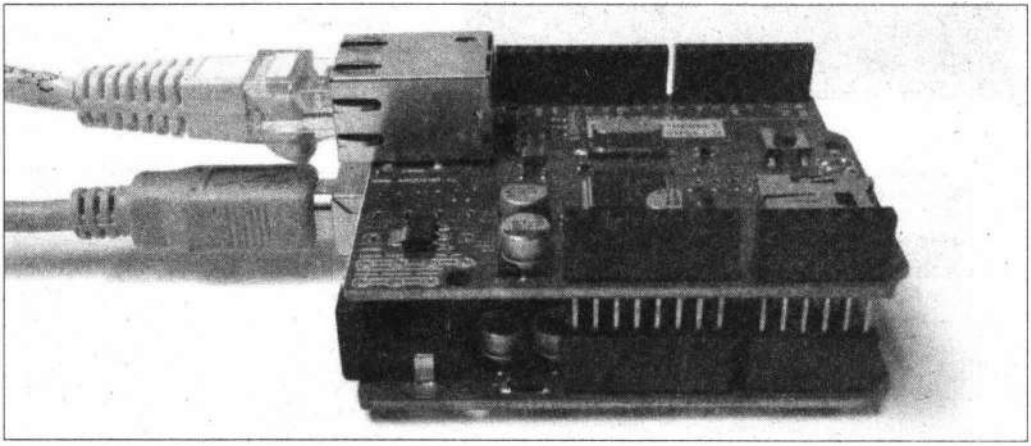


Рис. 3.2. Подключение Ethernet Shield W5100 к плате Arduino и к сети

Что такое IP-адрес?

Каждое устройство локальной сети должно иметь в этой сети свой уникальный адрес. Его можно назначить вручную или получить автоматически от устройства (обычно это роутер), на котором выполняется специальная программа — DHCP-сервер. С помощью написанной нами программы плата Arduino может при подключении к сети обратиться к ее DHCP-серверу и получить от него свободный уникальный IP-адрес.

Для создания такой программы подсоединяем плату Arduino к компьютеру с помощью USB-кабеля и запускаем среду Arduino IDE. В меню **Инструменты | Плата** выбираем плату Arduino Uno (рис. 3.3). В меню **Инструменты | Порт** выбираем порт подключения платы Arduino (рис. 3.4) и заносим в поле кода среды Arduino IDE скетч из листинга 3.1.

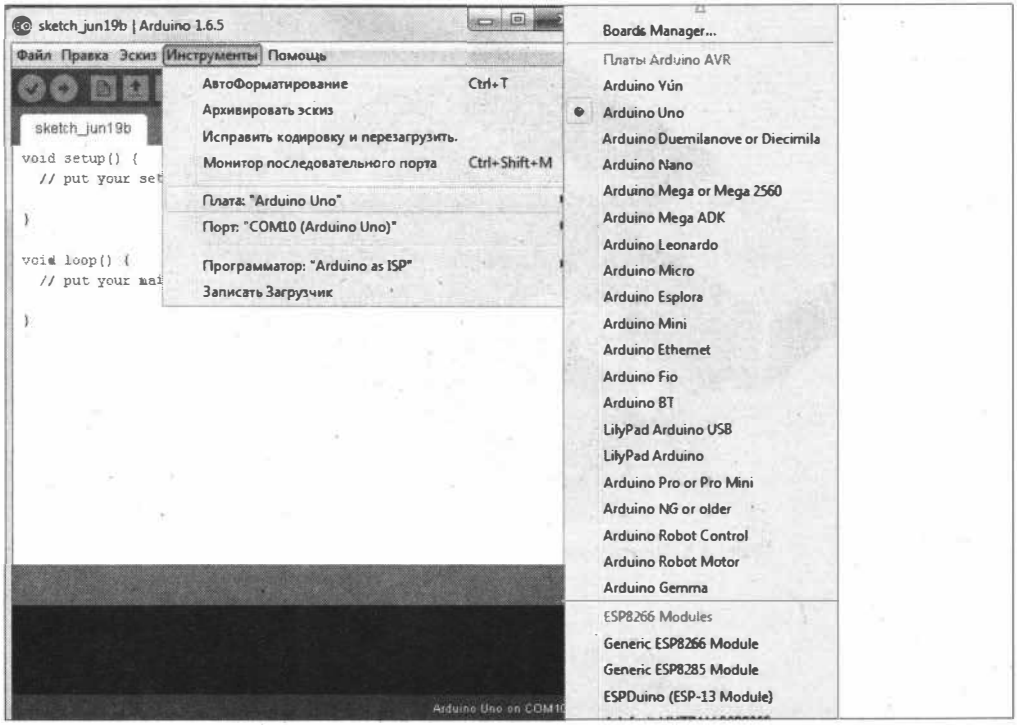


Рис. 3.3. Выбор платы Arduino Uno в среде Arduino IDE

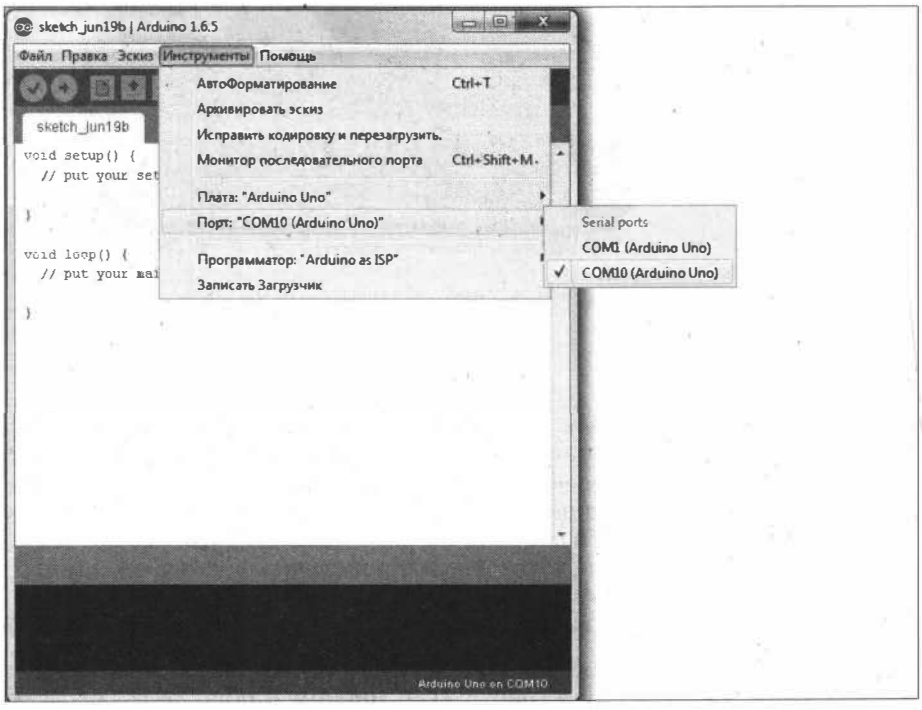


Рис. 3.4. Выбор порта подключения платы Arduino Uno в Arduino IDE

Листинг 3.1

```
// Получение IP-адреса по DHCP
// MAC-адрес Ethernet Shield (можно увидеть на наклейке на плате)
// или произвольный уникальный в сети
#include <Ethernet.h>
#include <SPI.h>

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};

void setup() {
  // запуск последовательного порта:
  Serial.begin(9600);
  // запуск Ethernet-соединения
  if (Ethernet.begin(mac) == 0) {
    Serial.println("Failed to configure Ethernet using DHCP");
    for (;;)
      ;
  }
  // печать в последовательный порт полученного от DHCP адреса
  Serial.print("My IP address: ");
  for (byte thisByte = 0; thisByte < 4; thisByte++) {
    Serial.print(Ethernet.localIP()[thisByte], DEC);
    Serial.print(".");
  }
  Serial.println();
}

void loop() {;}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 3.1, можно найти в папке *examples\03\03_01* сопровождающего книгу электронного архива (см. *приложение*).

В этом скетче задействуется библиотека Ethernet (`Ethernet.h`), входящая в состав Arduino IDE.

В строке:

```
byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
```

вводим MAC-адрес своей платы Ethernet Shield (его можно увидеть на наклейке на плате), в случае отсутствия такой наклейки оставляем строку без изменения или вносим произвольные данные.

Теперь сохраняем скетч на компьютере и загружаем его в плату Arduino, нажав на кнопку загрузки, — скетч будет скомпилирован и загружен в плату.

Открываем монитор последовательного порта — в нем будут отображены данные об IP-адресе, полученном платой Arduino от DHCP-сервера (рис. 3.5).

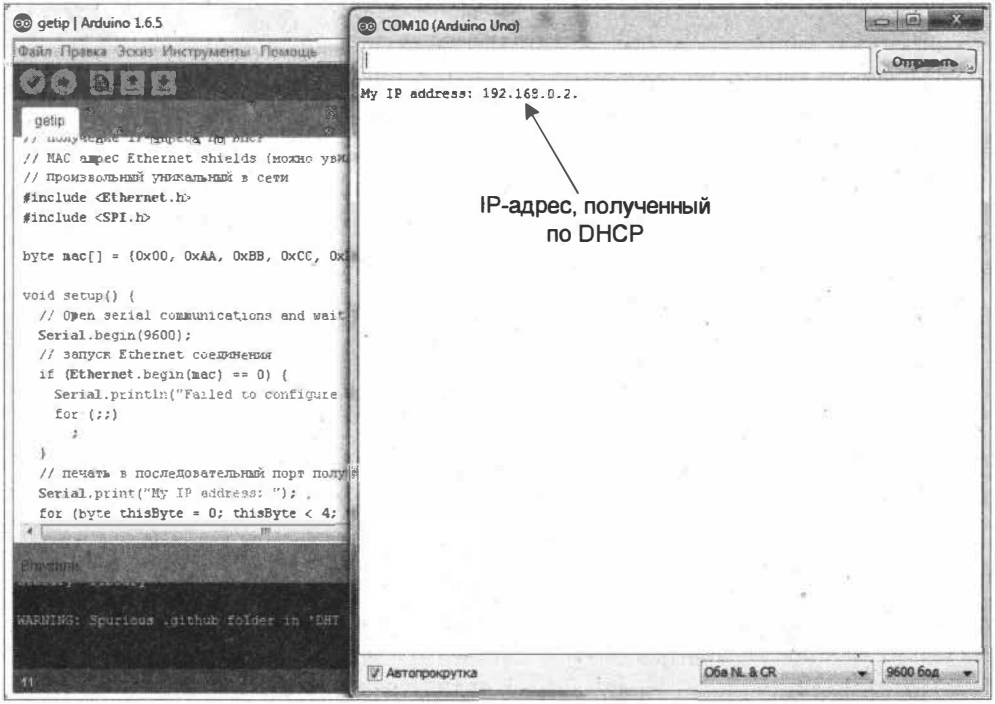


Рис. 3.5. Вывод в последовательный порт IP-адреса, полученного от DHCP-сервера (на роутере)

Мы можем убедиться, что плата Arduino подключена к сети и имеет связь с компьютером по локальной сети. Для этого на компьютере запустим командную строку (**Пуск | Все программы | Стандартные | Командная строка**) и наберем в ней команду:

```
ping <ip-адрес> -t
```

где `<ip-адрес>` — IP-адрес, полученный платой Arduino.

В окно командной строки будут приходить ответы от платы Arduino (рис. 3.6).

IP-адрес можно назначить и вручную (так называемый *статический IP-адрес*). Для этого надо знать следующие параметры локальной сети:

- IP-адрес шлюза;
- маску подсети;
- адрес DNS-сервера (сервер, который преобразует IP-адреса в названия сайтов).

Зная эти параметры, назначим плате Arduino IP-адрес самостоятельно. Для этого создадим в Arduino IDE новый скетч и занесем в поле кода скетч из листинга 3.2.

Листинг 3.2

```
// Назначение статического IP-адреса
// MAC-адрес Ethernet Shield (можно увидеть на наклейке на плате)
// или произвольный уникальный в сети
```

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

byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
// IP-адрес, назначаемый Ethernet Shield:
byte ip[] = { 192, 168, 0, 111 };
// IP-адрес DNS-сервера:
byte sdns[] = { 192, 168, 1, 1 };
// адрес шлюза:
byte gateway[] = { 192, 168, 0, 1 };
// маска:
byte subnet[] = { 255, 255, 255, 0 };

void setup() {
  // запуск последовательного порта:
  Serial.begin(9600);
  // запуск Ethernet-соединения
  Ethernet.begin(mac, ip, sdns, gateway, subnet);
  delay(1000);
  Serial.println(Ethernet.localIP());
}
void loop() {;}
```

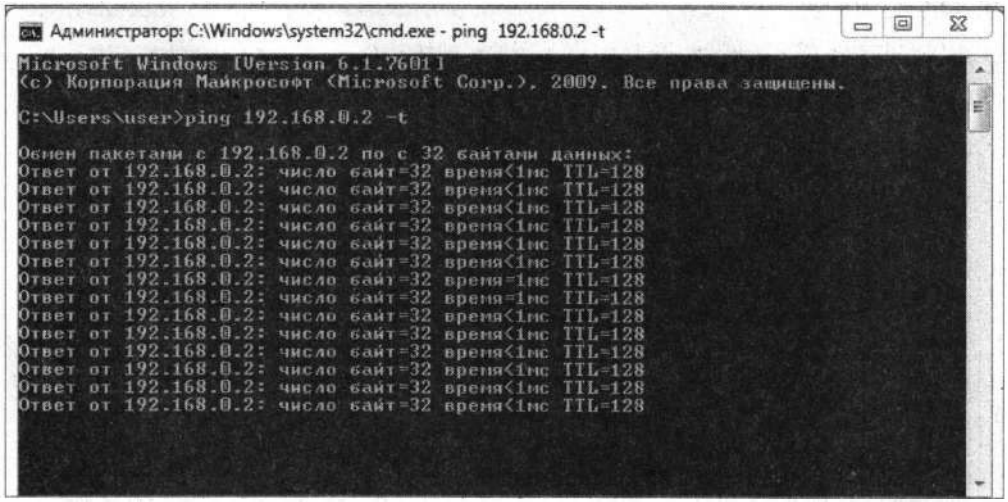


Рис. 3.6. Проверка связи по сети компьютера с платой Arduino Uno

Электронный архив

Скетч, соответствующий листингу 3.2, можно найти в папке *examples\03\03_02* сопровождающего книгу электронного архива (см. *приложение*).

Загружаем скетч в плату Arduino и открываем монитор последовательного порта — в нем будут отображены данные об IP-адресе (рис. 3.7).

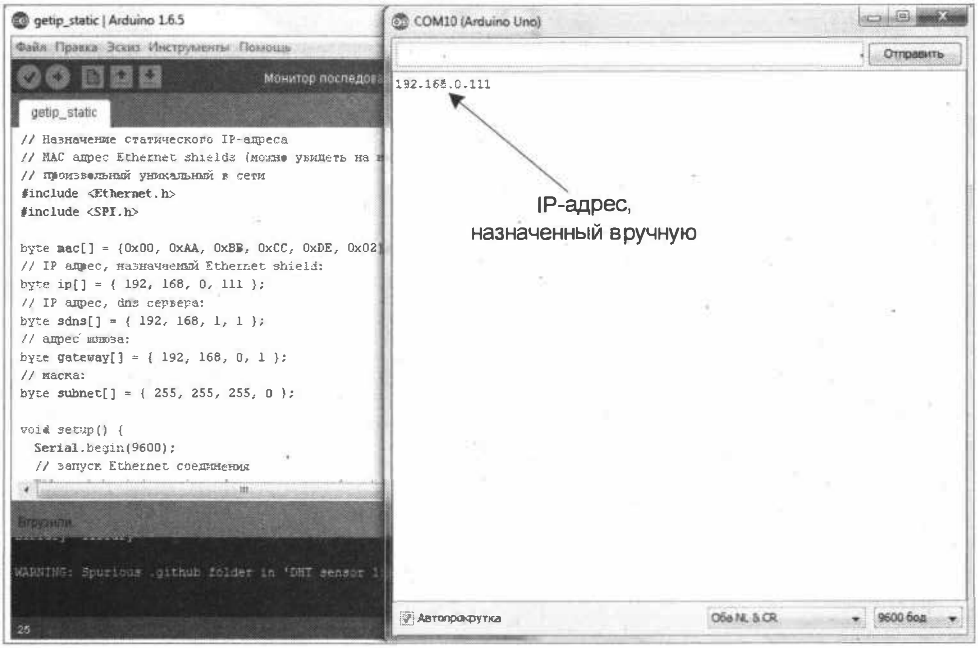


Рис. 3.7. Вывод в последовательный порт статического IP-адреса, назначенного вручную

Проверим, что есть связь по сети с платой Arduino, набрав на компьютере в командной строке:

```
ping <ip-адрес> -t
```

В окно командной строки должны приходить ответы от платы Arduino (рис. 3.8).

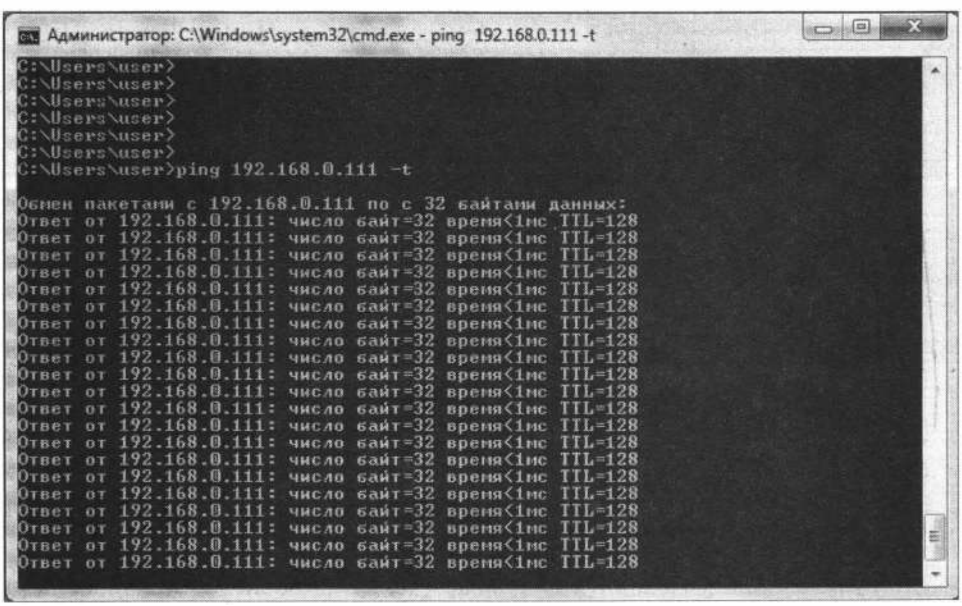


Рис. 3.8. Проверка связи по сети компьютера с платой Arduino Uno

Теперь плата Arduino Uno подключена к локальной сети, а через роутер — к сети Интернет. В следующем разделе мы рассмотрим второй вариант подключения платы Arduino Uno к роутеру — подключение по Wi-Fi.

3.1.2. Подключение к Интернету по Wi-Fi

Если нет возможности или желания тянуть провода от Arduino Uno к роутеру и роутер поддерживает подключение по Wi-Fi, для подключения платы Arduino Uno к роутеру можно воспользоваться модулем ESP8266. Существует огромное количество разновидностей модуля ESP8266. Наиболее бюджетным вариантом из них является ESP-01 (рис. 3.9).

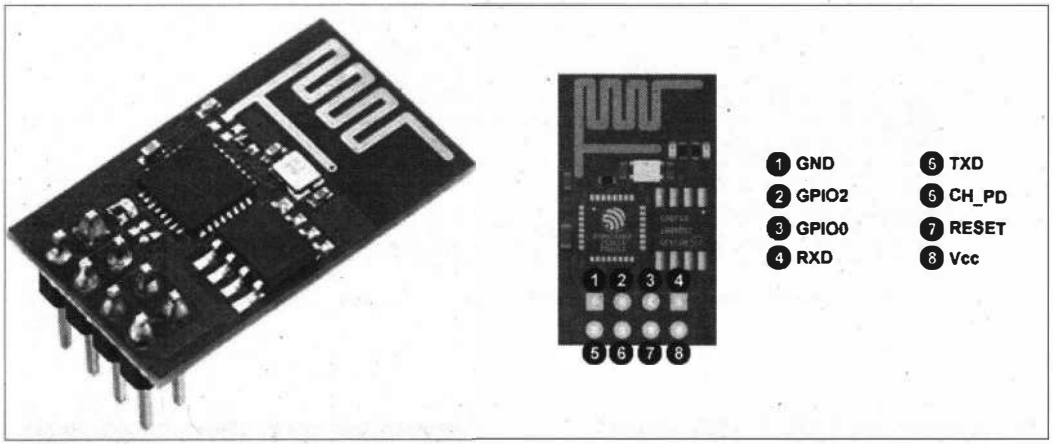


Рис. 3.9. Плата ESP8266 ESP-01

Управление модулями ESP8266 осуществляется отправкой им AT-команд. Подключим модуль ESP-01 к компьютеру через переходник USB-Serial по схеме, показанной на рис. 3.10, и запустим на компьютере программу ESP8266_Config.exe (рис. 3.11).

> ***Внимание!***
>
> Питание модуля ESP-01 подключаем к выводу 3,3 В платы USB-Serial.

> ***Электронный архив***
>
> Программу *ESP8266_Config.exe* можно найти в папке *prgs\ESP8266_Config.exe* сопровождающего книгу электронного архива (см. *приложение*).

Выбираем в программе порт подключения (поле **Port**) и скорость последовательного порта (поле **Baudrate**) и нажимаем на кнопку **Connect**. Нажимаем на кнопку **Serial Monitor**. Если картинка на мониторе компьютера отличается от приведенной на рис. 3.11, пробуем отключить и заново подключить провод к выводу **CH_PD**. Если это не поможет — изменяем скорость последовательного порта и заново устанавливаем соединение.

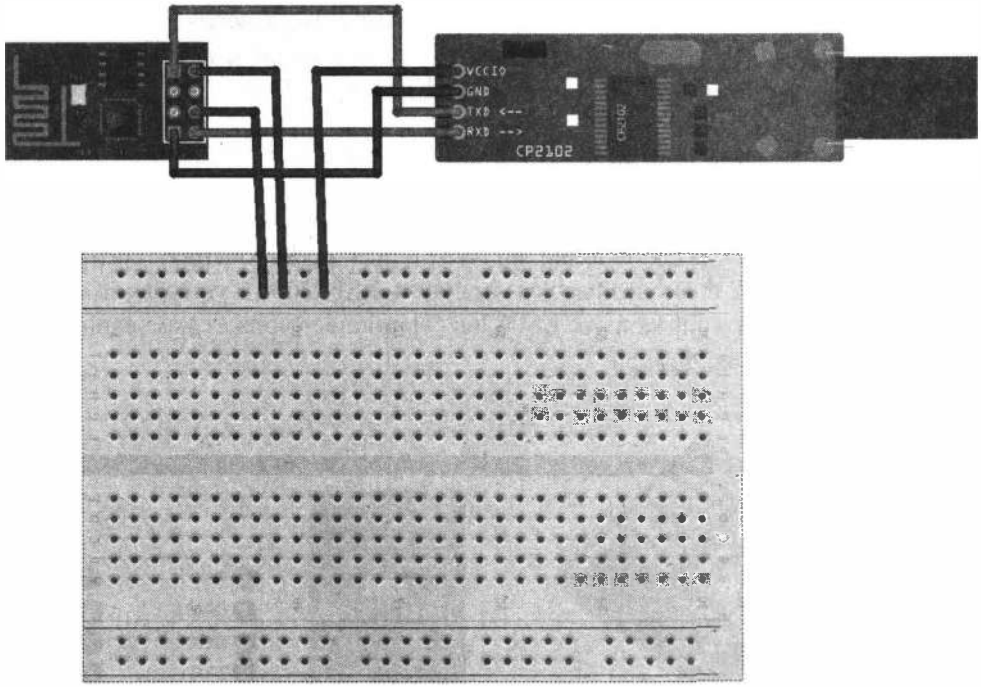


Рис. 3.10. Подключение платы ESP8266 ESP-01 к переходнику USB-Serial

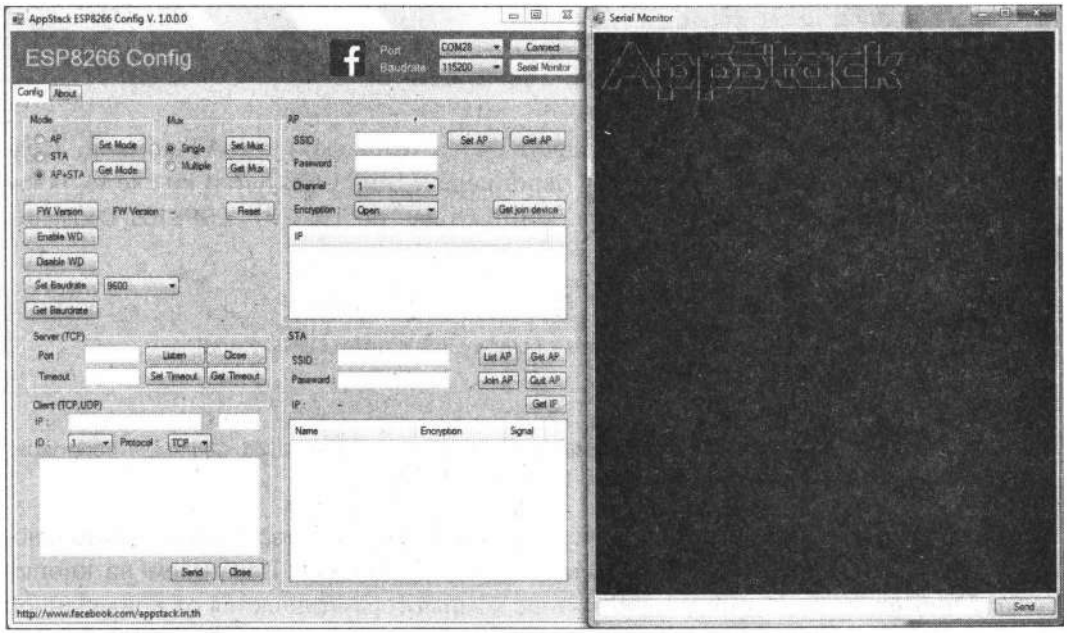


Рис. 3.11. Подключение к плате ESP8266 ESP-01 в программе ESP8266 Config

Теперь пробуем отправлять в последовательный порт AT-команды (рис. 3.12). Тестовая команда:

```
AT
```

Ответ должен быть: **OK**.

Команда

```
AT+CWMODE?
```

вернет режим работы модуля.

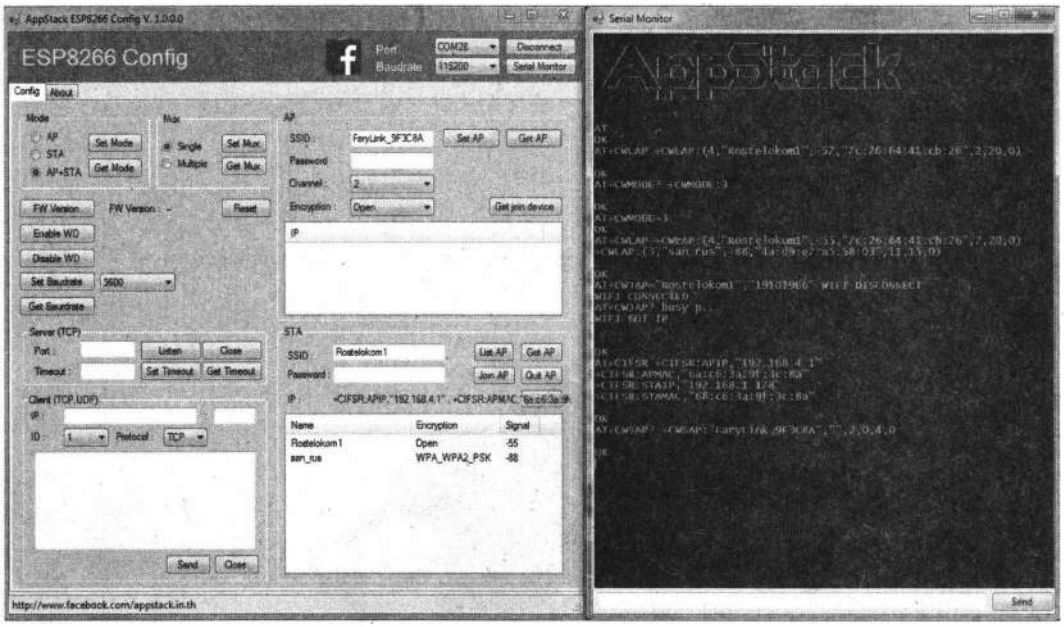


Рис. 3.12. Отправка AT-команд на модуль ESP8266 ESP-01

Команда

```
AT+CWMODE=3
```

установит режим работы модуля 3 (модуль в режиме точки доступа Wi-Fi и клиента Wi-Fi). Если модуль находится в режиме точки доступа Wi-Fi, к нему можно подключиться по Wi-Fi, например, с телефона или планшета (рис. 3.13).

Поиск доступных беспроводных сетей:

```
AT+CWLAP
```

Подключение к беспроводной сети:

```
AT+CWJAP="<имя сети>","<пароль>"
```

Теперь пришло время подключить модуль ESP-01 к плате Arduino Uno. Питание модуля — 3,3 В, поэтому брать питание с вывода +5 В Arduino Uno нельзя. От контакта 3,3 В взять питание не получится, т. к. вывод 3,3 В платы Arduino Uno может

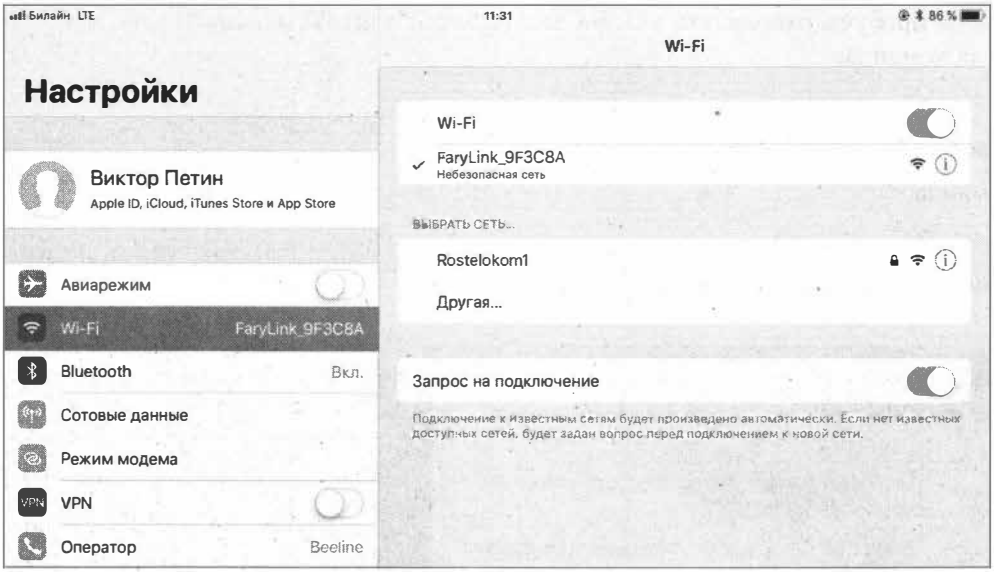


Рис. 3.13. Подключение к плате ESP8266 ESP-01 с телефона (планшета)

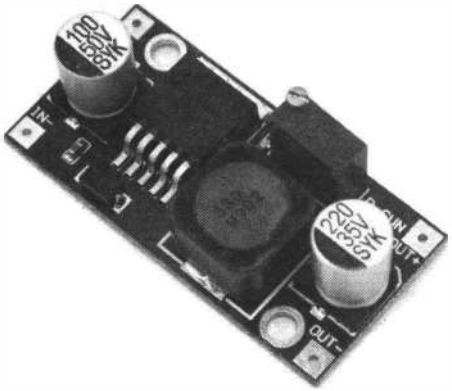


Рис. 3.14. Преобразователь напряжения на микросхеме LM2596

выдать ток только 50 мА, а модуль ESP-01 потребляет ток более 200 мА. Поэтому мы воспользуемся преобразователем напряжения на микросхеме LM2596 (рис. 3.14).

Напряжение подстраиваем с помощью находящегося на плате потенциометра и проверяем мультиметром или вольтметром, а затем подключаем модуль ESP-01 к плате Arduino по схеме, показанной на рис. 3.15.

Связь ESP-01 и Arduino осуществляется по последовательному порту. У платы Arduino имеется один физический последовательный порт на контактах 0 (Rx) и 1 (Tx). Этот последовательный порт мы используем для отправки AT-команд с компьютера на плату Arduino и для вывода отладочной информации. А для связи ESP-01 и Arduino мы воспользуемся программным последовательным портом на двух любых контактах Arduino, для чего применим библиотеку SoftwareSerial, входящую в состав Arduino IDE.

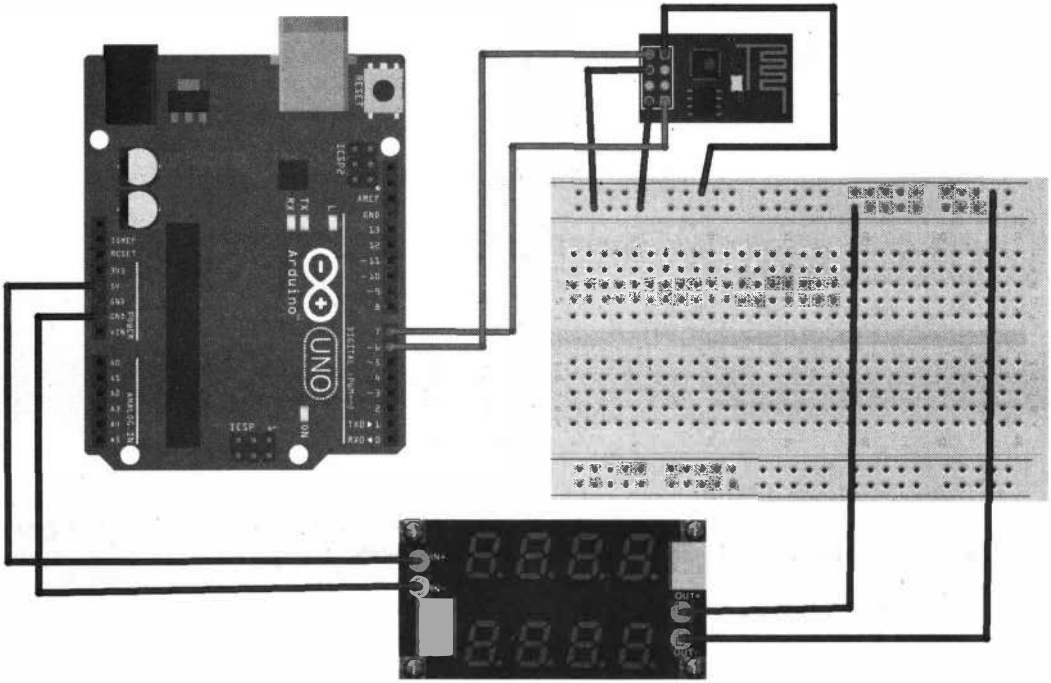


Рис. 3.15. Подключение платы ESP-01 к плате Arduino Uno

Итак, создадим в Arduino IDE новый скетч с именем arduino_esp8266.ino и занесем в поле кода содержимое листинга 3.3.

Листинг 3.3

```
// подключение библиотеки SoftwareSerial
#include <SoftwareSerial.h>
// константы для Software Serial
#define PIN_TX 7
#define PIN_RX 6
// пин Arduino для подключения CH_PD
#define PIN_CH_PD 5
// создание экземпляра SoftwareSerial
//  6 --> Tx (ESP8266)
//  7 --> Rx (ESP8266)
SoftwareSerial ESP8266Serial(PIN_RX, PIN_TX);

void setup() {
  Serial.begin(9600);
  ESP8266Serial.begin(115200);
  pinMode(PIN_CH_PD,OUTPUT);
  // сигнал LOW на CH_PD
  digitalWrite(PIN_CH_PD,LOW);
  delay(100);
```

```
  // сигнал HIGH на CH_PD
  digitalWrite(PIN_CH_PD,HIGH);
  delay(1000);
}
void loop() {
  // получение Serial --> отправка ESP8266Serial
  if (Serial.available())
      ESP8266Serial.write(Serial.read());
  // получение ESP8266Serial --> отправка Serial
  if (ESP8266Serial.available())
      Serial.write(ESP8266Serial.read());
  delay(10);
}
```

Электронный архив

Скетч, соответствующий листингу 3.3, можно найти в папке *examples\03\03_03* сопровождающего книгу электронного архива (см. *приложение*).

Но это еще не весь код. Создадим дополнительную вкладку **ATcommands** — при этом в папке arduino_esp8266 будет создан файл ATcommands.ino (рис. 3.16) и занесем в поле ввода вкладки **ATcommands** содержимое листинга 3.4.

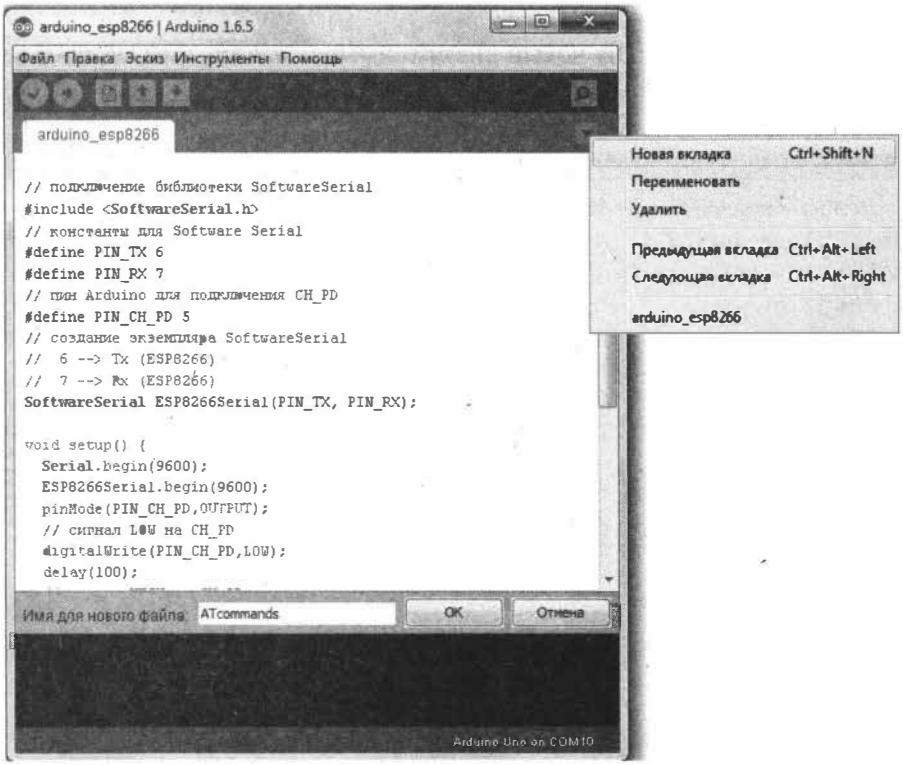


Рис. 3.16. Создание дополнительной вкладки в скетче

Пояснение

Дополнительные файлы создаются, чтобы уменьшить размер кода в основном файле *arduino_esp8266.ino*. В них записывается код функций и процедур, которые вызываются на выполнение из файла *arduino_esp8266.ino*.

Листинг 3.4

```
// отправка AT-команд
int8_t sendATcommand(char* ATcommand, char* expected_answer, unsigned int
timeout)
  {
   uint8_t x=0,  answer=0;
   char response[150];
   unsigned long previous;

   memset(response, '\0', 150);    // Initialize the string
   delay(100);
   // Очистить буфер входящих данных
   while(ESP8266Serial.available() > 0)
      ESP8266Serial.read();
   // Отправка AT-команды
   ESP8266Serial.println(ATcommand);
   x = 0;
   previous = millis();
   // ждать необходимый ответ answer или лимит времени
   do{
     if(ESP8266Serial.available() != 0)
       {
       // получать данные из ESP8266
       response[x] = ESP8266Serial.read();
       //Serial.print(response[x]);
       x++;
       // сравнение полученных данных с необходимым ответом
       if (strstr(response, expected_answer) != NULL)
         {
         answer = 1;
         }
       }
    }
    while((answer == 0) && ((millis() - previous) < timeout));
    Serial.println(response);

    return answer;
  }]
```

Электронный архив

Скетч, соответствующий листингу 3.4 (файл *ATcommands.ino*), можно найти в папке *examples\03\03_04* сопровождающего книгу электронного архива (см. *приложение*).

Функция `sendATcommand()` **в качестве входных аргументов получает АТ-команду, необходимый ответ на правильное выполнение команды и время ожидания ответа в мсек.**

Сначала функция очищает буфер входящих данных и считывает все пришедшие данные из последовательного порта:

```
// Очистить буфер входящих данных
while(ESP8266Serial.available() > 0)
   ESP8266Serial.read();
```

Затем отправляет по последовательному порту на ESP-01 АТ-команду:

```
// Отправка АТ команды
ESP8266Serial.println(ATcommand);
```

После чего приступает к ожиданию и чтению данных из ESP-01 по последовательному порту, постоянно проверяя время ожидания ответа. Получая побайтно данные из ESP-01, функция сравнивает их с необходимым ответом и при совпадении возвращает значение 1. При превышении времени — возвращает значение 0.

```
do {
  if(ESP8266Serial.available() != 0)
    {
    // получать данные из ESP8266
    response[x] = ESP8266Serial.read();
    //Serial.print(response[x]);
    x++;
    // сравнение полученных данных с необходимым ответом
    if (strstr(response, expected_answer) != NULL)
      {
      answer = 1;
      }
    }
 }
 while((answer == 0) && ((millis() - previous) < timeout));
```

Теперь загружаем скетч в плату Arduino, открываем монитор последовательного порта и отправляем АТ-команды на плату Arduino — ответы от ESP-01 также печатаются в монитор последовательного порта (рис. 3.17).

Если команды отправляются и ответ от ESP-01 приходит, пришло время написать скетч для подключения платы Arduino к роутеру по Wi-Fi отправкой АТ-команд.

Создадим в Arduino IDE новый скетч `arduino_esp8266_connectWiFi.ino` **и занесем в поле кода содержимое листинга 3.5.**

Листинг 3.5

```
// подключение библиотеки SoftwareSerial
#include <SoftwareSerial.h>
```

```
// константы для Software Serial
#define PIN_TX 7
#define PIN_RX 6
// пин Arduino для подключения CH_PD
#define PIN_CH_PD 5
// создание экземпляра SoftwareSerial
//  6 --> Tx (ESP8266)
//  7 --> Rx (ESP8266)
SoftwareSerial ESP8266Serial(PIN_RX, PIN_TX);

void setup() {
  Serial.begin(9600);
  ESP8266Serial.begin(115200);
  pinMode(PIN_CH_PD,OUTPUT);
  // сигнал LOW на CH_PD
  digitalWrite(PIN_CH_PD,LOW);
  delay(100);
  // сигнал HIGH на CH_PD
  digitalWrite(PIN_CH_PD,HIGH);
  delay(1000);
  // Отправка AT-команд
  // проверка - команда AT
  while(sendATcommand("AT","OK",1000)<1) {
     Serial.print("Connect ESP8266 - error");
     delay(1000);
  }
  Serial.print("Connect ESP8266 - OK");
  delay(1000);
  //
  while(sendATcommand("AT+CWJAP=\"Rostelokom1\",\"19101966\"","OK",6000)<1) {
     Serial.print("Connect WiFi - error");
     delay(1000);
  }
  Serial.print("Connect WiFi - OK");
  delay(1000);

}

void loop() {
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 3.5, можно найти в папке *examples\03\03_05* сопровождающего книгу электронного архива (см. *приложение*).

Загружаем этот скетч в плату Arduino, открываем монитор последовательного порта и наблюдаем процесс подключения ESP-01 к сети Wi-Fi (рис. 3.18).

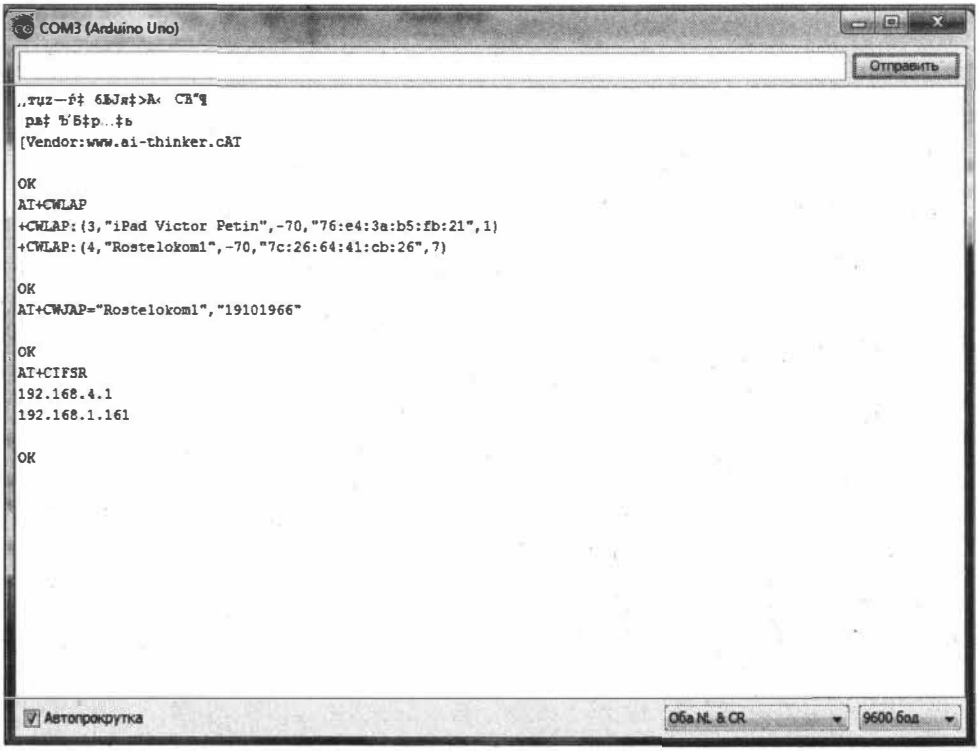


Рис. 3.17. Отправка AT-команд на ESP-01 и получение ответов

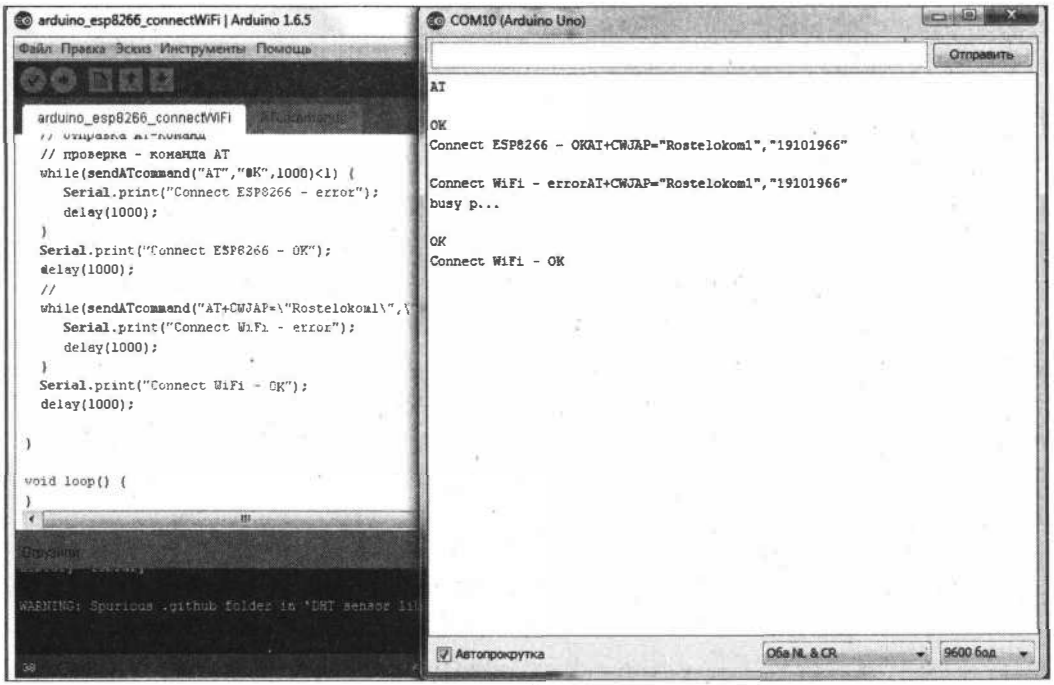


Рис. 3.18. Процесс подключения платы Arduino Uno к роутеру по Wi-Fi

Как можно видеть, плата Arduino Uno подключена по Wi-Fi к роутеру, а через роутер — к сети Интернет.

Вам необходимо внести в скетч следующее изменение — в строке

```
while(sendATcommand("AT+CWJAP=\"Rostelekom1\",\"19101966\"","OK",6000)<1)
```

заменить названия точки доступа и пароль (выделены полужирным) на свои.

3.2. Подключение по Wi-Fi плат Arduino MKR и Nano 33 IoT

В этом разделе приведены практические примеры подключения по Wi-Fi новых плат Arduino: MKR и Nano 33.

3.2.1. Подключение по Wi-Fi платы Arduino MKR1000 WiFi

С платой Arduino MKR1000 WiFi мы познакомились в *разд. 2.2.1*. Здесь же мы рассмотрим программирование этой платы в среде Arduino IDE.

По умолчанию среда Arduino IDE настроена только на AVR-платы. Для работы с платой Arduino MKR1000 WiFi необходимо добавить в Менеджере плат среды Arduino IDE поддержку платформы Arduino SAMD (32bit ARM Cortex-M0+). Для этого выполним в среде Arduino IDE команду **Инструменты | Плата | Менеджер плат**, найдем **Arduino SAMD** и нажмем на кнопку **Установка** (рис. 3.19). После установки программного обеспечения в меню **Инструменты | Плата** среды Arduino IDE появится поддержка плат Arduino Zero, MKR и Nano, в том числе и платы MKR1000 WiFi (рис. 3.20).

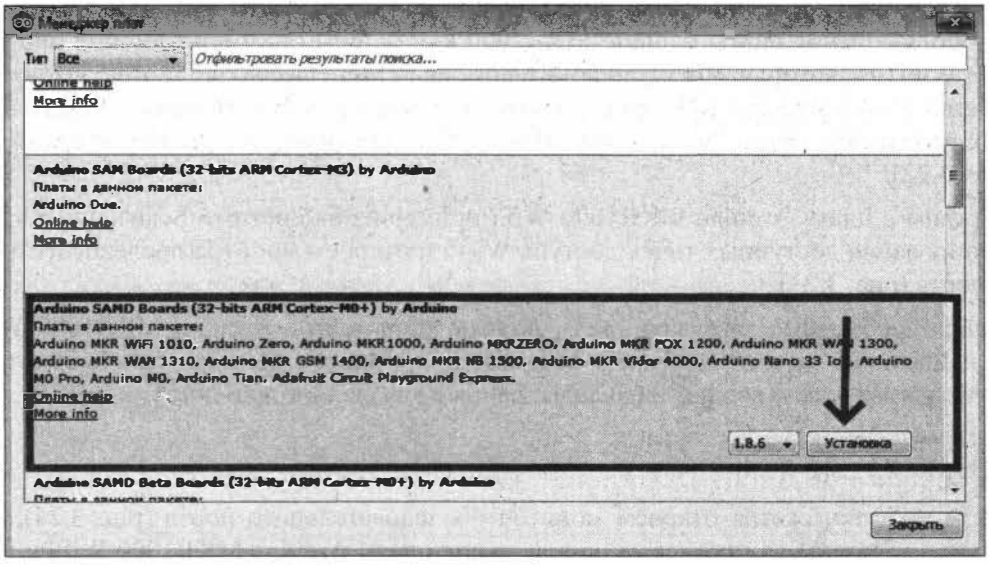


Рис. 3.19. Установка программного обеспечения поддержки плат Arduino MKR

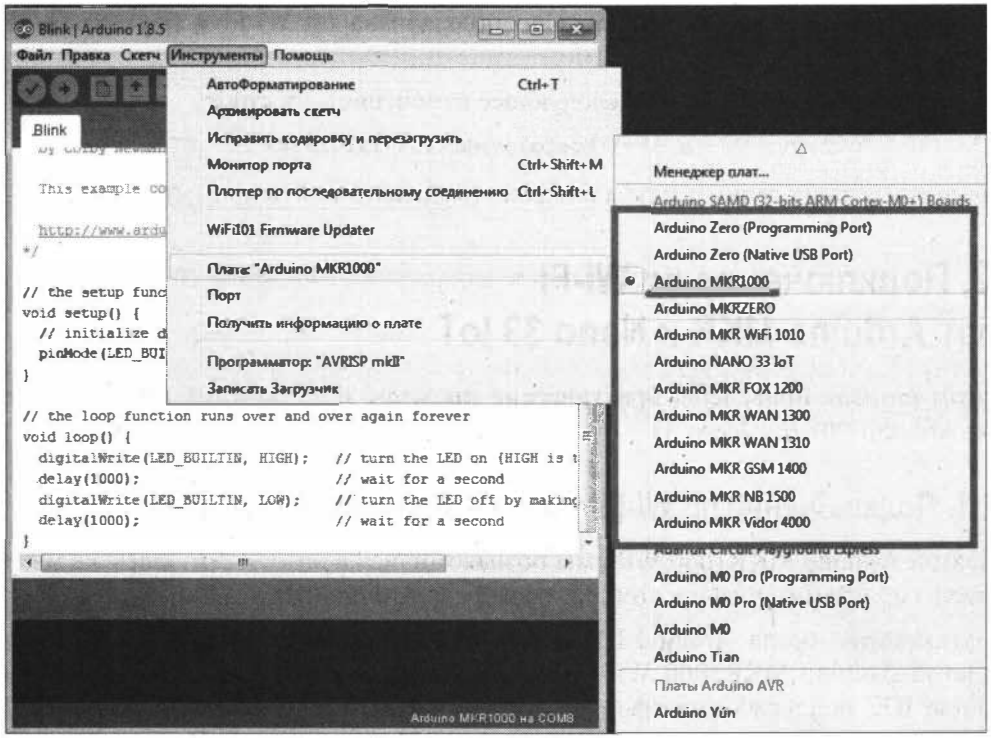


Рис. 3.20. Поддержка плат Arduino MKR в Arduino IDE установлена

Подключим плату MKR1000 WiFi к компьютеру с помощью USB-кабеля. После подключения соответствующий драйвер должен будет установиться автоматически. Для проверки работоспособности запустим Диспетчер устройств и убедимся, что наша плата появилась в разделе **Порты (COM и LPT)** (рис. 3.21).

Для подключения платы Arduino MKR1000 к сети Wi-Fi нам понадобится библиотека WiFi101, которую мы установим с помощью Менеджера библиотек: выбираем в меню среды Arduino IDE пункт **Эскиз | Подключить библиотеку | Управлять библиотеками**, ищем библиотеку **WiFi101** и нажимаем на кнопку **Установка** (рис. 3.22).

Загрузим в плату Arduino MKR1000 WiFi пример из библиотеки ScanNetworks для сканирования доступных точек доступа Wi-Fi и откроем монитор последовательного порта (рис. 3.23).

Теперь необходимо загрузить скетч подключения платы Arduino MKR1000 WiFi к найденной точке доступа. Загружаем в плату пример из библиотеки **connectWithWPA.ino** и вводим на вкладке данные для своей точки доступа Wi-Fi:

```
#define SECRET_SSID "my_DLink"
#define SECRET_PASS "********".
```

После загрузки скетча откроем монитор последовательного порта (рис. 3.24), где можно наблюдать за процессом подключения платы Arduino MKR1000 WiFi к точке доступа.

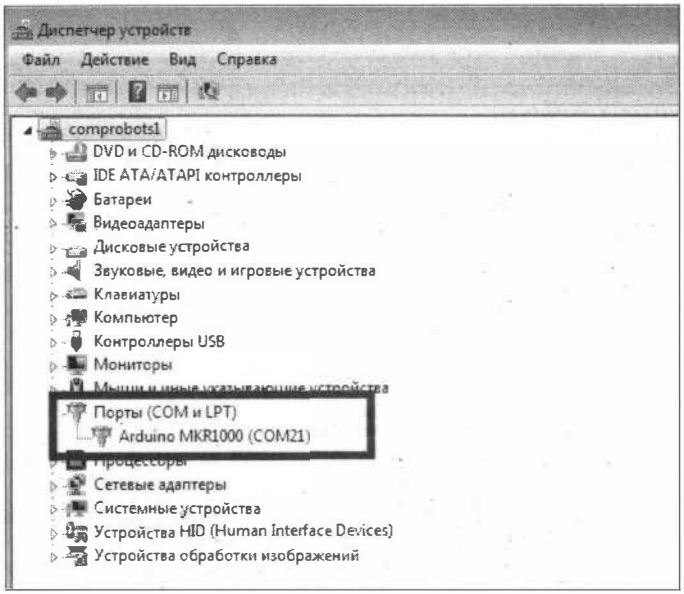


Рис. 3.21. Драйвер платы Arduino MKR1000 установлен

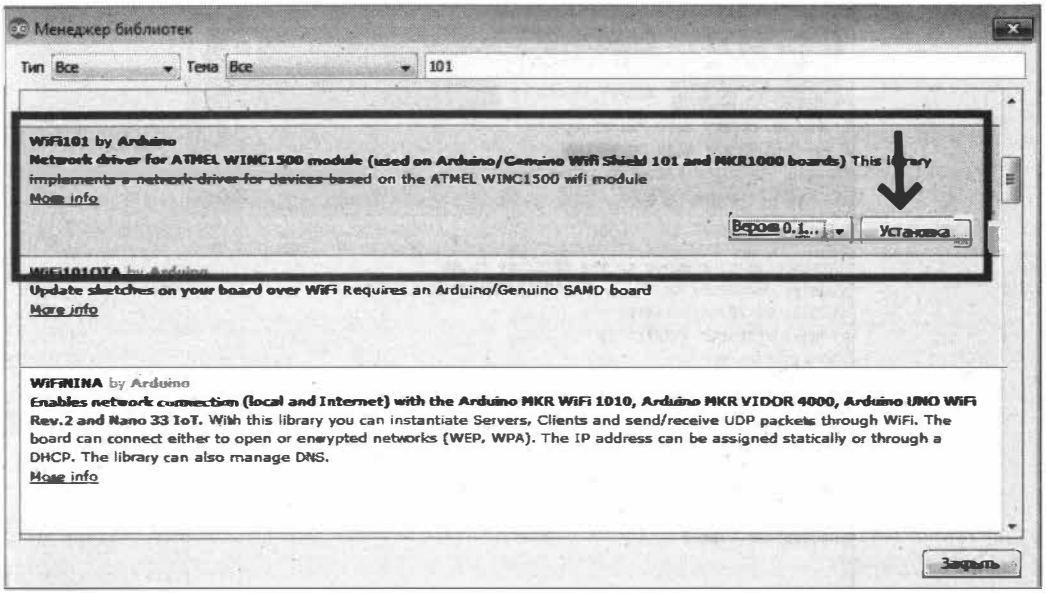


Рис. 3.22. Установка библиотеки WiFi101

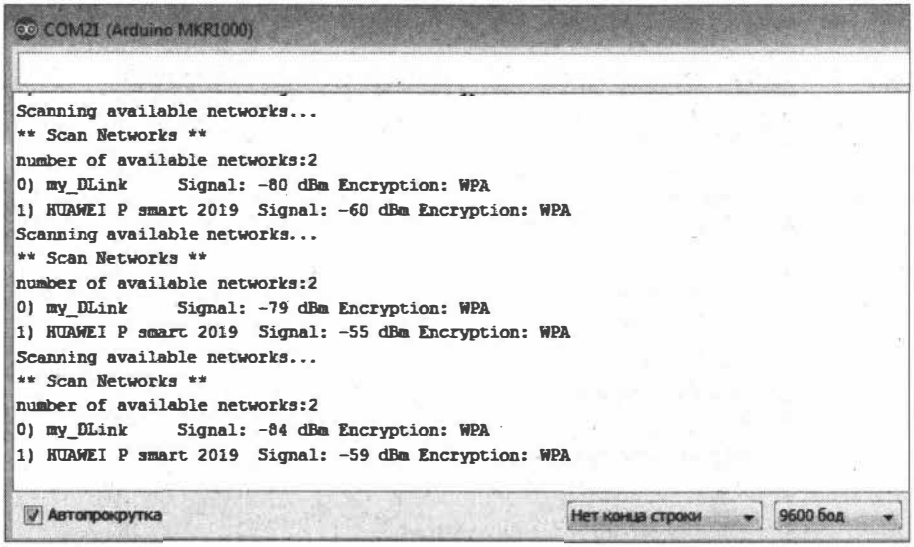


Рис. 3.23. Поиск доступных точек доступа Wi-Fi для платы Arduino MKR1000 WiFi (пример ScanNetworks)

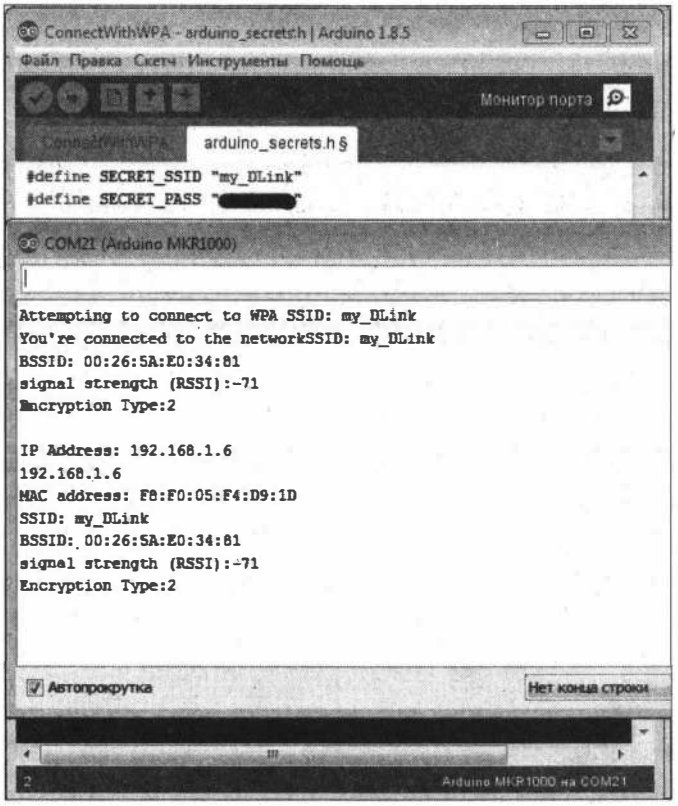


Рис. 3.24. Подключение платы Arduino MKR1000 WiFi к точке доступа

3.2.2. Подключение по Wi-Fi платы Arduino Nano 33 IoT

С платой Arduino Nano 33 IoT мы познакомились в *разд. 2.3.1*. Здесь же мы рассмотрим программирование этой платы в среде Arduino IDE.

По умолчанию среда Arduino IDE настроена только на AVR-платы. Для работы с платой Arduino Nano 33 IoT необходимо добавить в Менеджере плат среды Arduino IDE поддержку платформы Arduino SAMD (32bit ARM Cortex-M0+). Для этого выполним в среде Arduino IDE команду **Инструменты | Плата | Менеджер плат**, найдем **Arduino SAMD** и нажмем на кнопку **Установка** (рис. 3.25). После установки программного обеспечения в меню **Инструменты | Плата** среды Arduino IDE появится поддержка плат Arduino Zero, MKR и Nano, в том числе и платы Arduino Nano 33 IoT (рис. 3.26).

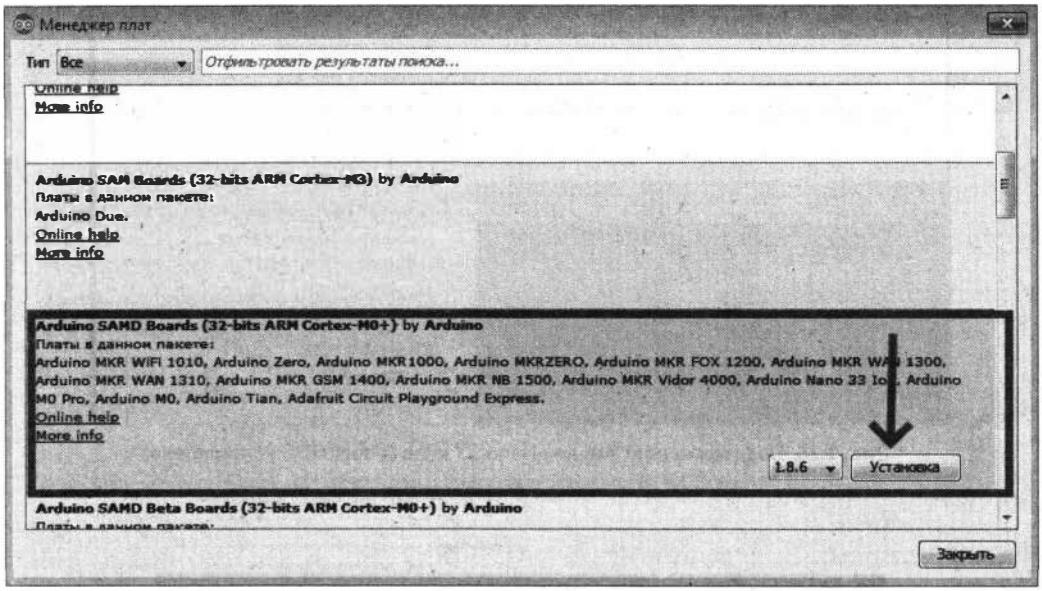


Рис. 3.25. Установка программного обеспечения поддержки плат Arduino Nano 33 IoT

Подключим плату Arduino Nano 33 IoT к компьютеру с помощью USB-кабеля. После подключения соответствующий драйвер должен будет установиться автоматически (рис. 3.27).

Для подключения платы Arduino Nano 33 IoT к сети Wi-Fi нам понадобится библиотека WiFiNINA, которую мы установим с помощью Менеджера библиотек: выбираем в меню среды Arduino IDE пункт **Эскиз | Подключить библиотеку | Управлять библиотеками**, ищем библиотеку **WiFiNINA** и нажимаем на кнопку **Установка** (рис. 3.28).

Загрузим в плату Arduino Nano 33 IoT пример из библиотеки ScanNetworks для сканирования доступных точек доступа Wi-Fi и откроем монитор последовательного порта (рис. 3.29).

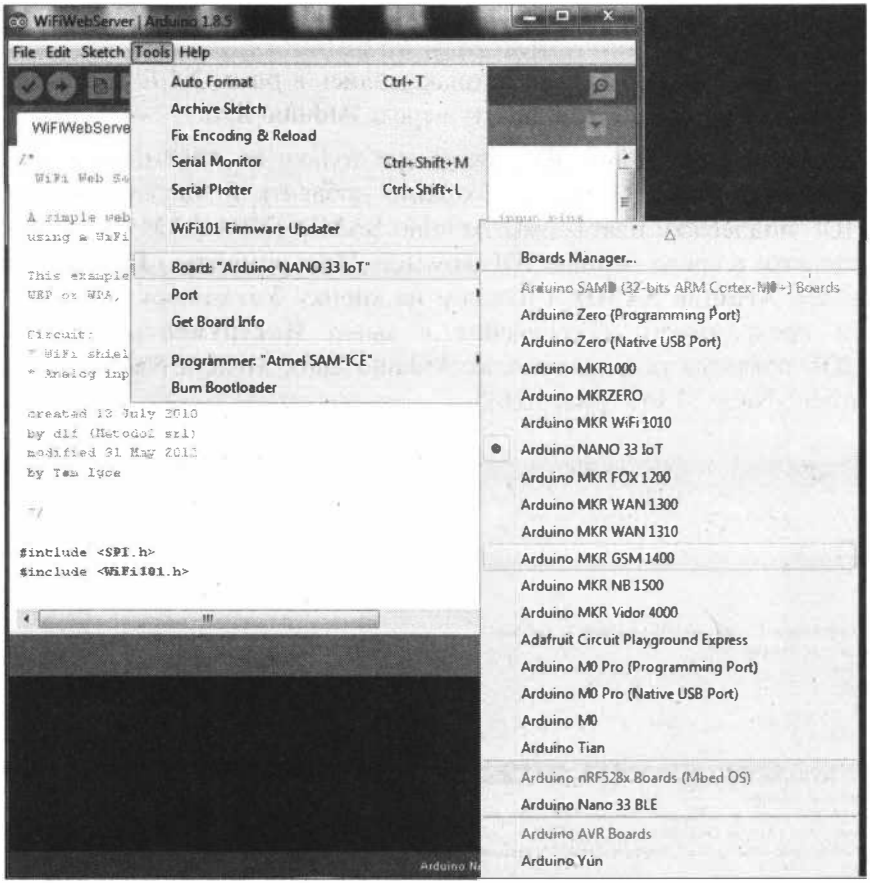


Рис. 3.26. Поддержка плат Arduino Nano 33 IoT в Arduino IDE установлена

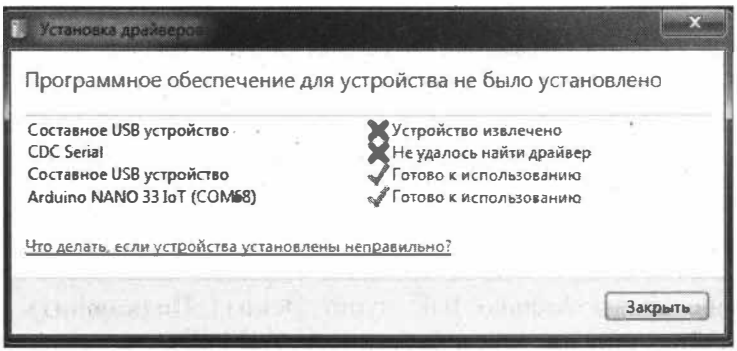


Рис. 3.27. Драйвер платы Arduino Nano 33 IoT установлен

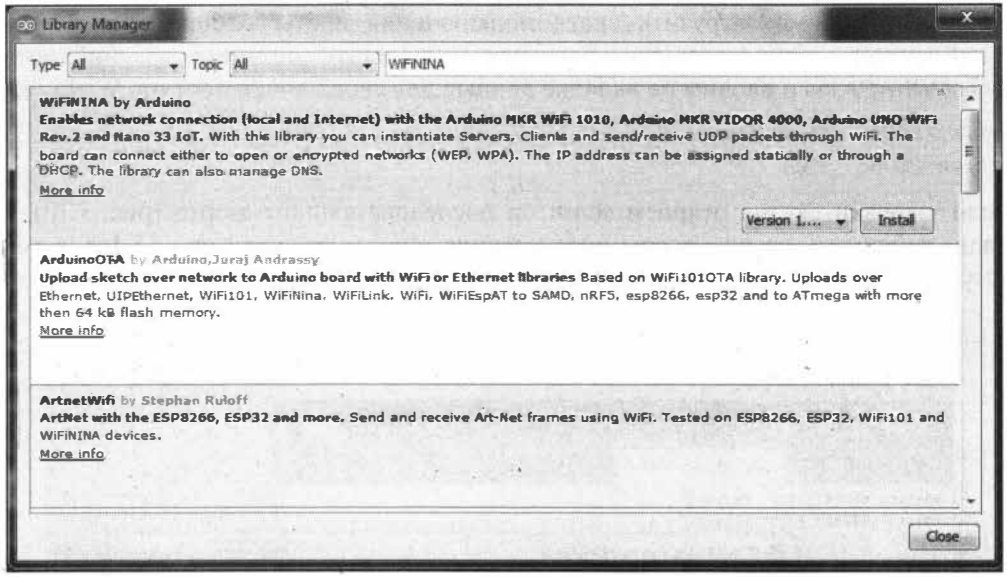


Рис. 3.28. Установка библиотеки WiFiNINA

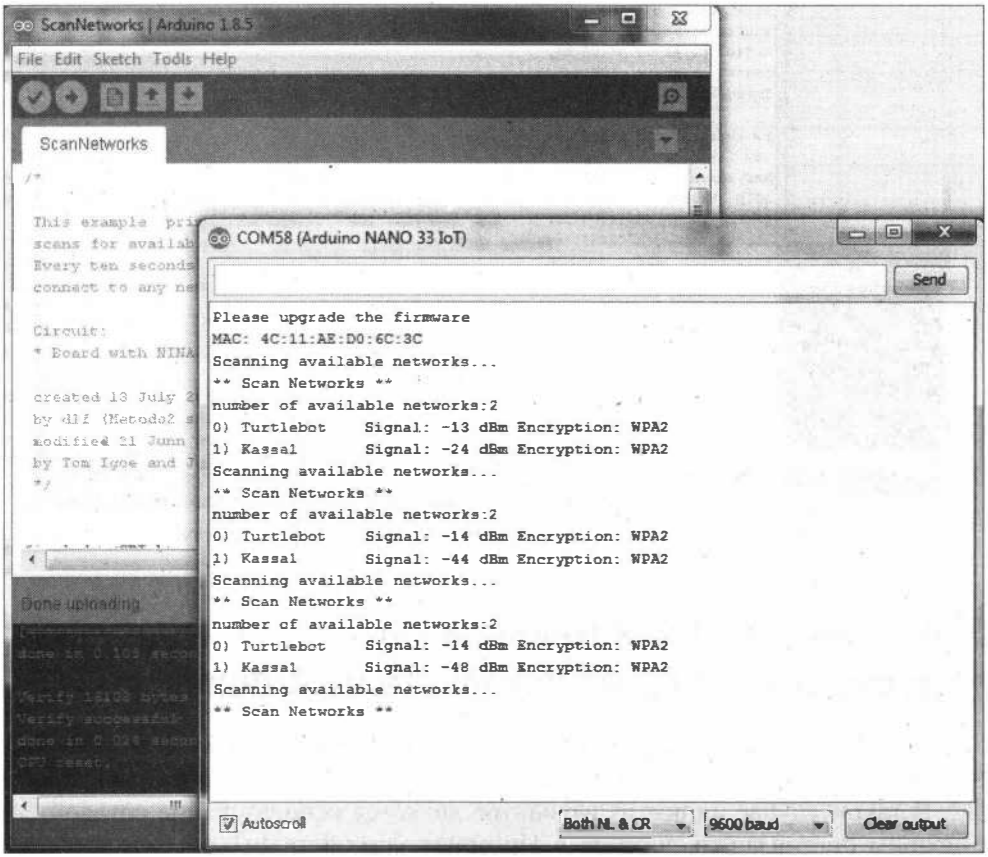


Рис. 3.29. Поиск доступных точек доступа Wi-Fi Arduino Nano 33 IoT (пример ScanNetworks)

Теперь необходимо загрузить скетч подключения платы Arduino Nano 33 IoT к найденной точке доступа. Загружаем в плату пример из библиотеки connectWithWPA.ino и вводим на вкладке данные для своей точки доступа Wi-Fi:

```
#define SECRET_SSID "Kassa1"
#define SECRET_PASS "********".
```

После загрузки скетча откроем монитор последовательного порта (рис. 3.30), где можно наблюдать за процессом подключения платы Arduino Nano 33 IoT к точке доступа.

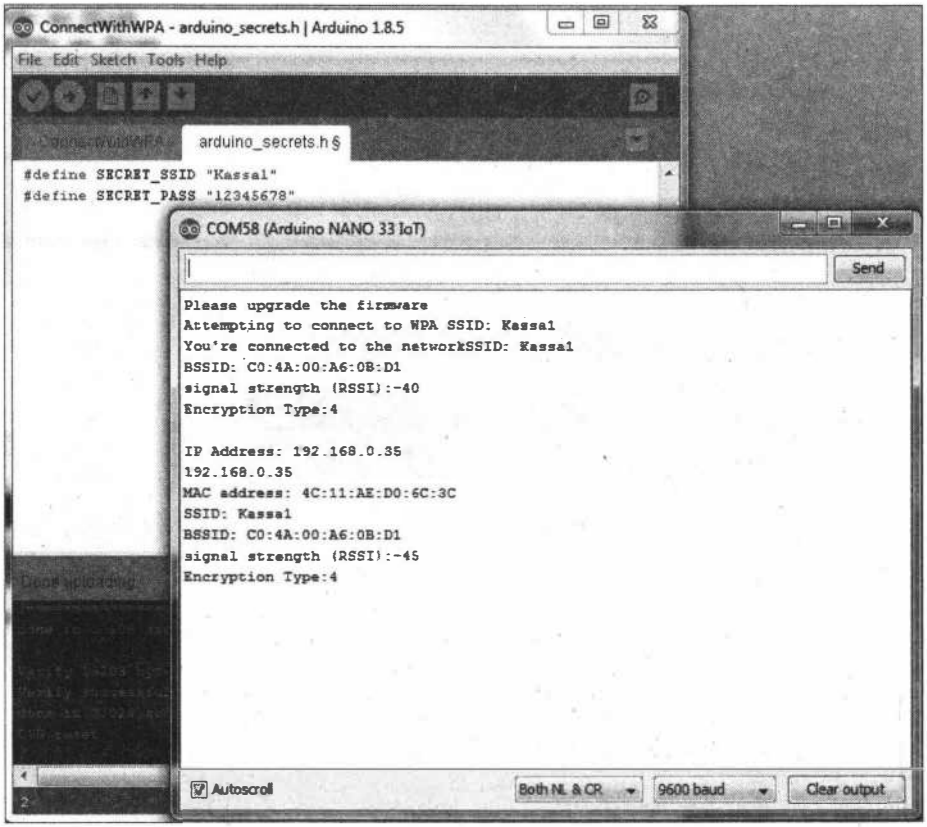


Рис. 3.30. Подключение платы Arduino Nano 33 IoT к точке доступа

3.3. Использование сотовой связи для доступа в Интернет устройств на Arduino

Устройства IoT далеко не всегда могут иметь доступ к сетевому фиксированному интернет-соединению — такому, как Ethernet или беспроводная точка доступа Wi-Fi. В таком случае одним из вариантов является использование сотовой связи. Рассмотрим организацию доступа в Интернет устройств IoT на базе Arduino по сотовой связи.

3.3.1. Arduino Uno и GSM/GPRS Shield SIM900

Плата расширения GSM/GPRS Shield SIM900 — это GSM-модем, на основе которого может быть построено множество проектов Интернета вещей (IoT). Вы можете использовать эту плату расширения, чтобы выполнять почти все, что может обычный сотовый телефон: отправлять и принимать текстовые SMS-сообщения, совершать или принимать телефонные звонки, выполнять подключение к Интернету через GPRS, TCP/IP и многое другое! Надо также отметить, что плата расширения GSM/GPRS Shield SIM900 поддерживает четырехдиапазонную сеть GSM/GPRS, а это означает, что она работает практически в любой точке мира.

Рассмотрим один из вариантов этого шилда — SIM900 Quad-Band GPRS Shield (рис. 3.31) на основе чипа SIM900.

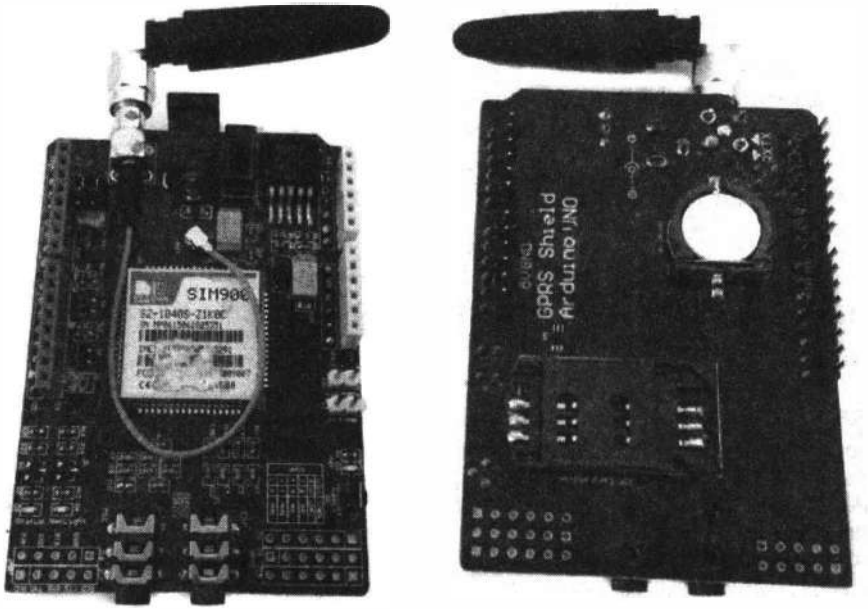


Рис. 3.31. Плата расширения Arduino SIM900 Quad-Band GPRS Shield

Плата расширения SIM900 Quad-Band GPRS Shield обладает удивительным количеством функций при столь незначительных размерах. Вот лишь некоторые из них:

- поддержка четырех диапазонов: GSM850, EGSM900, DCS1800 и PCS1900;
- подключение к любой глобальной сети GSM с любой SIM-картой 2G;
- совершение и прием голосовых звонков (с использованием внешних наушника и электретного микрофона);
- отправка и получение SMS-сообщений;
- отправка и получение данных GPRS (TCP/IP, HTTP и т. д.);
- сканирование и прием радиопрограмм FM;

- мощность передачи:
  - класс 4 (2 Вт) для GSM850;
  - класс 1 (1 Вт) для DCS1800;
- набор AT-команд для управления через последовательный порт;
- разъемы U.FL и SMA для сотовой антенны;
- работа с полноразмерной SIM-картой.

Плата расширения GSM/GPRS Shield SIM900 использует для связи с Arduino последовательный интерфейс UART. Она поддерживает скорость передачи от 1200 бит/с до 115200 бит/с с автоматическим определением скорости.

С помощью перемычек вы можете подключить контакты RX/TX платы расширения к программному последовательному порту D8/D7 (для работы с использованием библиотеки SoftwareSerial) или аппаратному последовательному порту D1/D0 платы Arduino.

Помимо платы расширения GSM/GPRS Shield SIM900 и самой платы Ардуино, вам также понадобится SIM-карта, через которую осуществляется взаимодействие с сотовой сетью. Провайдер (оператор) сотовой сети предоставляет вам SIM-карту и обеспечивает GSM-покрытие сети той области, где вы находитесь (или осуществляет роуминг с той компанией, у которой покрытие сети охватывает ваше местоположение).

Плату расширения GSM/GPRS Shield SIM900 можно включить двумя способами:

- аппаратным (нажатием кнопки PWRKEY);
- программным.

Для чего нужен программный сброс?

Иногда при обмене по GPRS возникают ситуации, после которых модуль может зависнуть. Причиной этого могут стать некорректные данные, пришедшие по сети и загнавшие чип SIM900 в ступор, или помехи на линии обмена модуля и контроллера, или еще какие бы то ни было неведомые проблемы. Производитель чипа предлагает в таких случаях перезагружать модуль с помощью специальной последовательности импульсов, подаваемых на вход PWRKEY (рис. 3.32).

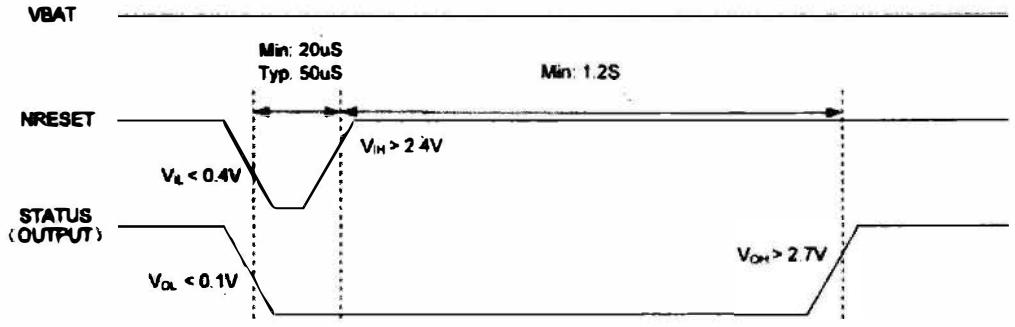


Рис. 3.32. Программный сброс для GSM/GPRS Shield SIM900

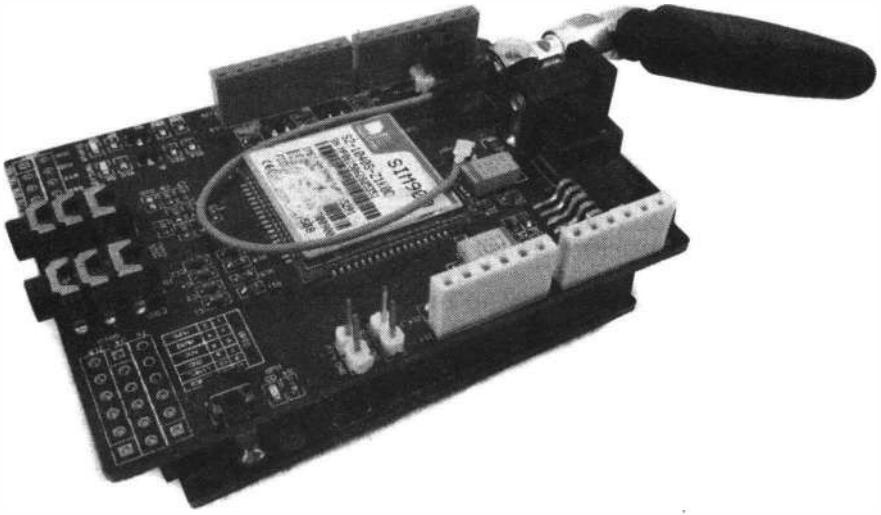


Рис. 3.33. Установка платы расширения GSM/GPRS Shield SIM900 на плату Arduino

Плату расширения GSM/GPRS Shield SIM900 выполнена в стандартном формате шилда для плат Arduino. Контакты шилда (гребенки) легко вставляются в разъемы платы, образуя при этом своего рода «бутерброд» (рис. 3.33).

Рассмотрим управление модулем GSM/GPRS Shield SIM900 с помощью AT-команд. Для этого установим модуль на плату Arduino и подключим ее к компьютеру. Arduino-скетч отправки и получения данных между компьютером и модулем GSM/GPRS Shield SIM900 через плату Arduino показан в листинге 3.6.

Листинг 3.6

```
#include <SoftwareSerial.h>
// создание объекта
 SoftwareSerial grs(7, 8); // RX, TX
 // скорость обмена
 #define GSMbaud 9600
 String str1;
 char buff[100];

 void setup() {
    Serial.begin(9600);
    gsm.begin(GSMbaud);
    Serial.println("Start");
 }

 void loop() {
    if (Serial.available()) {
        str1 = Serial.readStringUntil('\n');
        str1.toCharArray(buffer, hh.length() + 1);
        gsm.write(buffer);
        gsm.board.write('\n');
    }
```

```
    if (gsm.available()) {
       Serial.write(gprs.read());
    }
}
```

Электронный архив

Скетч, соответствующий листингу 3.6, можно найти в папке *examples\03\03_06* сопровождающего книгу электронного архива (см. *приложение*).

Для передачи данных по GPRS нужно отправить на шилд последовательно следующие AT-команды:

```
AT+SAPBR=1,1    //Открыть несущую (Carrier)
//тип подключения - GPRS
AT+SAPBR=3,1,"CONTYPE","GPRS"
//APN, для Билайна - internet
AT+SAPBR=3,1,"APN","internet.beeline.ru"
//Инициализировать HTTP
AT+HTTPINIT
//Carrier ID для использования.
AT+HTTPPARA="CID",1
//Отправить данные методом GET
AT+HTTPPARA="URL","http:/**************"
AT+HTTPACTION=0
//Дождаться ответа
AT+HTTPREAD
//Остановить HTTP
AT+HTTPTERM
```

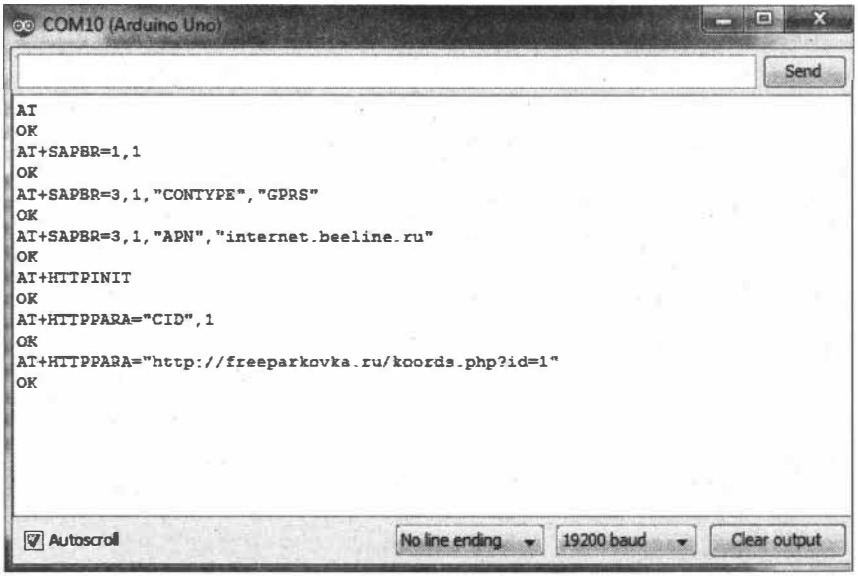


Рис. 3.34. Отправка AT-команд на GSM/GPRS Shield SIM900 из последовательного порта

Пробуем отправить эту последовательность из монитора последовательного порта (рис. 3.34).

3.3.2. Подключение к Интернету платы Arduino MKR GSM 1400

С платой Arduino MKR GSM 1400 мы познакомились в *разд. 2.2.3*. Здесь же мы рассмотрим программирование этой платы в среде Arduino IDE.

По умолчанию среда Arduino IDE настроена только на AVR-платы. Для работы с платой Arduino MKR GSM 1400 необходимо добавить в Менеджере плат среды Arduino IDE поддержку платформы Arduino SAMD (32bit ARM Cortex-M0+). Для этого выполним в среде Arduino IDE команду **Инструменты | Плата | Менеджер плат**, найдем **Arduino SAMD** и нажмем на кнопку **Установка**. После установки программного обеспечения в меню **Инструменты | Плата** среды Arduino IDE появится поддержка плат Arduino Zero, MKR и Nano, в том числе и платы Arduino MKR GSM 1400.

Подключим плату Arduino MKR GSM 1400 к компьютеру с помощью USB-кабеля. После подключения соответствующий драйвер должен будет установиться автоматически, и среда Arduino IDE определит плату (рис. 3.35).

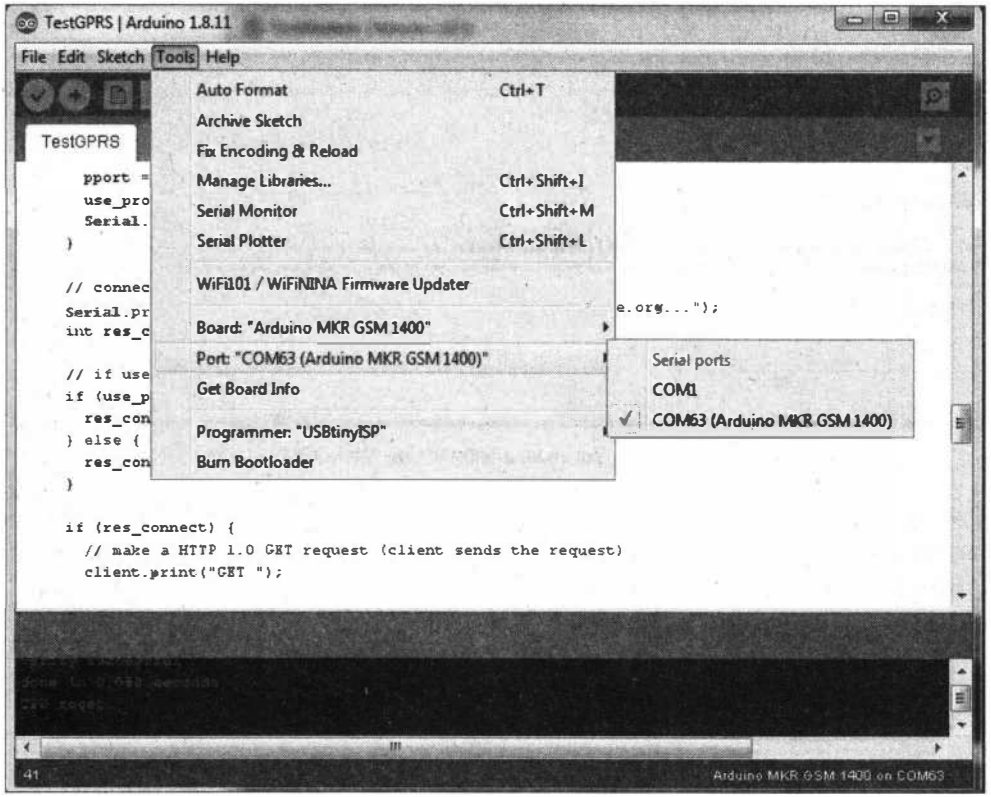


Рис. 3.35. Поддержка плат Arduino MKR GSM 1400 в Arduino IDE установлена

Для работы платы Arduino MKR GSM 1400 в сотовой сети установите в слот на ее обратной стороне SIM-карту формата micro-SIM. Вам также понадобится подключить к ней внешний источник питания — через разъем для подключения внешних Li-Po и Li-Ion аккумуляторов.

В плату Arduino MKR GSM 1400 встроен сотовый модем, который передает данные из ее последовательного порта в сеть GSM. По умолчанию этот модем выполняет операции с помощью серии AT-команд. Но мы воспользуемся библиотекой MKRGSM, которая абстрагирует низкоуровневую связь между модемом и SIM-картой и с помощью которой можно совершать и принимать голосовые вызовы, отправлять и получать SMS-сообщения и подключаться к Интернету через сеть GPRS. Установить библиотеку можно через Менеджер библиотек: выбираем в меню среды Arduino IDE пункт **Эскиз | Подключить библиотеку | Управлять библиотеками**, ищем библиотеку **MKRGSM** и нажимаем на кнопку **Установка** (рис. 3.36).

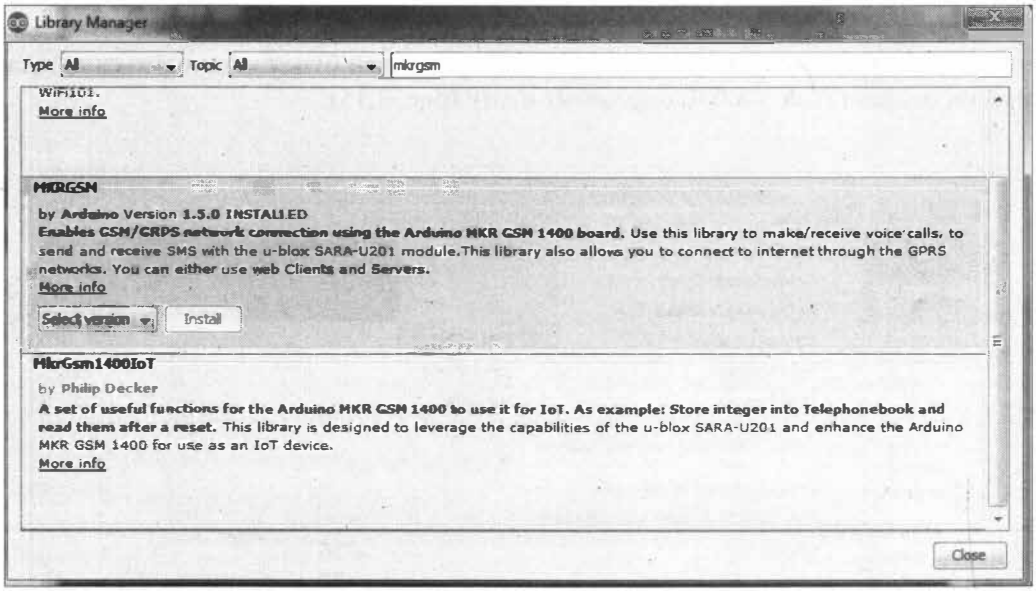


Рис. 3.36. Установка библиотеки MKRGSM

Рассмотрим подключение платы Arduino MKR GSM 1400 к сети GPRS, для чего загрузим в плату из библиотеки MKRGSM пример GPRSPing.ino. Вам понадобится при этом задать настройки доступа к сети GPRS для своего сотового оператора:

```
#define SECRET_PINNUMBER ""
#define SECRET_GPRS_APN  "internet.beeline.ru" // replace your GPRS APN
#define SECRET_GPRS_LOGIN "beeline"    // replace with your GPRS login
#define SECRET_GPRS_PASSWORD "beeline" // replace with your GPRS password
```

Можно изменить и сайт, до которого будет выполняться команда `ping`:

```
String hostName = "www.ya.ru";
```

Итак, загружаем скетч в плату и открываем монитор последовательного порта — связь установлена (рис. 3.37). Теперь мы можем использовать это соединение для подключения платы Arduino MKR GSM 1400 к сети Интернет.

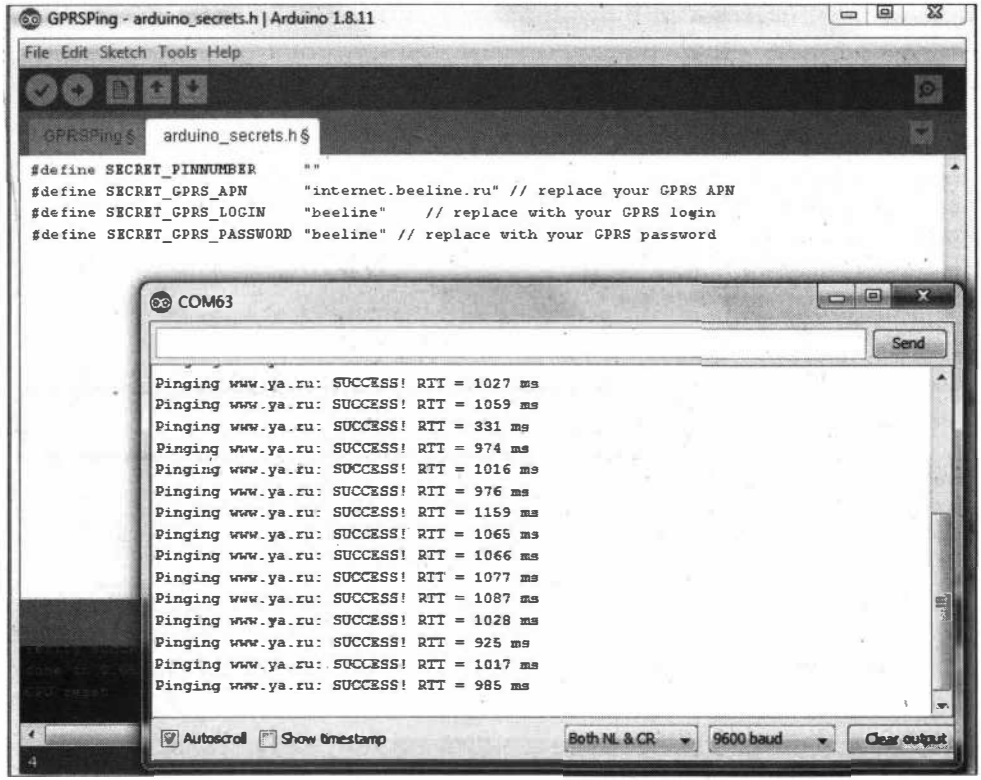


Рис. 3.37. Соединение GPRS для платы Arduino MKR GSM 1400 установлено

3.4. Подключение по Wi-Fi платы ESP32

С платой ESP32 мы познакомились в *разд. 2.4*. Здесь же мы рассмотрим программирование этой платы в среде Arduino IDE.

Для работы с платами ESP32 необходимо добавить в Менеджере плат среды Arduino IDE поддержку этих плат. Для этого выполните в среде Arduino IDE команду **Файл | Настройки,** в поле **Дополнительные ссылки для Менеджера плат** введите адрес: **https://dl.espressif.com/dl/package_esp32_index.json** и нажмите кнопку **OK**. Если в этом поле уже имеется ссылка для других плат (например, для плат ESP8266), то можно разделить ссылки запятой (рис. 3.38):

https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json

Далее выполните в среде Arduino IDE команду **Инструменты | Плата | Менеджер плат**, найдите **ESP32** и нажмите на кнопку **Установка** (рис. 3.39). После установки

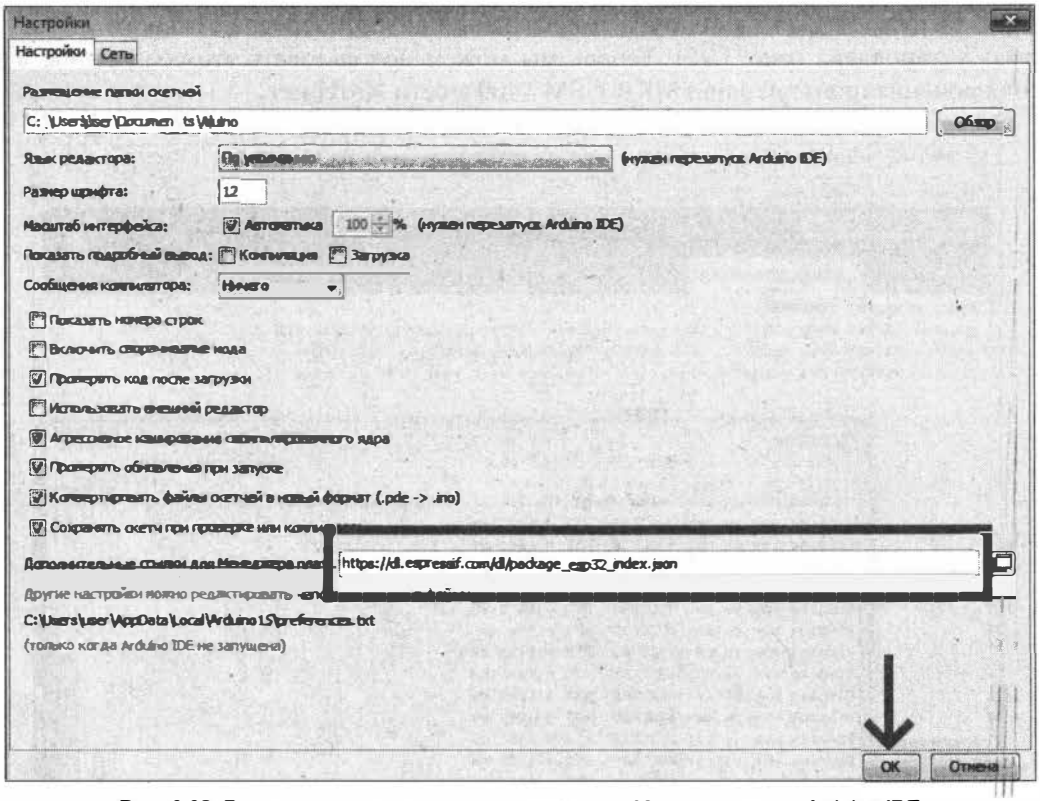


Рис. 3.38. Ввод адресов для дополнительных плат Менеджера плат Arduino IDE

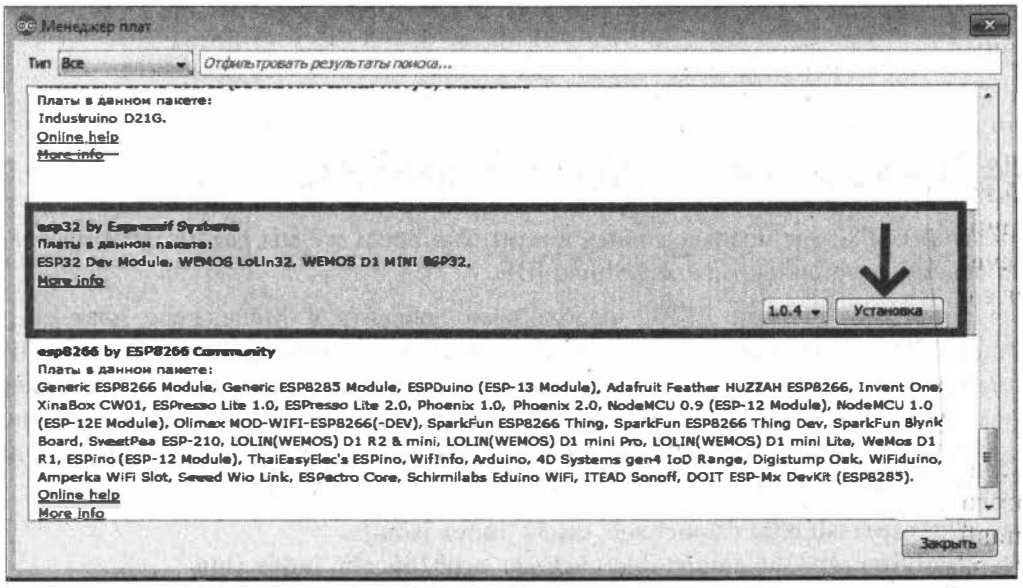


Рис. 3.39. Установка программного обеспечения поддержки плат ESP32

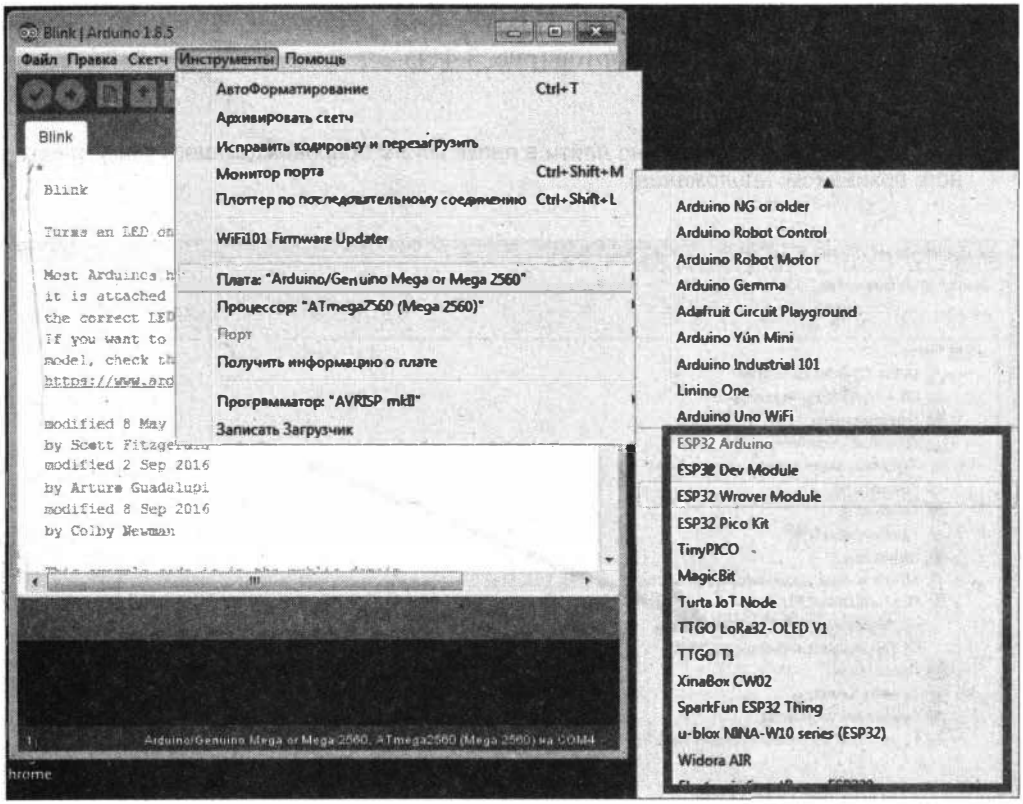


Рис. 3.40. Поддержка плат ESP32 в Arduino IDE установлена

программного обеспечения в меню **Инструменты | Плата** среды Arduino IDE появится поддержка плат ESP32 (рис. 3.40).

Подключим плату ESP32 к компьютеру с помощью USB-кабеля. Если в списке портов плата не появилась, значит, для нее не установлен драйвер. В зависимости от модели платы ESP32 ей могут понадобиться разные драйверы. Так, если ваша плата ESP32 содержит чип cp2102 от компании SiLabs, драйвер CP2102 можно загрузить с официального сайта этой компании на странице: **https://www.silabs.com/developers/usb-to-uart-bridge-vcp-drivers**. После установки драйвера плата появится в списке COM-портов.

> ***ЭЛЕКТРОННЫЙ АРХИВ***
>
> Папки с инсталляционными файлами драйвера CP2102 можно найти в папке *drivers* сопровождающего книгу электронного архива (см. *приложение*).

Если у Вас плата FireBeetle ESP32, необходимо установить другой драйвер. Для этого зайдите по ссылке: **https://git.oschina.net/dfrobot/FireBeetle-ESP32/raw/master/FireBeetle-ESP32.inf** и скопируйте содержимое этой страницы в файл FireBeetle-ESP32.inf. Затем в Диспетчере устройств (Прочие устройства) правой кнопкой выберите плату **FireBeetle-ESP32**, выполните команду **Обновить драйверы | Выполнить поиск драйверов на этом компьютере** и укажите папку, куда

сохранили файл FireBeetle-ESP32.inf. Подтвердите установку, и плата FireBeetle ESP32 появится в списке COM-портов (рис. 3.41).

ЭЛЕКТРОННЫЙ АРХИВ

Файл *FireBeetle-ESP32.inf* можно найти в папке *drivers* сопровождающего книгу электронного архива (см. *приложение*).

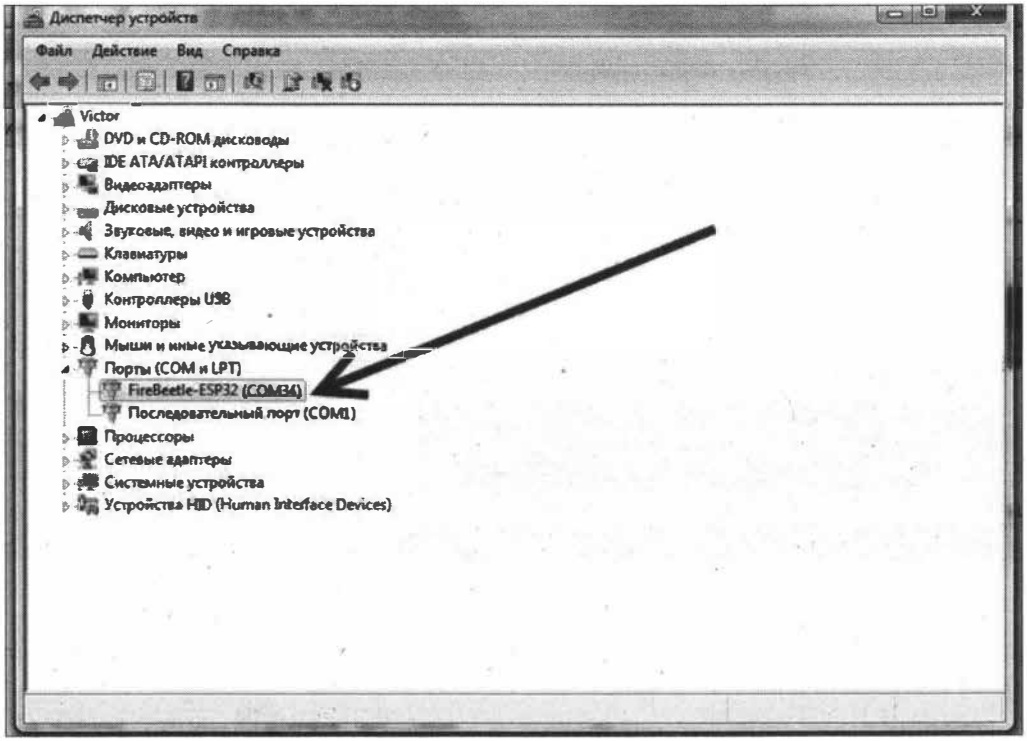


Рис. 3.41. Драйвер платы FireBeetle ESP32 установлен

Попробуем теперь подключить плату ESP32 к точке доступа Wi-Fi. Для этого загрузите в плату скетч из листинга 3.7, подключающий встроенную в среду Arduino IDE библиотеку WiFi.

Листинг 3.7

```
#include <WiFi.h>
// выставляем свои данные для точки доступа
const char* ssid     = "your-ssid";
const char* password = "your-password";

void setup()
{
    Serial.begin(115200);
    delay(10);
```

```
    // Коннект к точке доступа
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
    Serial.println(ssid);
    WiFi.begin(ssid, password);

    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    // подключено
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
}

void loop() {
   ;
}
```

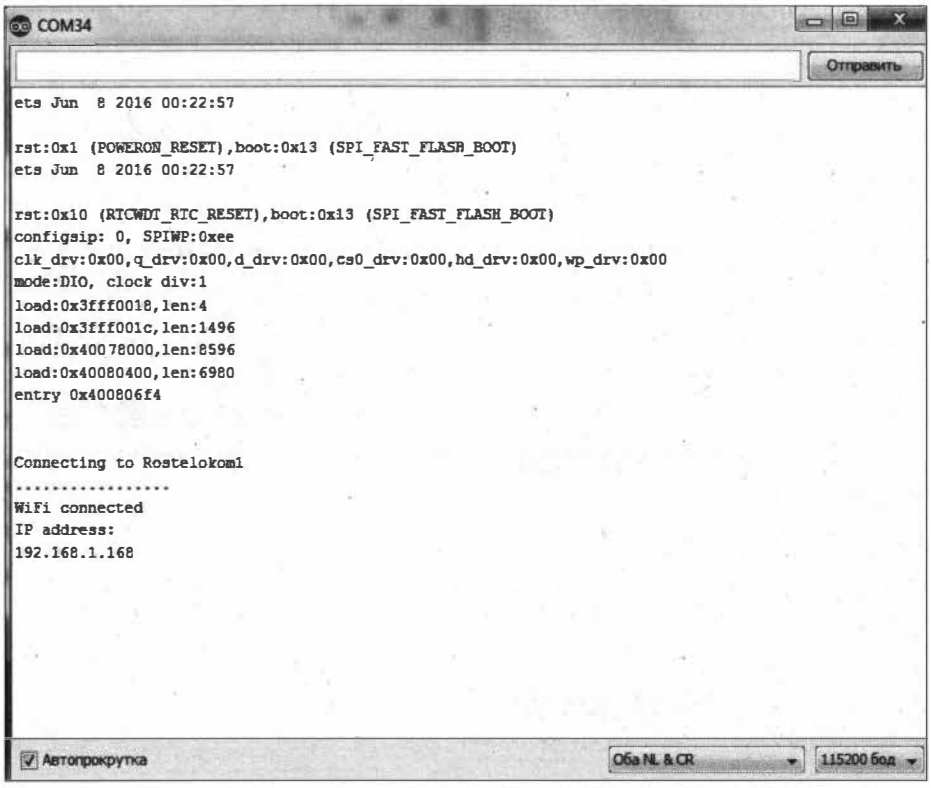


Рис. 3.42. Подключение платы ESP32 к точке доступа Wi-Fi

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 3.7, можно найти в папке *examples\03\03_07* сопровождающего книгу электронного архива (см. *приложение*).

Загрузив скетч в плату ESP32, открываем монитор последовательного порта и видим, что подключение платы ESP32 к сети Wi-Fi произошло успешно (рис. 3.42).

3.5. Подключение к Интернету микрокомпьютера Raspberry Pi Zero W

С микрокомпьютером Raspberry Pi Zero W познакомились в *разд. 2.5*. Здесь же мы рассмотрим установку операционной системы для него и организацию его подключения к сети Интернет.

3.5.1. Установка операционной системы Raspbian

Установку операционной системы Raspbian для микрокомпьютера Raspberry Pi Zero W и подключение его к локальной сети можно осуществить без использования монитора и клавиатуры в облегченном режиме.

Скачайте свежий образ Raspbian с официального сайта по адресу: **https://www.raspberrypi.org/downloads/raspbian/** (рис. 3.43).

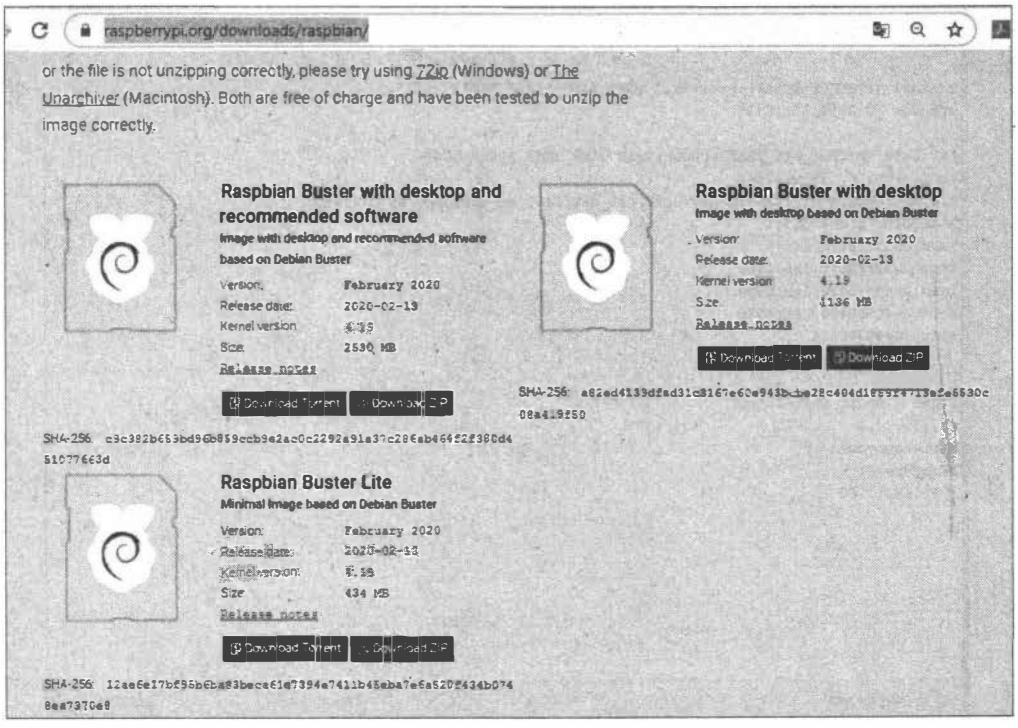


Рис. 3.43. Страница скачивания образа операционной системы Raspbian

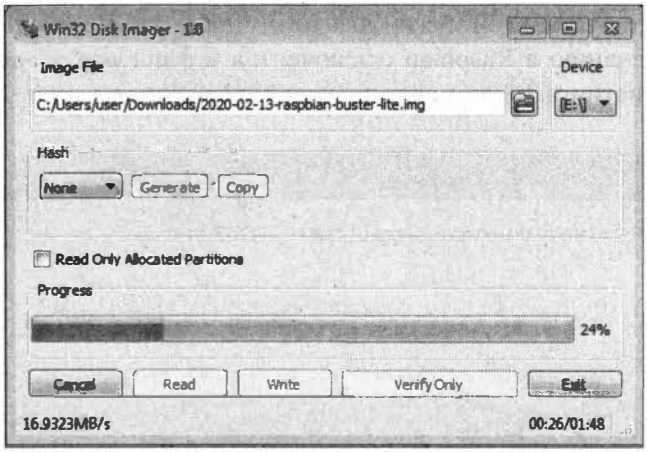


Рис. 3.44. Запись образа Raspbian на карту памяти microSD в программе Win32 Disk Imager

Запишите скачанный образ операционной системы Raspbian на карту памяти microSD. Для этого можно воспользоваться, например, программой Win32 Disk Imager (рис. 3.44).

Записав на карту образ операционной системы Raspbian, откройте эту карту на компьютере и создайте в ее корневом каталоге два файла: ssh (без расширения) и wpa_supplicant.conf (рис. 3.45).

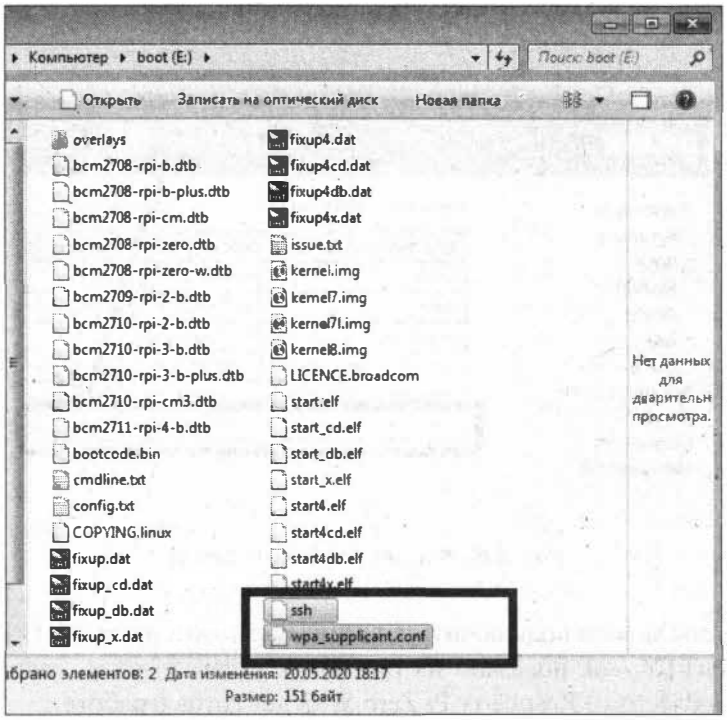


Рис. 3.45. Создание необходимых файлов в корневом каталоге карты microSD с образом операционной системы Raspbian

Файл ssh оставьте пустым (он нужен для того, чтобы активировать доступ по SSH, который по умолчанию в Raspbian отключен), а в файл wpa_supplicant.conf внесите содержимое из листинга 3.8, поменяв значения параметров `ssid` и `psk` на свои.

Листинг 3.8

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=RU
network={
ssid="my_DLink"
psk="********"
key_mgmt=WPA-PSK
}
```

3.5.2. Подключение Raspberry Pi Zero W к сети

Установите карту microSD с образом операционной системы Raspbian в слот микрокомпьютера Raspberry Pi Zero W и подайте на него питание. Если имя точки доступа Wi-Fi и ее пароль указаны корректно, то Raspberry Pi Zero W при загрузке должен автоматически подключиться к локальной сети и получить локальный IP-адрес, который можно посмотреть на роутере (рис. 3.46).

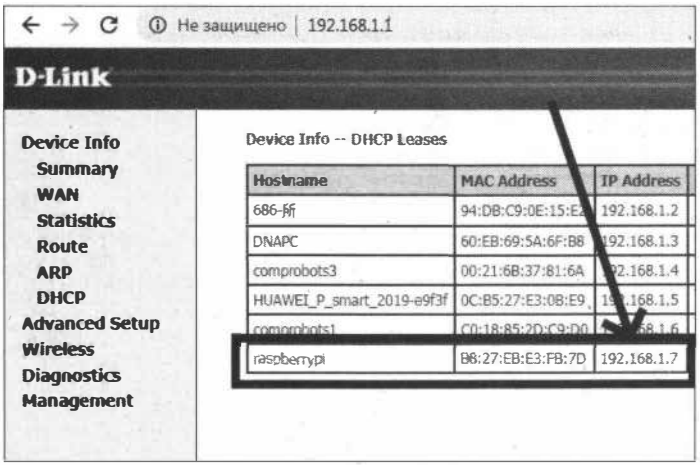


Рис. 3.46. IP-адрес Raspberry Pi Zero W

Зная IP-адрес, мы можем подключиться к микрокомпьютеру по ssh (из ОС Windows программой PuTTY, как показано на рис. 3.47), задав логин `pi`, пароль `raspberry` (рис. 3.48). После этого Raspberry Pi Zero W будет готов к работе.

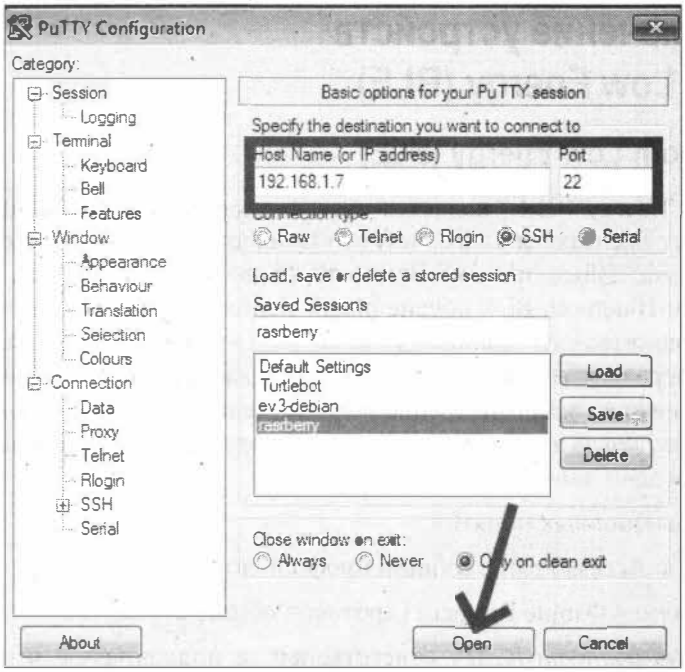


Рис. 3.47. Задание IP-адреса Raspberry Pi Zero W в программе PuTTY

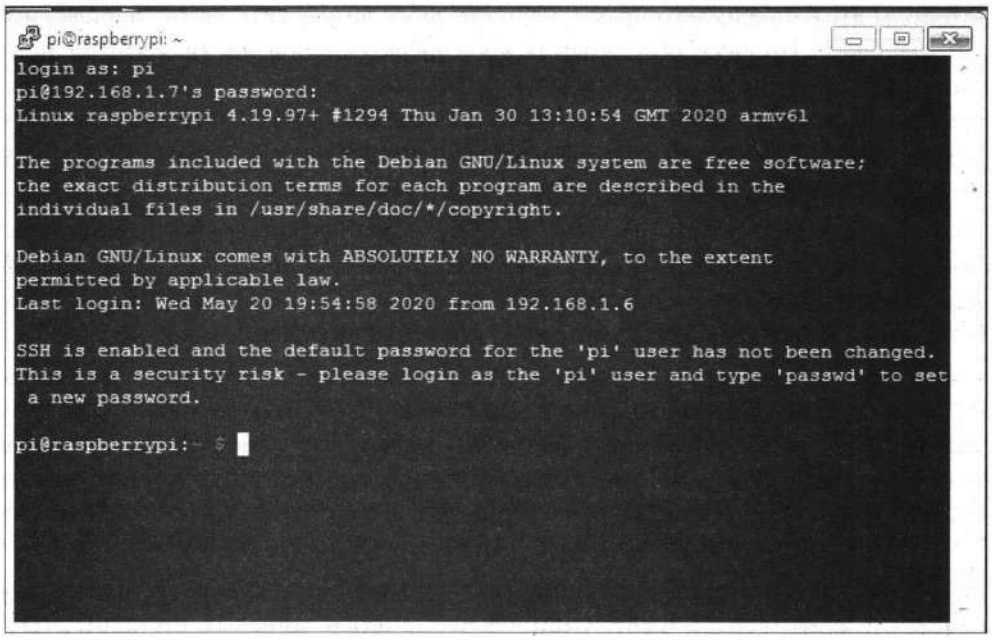


Рис. 3.48. Подключение к Raspberry Pi Zero W по ssh

3.6. Подключение устройств Bluetooth Low Energy (BLE)

3.6.1. Bluetooth Low Energy (BLE)

Bluetooth Low Energy (BLE) является частью спецификации Bluetooth 4.0, которая также включает протоколы классического Bluetooth и протокол высокоскоростного Bluetooth (Classic Bluetooth and Bluetooth High Speed Protocols). По сравнению с классическим Bluetooth BLE обеспечивает использование меньшей мощности при сохранении аналогичного диапазона связи. BLE — это технология, которая всегда отключена и передает только короткие объемы данных, когда это необходимо. Это значительно снижает энергопотребление, что делает его идеальным для использования в случаях, когда требуется постоянное долговременное соединение с низкой скоростью передачи данных.

В BLE есть два основных понятия:

- GAP, Generic Access Profile (общий профиль доступа);
- GATT, Generic Attribute Protocol (протокол общих атрибутов).

Общий профиль доступа (GAP) ответственен за подключение и распространение информации о наличии устройства BLE. GAP отвечает за видимость устройства во внешнем мире, а также играет важную роль в определении того, как устройство взаимодействует с другими устройствами.

Протоколы BLE предусматривают наличие двух видов устройств: периферийных и центральных. Процесс обеспечения *видимости устройств* (advertising process) заключается в том, что периферийное устройство каждые 2 секунды отправляет в окружающую среду данные о своем существовании. Если эти данные получит центральное устройство, то оно отправит запрос на сканирование. В ответ периферийное устройство пришлет данные результата сканирования.

Используя *протокол общих атрибутов (GATT)*, два устройства BLE обмениваются данными друг с другом, используя понятия: *сервис* (service) и *характеристика* (characteristic). Этот протокол сохраняет все сервисы и характеристики в справочной таблице с использованием 16-битных идентификаторов.

Сервис может иметь много характеристик. Каждый сервис уникален сам по себе и имеет универсальный уникальный идентификатор (UUID), который может быть размером 16 битов — для официальных адаптированных сервисов или 128 битов — для пользовательских сервисов. Характеристики содержат одну точку данных и схожи с сервисами — каждая характеристика имеет уникальный идентификатор (UUID), который отличается от другой характеристики.

Спецификацией SIG для характеристик и сервисов устройств BLE (**https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf**) предусмотрено, что любое устройство BLE, которое официально приняло UUID от SIG, должно использовать идентификатор, указанный им в своих приложениях.

3.6.2. Подключение платы Arduino Nano 33 BLE Sence

С платой Arduino Nano 33 BLE Sence мы познакомились в *разд. 2.3.2*. Здесь же мы рассмотрим программирование этой платы в среде Arduino IDE.

Для работы с платой Arduino Nano 33 BLE Sence необходимо добавить ее поддержку в Менеджере плат среды Arduino IDE. Для этого выполним в среде Arduino IDE команду **Инструменты | Плата | Менеджер плат**, найдем **Arduino Nano 33 BLE** и нажмем на кнопку **Установка** (рис. 3.49) — начнется процесс установки соответствующего ПО и драйверов (рис. 3.50). По завершении этого процесса (рис. 3.51)

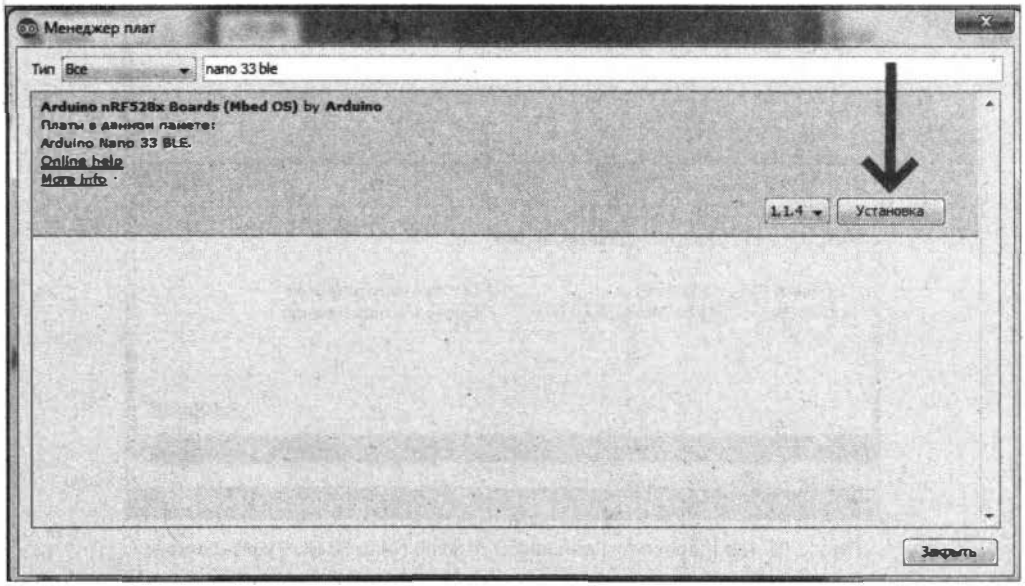


Рис. 3.49. Установка поддержки для Arduino Nano 33 BLE Sence в Arduino IDE

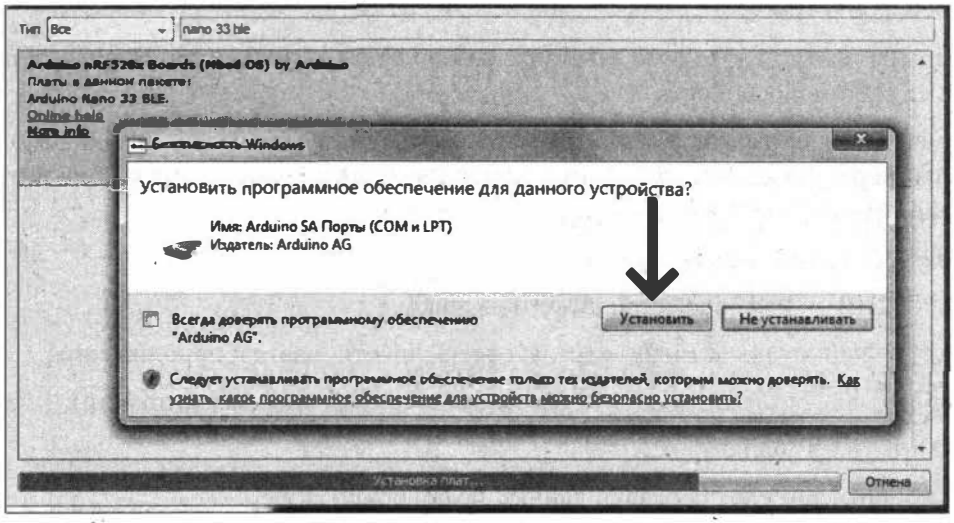


Рис. 3.50. Установка программного обеспечения и драйверов для Arduino Nano 33 BLE Sence

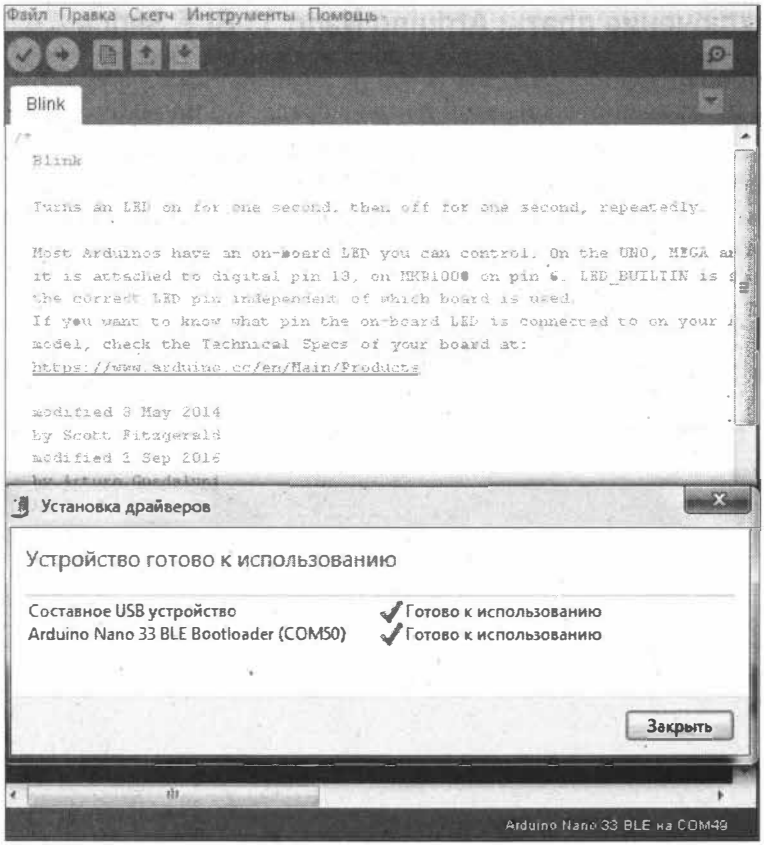


Рис. 3.51. ПО и драйверы для платы Arduino Nano 33 BLE установлены

в списке плат Arduino IDE появится Arduino Nano 33 BLE и обнаружится подключение к порту (рис. 3.52).

Теперь на плату необходимо загрузить любой скетч — например, Blink — и проверить ее работоспособность.

Кроме того, для работы с датчиками, установленными на плате, необходимо установить через Менеджер библиотек командой **Скетч | Подключить библиотеку | Управлять библиотеками** следующие библиотеки:

- HTS221 — для датчика температуры и влажности;
- LPS22HB — для барометрического датчика;
- APDS9960 — для датчика жестов, освещенности, цвета и приближения;
- LSM9DS1 — для датчика IMU (акселерометр, гироскоп, магнитометр);
- PDM — для микрофона;
- ArduinoBLE — для передачи данных по Bluetooth BLE.

Загрузите соответствующие примеры в плату и проверьте их работу.

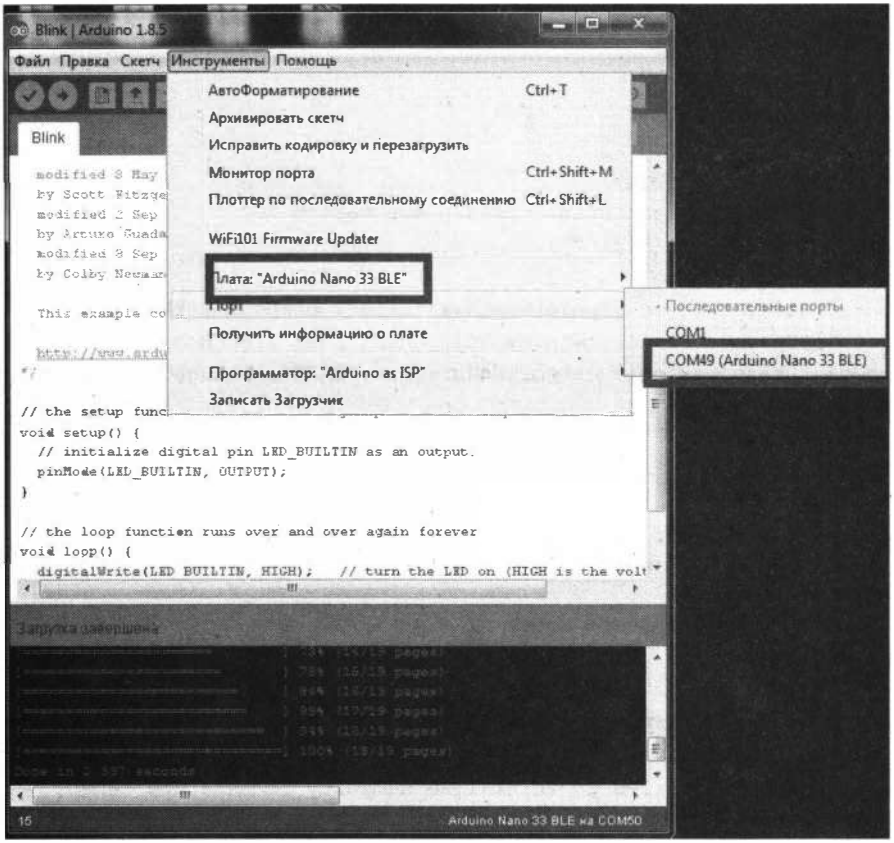


Рис. 3.52. Плата Arduino Nano 33 BLE готова к работе

3.6.3. Отправка по BLE данных с датчиков платы Arduino Nano 33 BLE Sence

Для отправки по BLE данных с датчиков HTS221 (относительной влажности воздуха и температуры) и LPS22HB (барометрический датчик) платы Arduino Nano 33 BLE Sence мы воспользуемся библиотекой ArduinoBLE.

В скетче (листинг 3.9) сначала определим BLE сервиса (uuid – 0x1815 Automation IO):

```
BLEService meteoBLEsense("1815");
```

и характеристики для температуры, влажности и атмосферного давления:

```
BLEIntCharacteristic meteoTemperatureChar("2A6E", BLERead | BLENotify);
BLEUnsignedIntCharacteristic meteoHumidityChar("2A6F", BLERead | BLENotify);
BLEUnsignedIntCharacteristic meteoPressureChar("2AA3", BLERead | BLENotify);
```

Данные UUID для сервиса и характеристик берем в спецификации SIG для характеристик и сервисов.

Данные с датчиков отправляем с периодичностью 1 раз в 3 секунды.

Листинг 3.9

```
// подключение библиотек
#include <ArduinoBLE.h>
#include <Arduino_HTS221.h>
#include <Arduino_LPS22HB.h>
// определение BLE сервиса (uuid - 0x1815 Automation IO)
BLEService meteoBLEsense("1815");
// характеристики сервиса
BLEIntCharacteristic meteoTemperatureChar("2A6E", BLERead | BLENotify);
BLEUnsignedIntCharacteristic meteoHumidityChar("2A6F", BLERead | BLENotify);
BLEUnsignedIntCharacteristic meteoPressureChar("2AA3", BLERead | BLENotify);
// служебные переменные
float temperature;
float humidity;
float pressure;
unsigned long millissend;

void setup() {
   // запуск последовательного порта
   Serial.begin(9600);
   while (!Serial);
   // запуск датчика влажности и температуры
   if (!HTS.begin()) {
      Serial.println("Failed to initialize humidity temperature sensor!");
      while (1);
   }
   // запуск барометра
   if (!BARO.begin()) {
      Serial.println("Failed to initialize pressure sensor!");
      while (1);
   }
   //
   pinMode(LED_BUILTIN, OUTPUT);
   // запуск BLE
   if (!BLE.begin()) {
      Serial.println("BLE failed to Initiate");
      delay(500);
      while (1);
   }

   // чтение данных с датчиков
   temperature = HTS.readTemperature();
   humidity = HTS.readHumidity();
   pressure = BARO.readPressure();

   BLE.setLocalName("ArduinoMeteoBLEsense");
   BLE.setAdvertisedService(meteoBLEsense);
```

```
   meteoBLEsense.addCharacteristic(meteoTemperatureChar);
   meteoBLEsense.addCharacteristic(meteoHumidityChar);
   meteoBLEsense.addCharacteristic(meteoPressureChar);
   BLE.addService(meteoBLEsense);
   meteoTemperatureChar.writeValue(temperature*100);
   meteoHumidityChar.writeValue(humidity*100);
   meteoPressureChar.writeValue(pressure*7.5);
   // advertising process
   BLE.advertise();
   Serial.println("Bluetooth device is now active, waiting for
connections...");
}

void loop() {
   // подключение центрального устройства
   BLEDevice central = BLE.central();
   if (central) {
      Serial.print("Connected to central: ");
      Serial.println(central.address());
      digitalWrite(LED_BUILTIN, HIGH);
      while (central.connected()) {
         if(millis()-millissend>=3000) {
            temperature = HTS.readTemperature();
            humidity    = HTS.readHumidity();
            pressure = BARO.readPressure();
            meteoTemperatureChar.writeValue(temperature*100);
            meteoHumidityChar.writeValue(humidity*100);
            meteoPressureChar.writeValue(pressure*7.5);
            //
            Serial.println("Meteo BLEsense data:");
            Serial.print("t=");Serial.print(temperature);Serial.print(" °C");
            Serial.print(" h=");Serial.print(humidity);Serial.print(" %");
            Serial.print(" p=");Serial.print(pressure);Serial.print(" kPa");
            Serial.println("");
            Serial.println("");
            //
            millissend=millis();
         }
      }
   digitalWrite(LED_BUILTIN, LOW);
   Serial.print("Disconnected from central: ");
   Serial.println(central.address());
   }
}
```

Электронный архив

Скетч, соответствующий листингу 3.9, можно найти в папке *examples\03\03_09* сопровождающего книгу электронного архива (см. *приложение*).

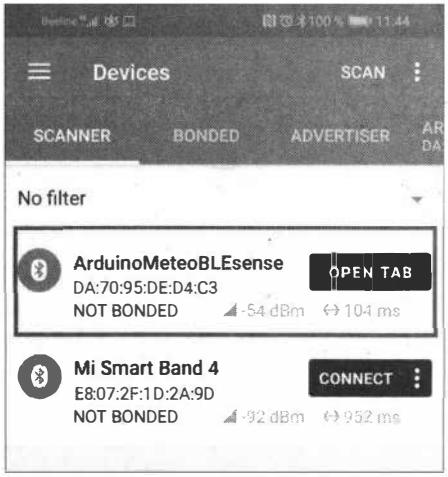


Рис. 3.53. Получение данных в программе nrfConnect из периферийного устройства: этап 1

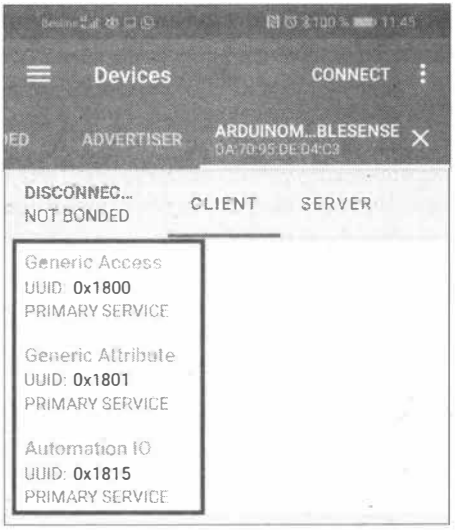


Рис. 3.54. Получение данных в программе nrfConnect из периферийного устройства: этап 2

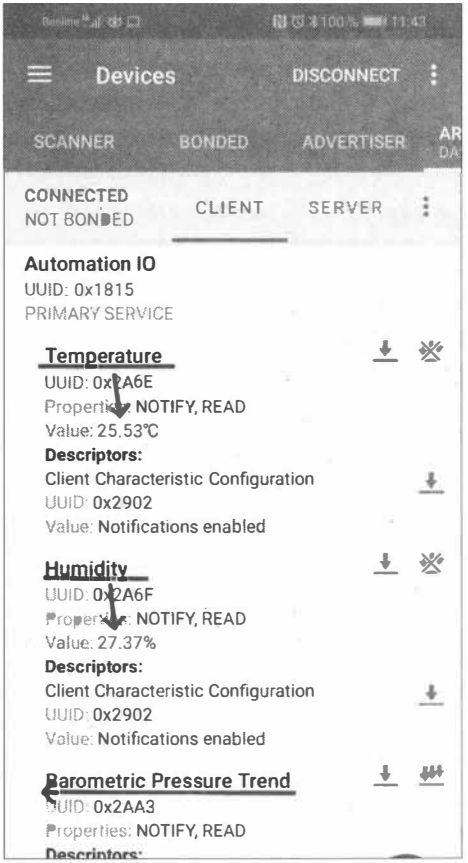


Рис. 3.55. Получение данных в программе nrfConnect из периферийного устройства: этап 3

В качестве центрального устройства мы воспользуемся смартфоном на Android с установленным приложением nrfConnect (рис. 3.53–3.55).

3.7. BLE-связь Arduino Nano 33 BLE Sence и Raspberry Pi Zero W

В *разд. 3.6.3* мы рассмотрели отправку данных по BLE с платы Arduino Nano 33 BLE Sence и получение их на смартфоне. Здесь же мы покажем, как можно организовать получение микрокомпьютером Raspberry Pi Zero W с операционной системой Raspbian данных, отправляемых по BLE платой Arduino Nano 33 BLE Sence.

Установим на Raspberry Pi Zero W пакет Bluez:

```
sudo apt-get update
sudo apt-get install bluez
```

После установки пакета запустим утилиту hcitool для поиска BLE-устройств:

```
sudo hcitool lescan
```

Утилита выводит список BLE-устройств — нас здесь интересует устройство ArduinoMeteoBLEsense (рис. 3.56).

Доступ к службам, запущенным на устройстве Bluetooth, обеспечивается пакетом gatttool. Задействуем его для доступа к службам Bluetooth Low Energy и подключимся к устройству ArduinoMeteoBLEsense, используя его адрес:

```
sudo gatttool -b DA:70:95:DE:D4:C3 -I
```

где флаг `-I` указывает, что мы открываем интерактивный сеанс.

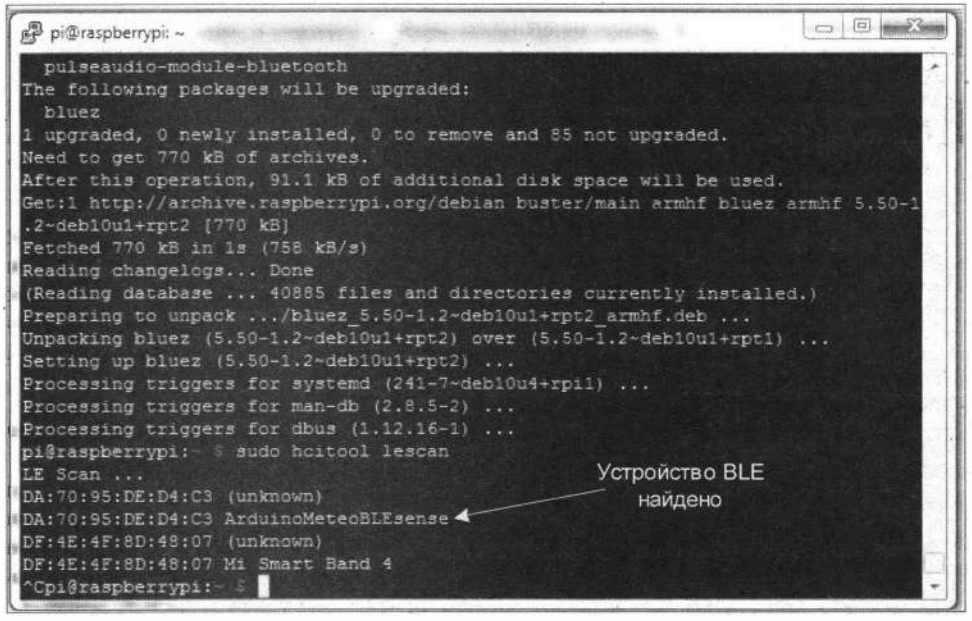


Рис. 3.56. Результат поиска BLE-устройств утилитой hcitool

После открытия сеанса вводим для подключения команду `connect` (рис. 3.57).

Команда `primary` выводит список всех доступных служб, работающих на устройстве с низким энергопотреблением (рис. 3.58).

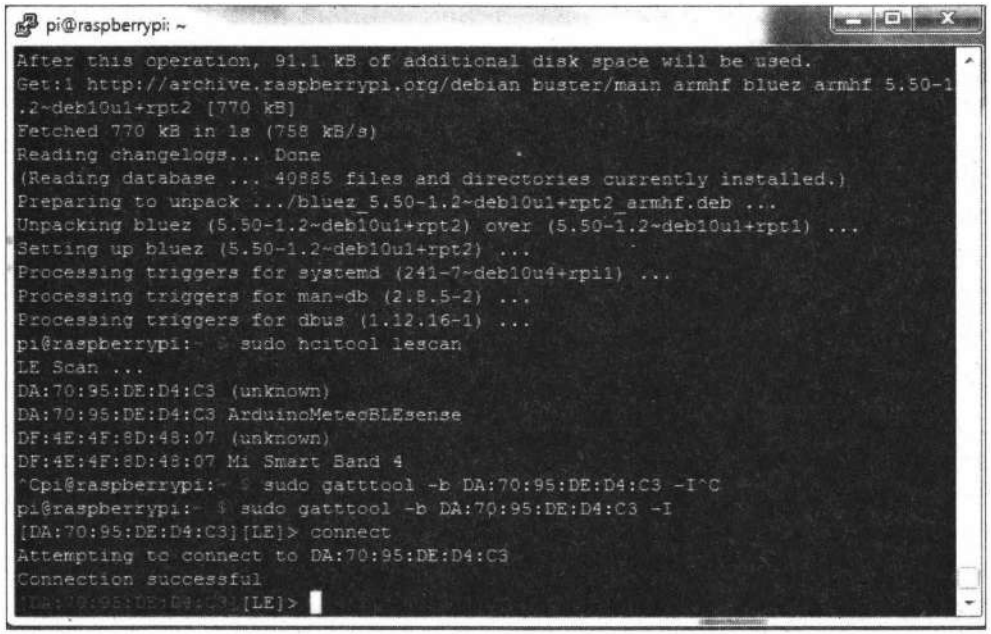


Рис. 3.57. Подключение к устройству BLE

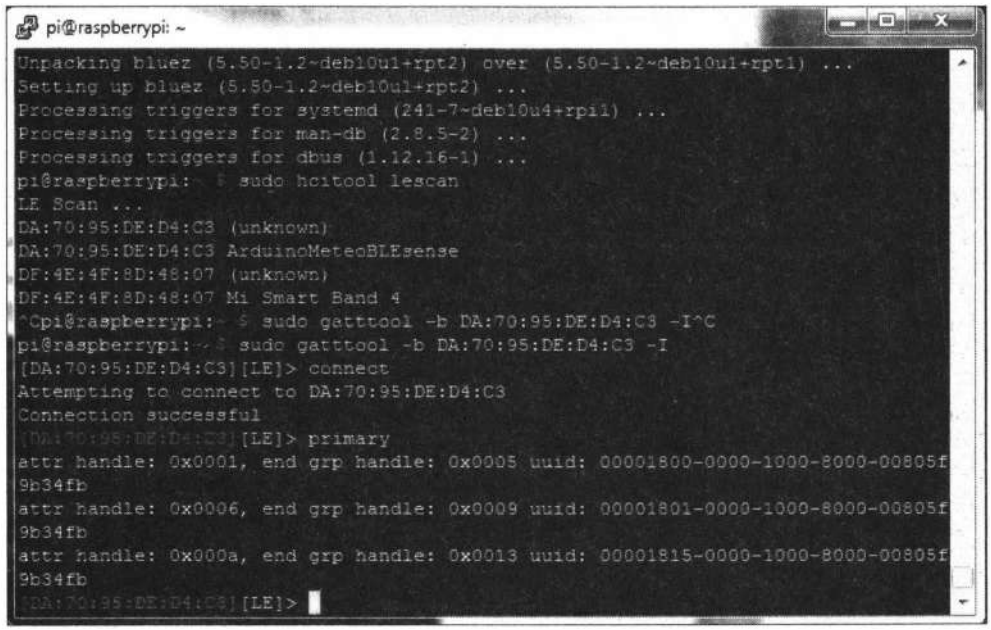


Рис. 3.58. Список доступных служб BLE-устройства

Команда `char-desc` выводит список всех доступных *дескрипторов* — «точек подключения», где можно читать и записывать данные (рис. 3.59). В этом списке находим дескрипторы для данных температуры (**0x000c – uuid 2A6E**), влажности (**0x000f – uuid 2A6F**) и атмосферного давления (**0x0012 – uuid 2AA3**).

Рис. 3.59. Список доступных дескрипторов BLE-устройства

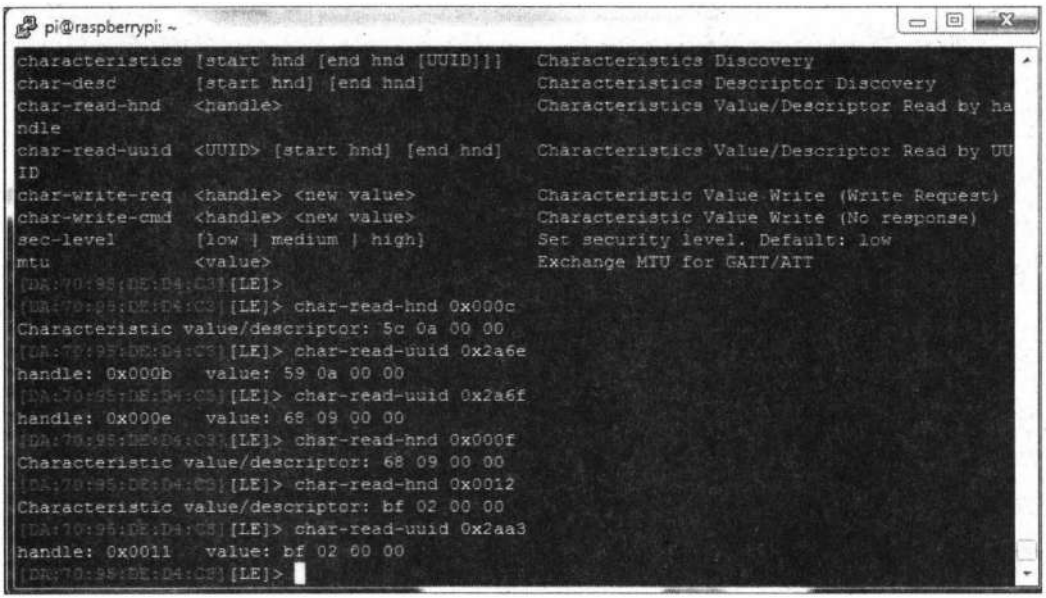


Рис. 3.60. Чтение данных температуры, влажности, атмосферного давления с BLE-устройства

Зная дескрипторы, можно получать данные, отправляемые по BLE.

Команда для чтения данных по дескриптору:

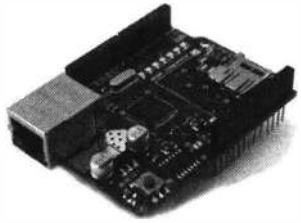
```
char-read-hnd <handle>
```

или по uuid:

```
char-read-uuid <uuid>
```

Эти команды возвращают данные (рис. 3.60) — шестнадцатеричное число стандарта float ieee-754. Для приведения этого числа к десятичному виду требуется соответствующее преобразование.

ГЛАВА 4



Протоколы Интернета вещей

Протоколы передачи данных IoT — это правила, определяющие способы обмена данными между объектами сети Интернета вещей. IoT-устройства, серверы и пользовательские приложения непрерывно обмениваются информацией. Они объединены друг с другом сетями и общаются с помощью различных протоколов передачи данных — это что-то вроде языка, который использует оборудование IoT. При построении IoT-систем могут задействоваться специфичные протоколы: MQTT, AMQP, CoAP, DDS, XMPP, JMS и другие, а также стандартные для обычного Интернета протоколы — например, HTTP. Выбор протокола зависит от решаемой задачи.

4.1. Использование протокола HTTP для связи устройств IoT

HTTP — это повсеместно используемый в Интернете протокол для передачи данных. Он описывает взаимодействие между клиентом и сервером, построенное на базе сообщений, называемых: *запрос* (Request) и *ответ* (Response). Каждое сообщение состоит из трех частей: стартовой строки, заголовка и тела. Стартовая строка, или строка состояния, определяет тип HTTP-сообщения. При этом обязательной для любого сообщения является только стартовая строка. Заголовок HTTP-сообщения может включать одно поле `Host` или несколько полей для передачи различной служебной информации. Тело HTTP-сообщения содержит HTTP-объекты и служит для передачи пользовательской информации. Тело есть не у каждого HTTP-сообщения.

Стартовые строки для запроса и ответа имеют различный формат, но нам интересна только *стартовая строка запроса*, которая выглядит так:

METHOD URI HTTP/*VERSION*

где: *METHOD* — метод HTTP-запроса; *URI* — идентификатор ресурса; *VERSION* — версия протокола.

Пример стартовой строки запроса:

```
POST /cgi-bin/process.cgi HTTP/1.1
```

Заголовок — это набор пар «имя — значение», разделенных двоеточием. В заголовках передается различная служебная информация: кодировка сообщения, название и версия браузера, адрес, с которого пришел клиент (Referrer) и т. д.

Пример заголовка:

```
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.example.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
Accept-Language: ru-ru
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

Тело сообщения — это, собственно, передаваемые данные. В запросе (например, в теле сообщения) передается содержимое файлов, загружаемых на сервер. Но тело сообщения в запросе вообще может отсутствовать. Вот пример тела сообщения:

```
licenseID=string&content=string&/paramsXML=string
```

В ответе передаваемыми данными, как правило, является HTML-страница, которую запросил браузер. Пример ответа сервера:

```
HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

 <html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

Тип HTTP-запроса (также называемый *HTTP-методом*) указывает серверу на то, какое действие мы хотим произвести с ресурсом. Изначально предполагалось, что клиент может хотеть от ресурса только одно — получить его, однако сейчас по протоколу HTTP можно создавать посты, редактировать профиль, удалять сообщения и многое другое.

Для разграничения действий с ресурсами на уровне HTTP-методов были придуманы и другие варианты, основные из которых:

- `GET` — получение ресурса;
- `POST` — создание ресурса;
- `PUT` — обновление ресурса;
- `DELETE` — удаление ресурса.

Сервер дает доступ к своим данным клиентам по определенному интернет-адресу — универсальному идентификатору ресурса (URL), используя основные методы HTTP. Например:

- создать пользователя: `POST /users`;
- удалить пользователя: `DELETE /users/1`;
- получить всех пользователей: `GET /users`;
- получить одного пользователя: `GET /users/1`.

Формат обмена данными может быть любой — здесь нет никаких ограничений. Так, одним из популярных форматов является JSON, хотя можно использовать и другие, например XML.

В завершение несколько слов о сервисе REST. Расшифровывается он так: **RE**presentational **S**tate **T**ransfer. Этот термин первоначально ввел Рой Филдинг (Roy Fielding), один из создателей протокола HTTP. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP.

4.2. MQTT — протокол для сетей с низкой пропускной способностью

Рассмотрим еще один протокол взаимодействия устройств IoT — MQTT (Message Queue Telemetry Transport).

Он обладает рядом преимуществ по сравнению с другими протоколами:

- низкое потребление трафика;
- соединение между клиентом и сервером всегда открыто;
- не нагружает интернет-канал;
- отсутствие задержек в передаче данных;
- удобная система подписок на топики (темы).

Все это дает возможность мониторинга устройств и управления ими в режиме реального времени.

Обмен сообщениями по протоколу MQTT осуществляется между *клиентом* (client), который может быть *издателем* или *подписчиком* (publisher/subscriber) сообщений, и *брокером* (broker) сообщений (например, Mosquitto MQTT). Издатель и подписчик не передают друг другу сообщения напрямую, не устанавливают прямой контакт и могут не знать о существовании друг друга. Издатель отправляет данные на MQTT-брокер, указывая в сообщении определенную тему — топик (topic). Подписчики могут получать разные данные от множества издателей в зависимости от подписки на соответствующие топики.

Топики представляют собой символы с кодировкой UTF-8. Иерархическая структура топиков имеет формат «дерева», что упрощает их организацию и доступ к дан-

ным. Топики состоят из одного или нескольких уровней, которые разделены между собой символом: /. Например:

```
/home/living /living-room1/temperature
```

Устройства MQTT используют определенные типы сообщений для взаимодействия с брокером. Основные типы сообщений представлены далее:

- `Connect` — установить соединение с брокером;
- `Disconnect` — разорвать соединение с брокером;
- `Publish` — опубликовать данные в топик на брокере;
- `Subscribe` — подписаться на топик на брокере;
- `Unsubscribe` — отписаться от топика.

Схема взаимодействия между подписчиком, издателем и брокером по протоколу MQTT показана на рис. 4.1.

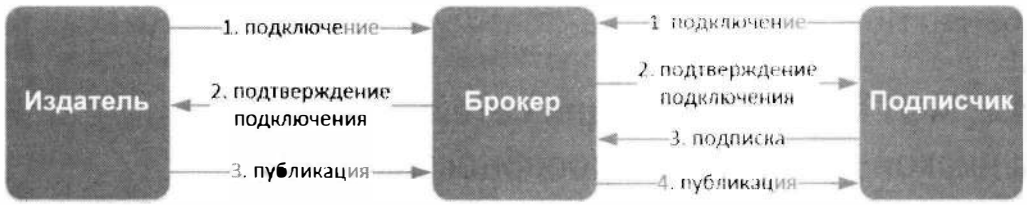


Рис. 4.1. Схема взаимодействия по протоколу MQTT

Однако протокол MQTT требует наличие своего собственного сервера брокера. Здесь могут быть два варианта: либо создавать свой сервер (мы рассмотрим этот вариант в *разд. 6.1*), либо использовать сторонние сервисы. Например, удобный сервис **www.cloudmqtt.com,** у которого есть бесплатный тарифный план (**Cute Cat**).

Разберемся далее, как работать с этим сервисом. После регистрации, выбора тарифного плана и местоположения сервера (EU или USA) создается страница устройства, на которой можно посмотреть ваши данные: адрес сервера, имя и пароль пользователя, порты подключения и ключ **API Key** (рис. 4.2).

4.2.1. Отправка данных по протоколу MQTT

Настроим отправку данных брокеру. Отправляться будут данные температуры и относительной влажности воздуха с датчика DHT11. Подключение платы Arduino к сети Интернет осуществляется через Ethernet Shield (см. *разд. 3.1.1*). Монтажная схема соединений показана на рис. 4.3.

Содержимое скетча приведено в листинге 4.1. Поменяйте в нем переменные `ip`, `sdns`, `gateway` и `subnet` на данные для своей сети (если ваша плата получает IP-адрес по DHCP обратитесь к листингу 3.1). С периодичностью 10 секунд мы получаем данные с датчика DHT11 и отправляем их на сервер в темы `meteo/humidity` и `/meteo/temperature`.

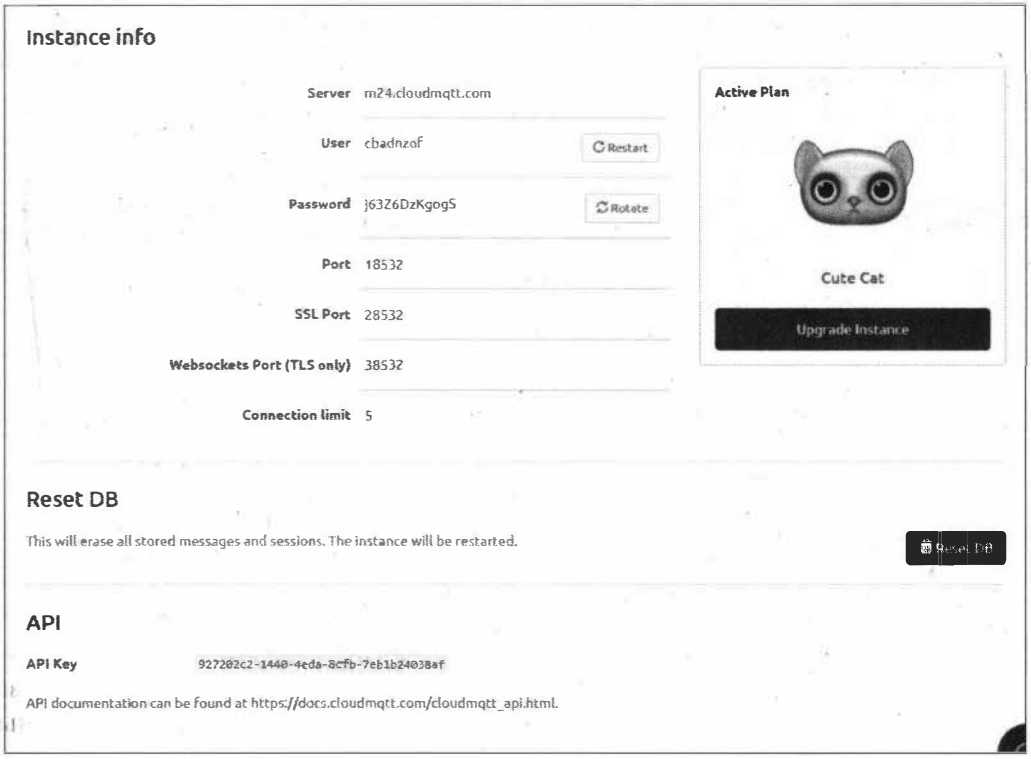


Рис. 4.2. Данные для взаимодействия с брокером MQTT

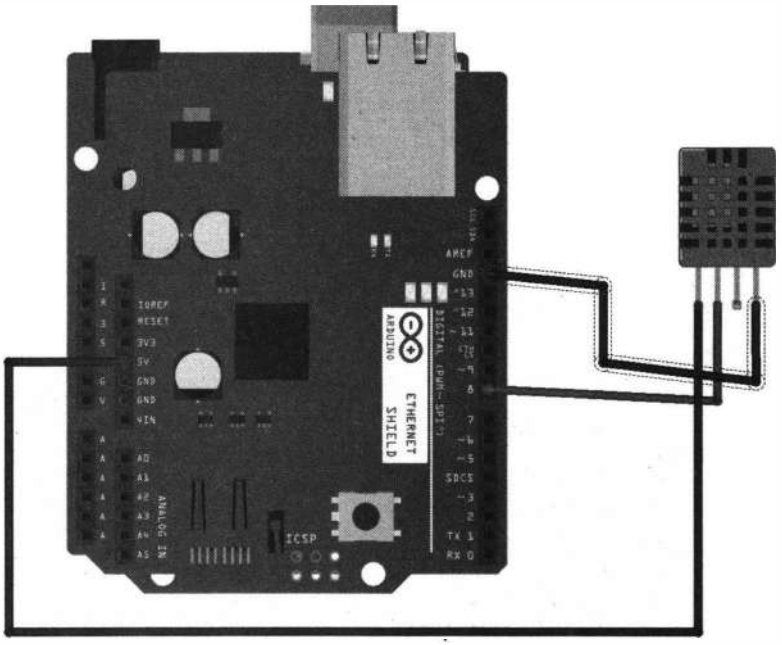


Рис. 4.3. Монтажная схема подключения платы Arduino, шилда Ethernet Shield и датчика DHT11 для отправки данных брокеру MQTT

При написании скетча задействована библиотека PubSubClient.

ЭЛЕКТРОННЫЙ АРХИВ

Библиотека PubSubClient размещена в каталоге *libraries* сопровождающего книгу электронного архива (см. *приложение*).

Листинг 4.1

```
// Получение статического IP-адреса
// MAC-адрес Ethernet Shield (можно увидеть на наклейке на плате) или
// произвольный, но уникальный в сети
#include <Ethernet.h>
#include <SPI.h>
byte mac[] = {0x00, 0xAA, 0xBB, 0xCC, 0xDE, 0x02};
// IP-адрес, назначаемый Ethernet Shield:
byte ip[] = { 192, 168, 0, 111 };
// IP-адрес, dns сервера:
byte sdns[] = { 192, 168, 1, 1 };
// адрес шлюза:
byte gateway[] = { 192, 168, 0, 28 };
// маска:
byte subnet[] = { 255, 255, 255, 0 };
EthernetClient ethclient;

#include <PubSubClient.h>
// данные mqtt
#define mqtt_server  "m24.cloudmqtt.com"
#define mqtt_port 18532
#define mqtt_user  "cbadnzof"
#define mqtt_pass  "j63Z6DzKgogS"
// создание pubsub клиента
PubSubClient client;
// список тем отправки
#define topich "/meteo/humidity"
#define topict "/meteo/temperature"

#include <SimpleDHT.h>
SimpleDHT11 dht11(8);
byte temperature = 0;
byte humidity = 0;

// периодичность отправки
unsigned long millist=0;

void setup() {
  Serial.begin(9600);
  // запуск Ethernet-соединения
  Ethernet.begin(mac, ip, sdns, gateway, subnet);
```

```
  delay(1000);
  Serial.println(Ethernet.localIP());
  client.setClient(ethclient);
  client.setServer(mqtt_server, mqtt_port);
  reconnect();
}
void loop() {
   if (!client.connected()) {
      reconnect();
   }
   client.loop();
   if(millis()-millist>10000) {
      dht11.read(&temperature, &humidity, NULL);
      Serial.print((int)temperature); Serial.print(" *C, ");
      Serial.print((int)humidity); Serial.println(" H");
      publishData((int)temperature, (int)humidity);
      //
      millist=millis();
   }
}

// переподключение к mqtt
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...$");
    // Attempt to connect
    if (client.connect("Arduino+WiFi", mqtt_user, mqtt_pass)) {
       Serial.println("Connected to MQTT server$");
    }
    else {
       Serial.println("Could not connect to MQTT server");
       Serial.println(" try again in 5 seconds$");
       // Wait 5 seconds before retrying
       delay(5000);
    }
  }
}

// отправка данных в темы брокера
void publishData(int t, int h) {
    client.publish(topich, String(h).c_str(), true);
    delay(500);
    client.publish(topict, String(t).c_str(), true);
    delay(500);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный вариант рассмотренного скетча находится в папке *examples\04\04_01* сопровождающего книгу электронного архива (см. *приложение*).

Загрузив скетч в плату, заходим в профиль созданного устройства на сайте **www.cloudmqtt.com** и в пункте **WEBSOCKET UI** наблюдаем приходящие в темы данные (рис. 4.4).

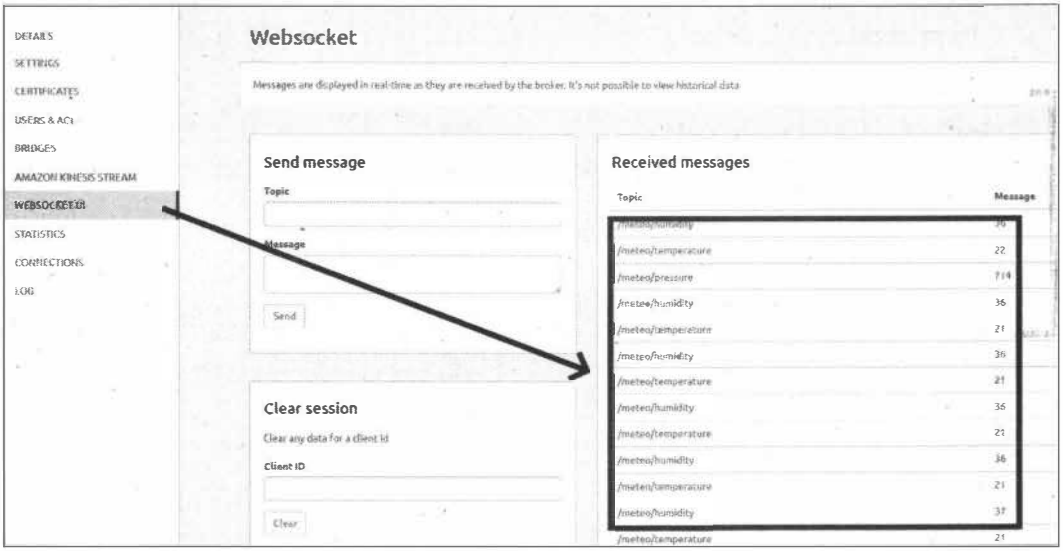


Рис. 4.4. Получение на брокере данных температуры и влажности, отправленных с датчика DHT11 в сервис www.cloudmqtt.com

4.2.2. Получение данных по протоколу MQTT

В *разд. 4.2.1* мы настроили отправку на брокер MQTT, расположенный по адресу **www.cloudmqtt.com**, показаний температуры и относительной влажности с датчика DHT11. Рассмотрим теперь получение данных по протоколу MQTT для управления исполнительными устройствами, подключенными к модулю управления реле Relay Shield. Монтажная схема соединений показана на рис. 4.5.

Отправить данные управляющей схеме на модуле Relay Shield можно из профиля на сайте **www.cloudmqtt.com** через пункт меню **WEBSOCKET UI** (рис. 4.6).

Отправлять данные мы будем в темы: `/meteo/relay1` и `/meteo/relay2`. Чтобы получать команды от сервера (брокера), наше устройство при подключении к брокеру должно подписаться на эти темы:

```
client.subscribe("/meteo/relay1");
client.subscribe("/meteo/relay2");
```

Теперь необходимо назначить функцию обратного вызова `callback` для обработки сообщений от брокера, которая формирует строку команд управления для отправки по последовательному порту на Arduino (листинг 4.2).

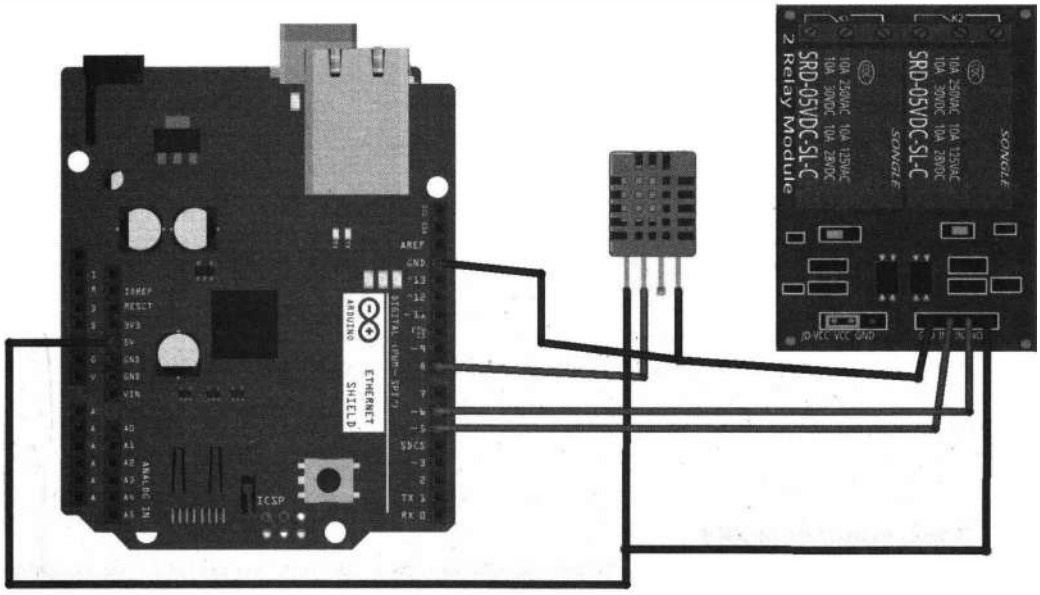


Рис. 4.5. Монтажная схема подключения для управления исполнительными устройствами по протоколу MQTT

Рис. 4.6. Отправка команд управления из профиля **cloudmqtt.com**

Листинг 4.2

```
void callback(char* topic, byte* payload, unsigned int length) {
 String ss(topic);
 int value=0;

 Serial.print(topic);
 Serial.print("=");
```

```
 for (int i = 0; i < length; i++) {
  char receivedChar = (char)payload[i];
  if (receivedChar == '0')
   {Serial.print("0");value=0;}
  if (receivedChar == '1')
   {Serial.print("1");value=1;}
 }
 if(ss.indexOf("/meteo/relay1")!=-1) {
    digitalWrite(pinRelay1,value);Serial.print(" relay1- ok");
 }
 else if(ss.indexOf("/meteo/relay2")!=-1) {
    digitalWrite(pinRelay2,value);Serial.print(" relay2- ok");
 }
 Serial.println();
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный вариант рассмотренного скетча находится в папке *examples\04\04_02* сопровождающего книгу электронного архива (см. *приложение*).

Открываем монитор последовательного порта и при отправке команд на сайте **www.cloudmqtt.com** в темы `/meteo/relay1` и `/meteo/relay2` видим получение сообщений от брокера и установку новых значений для реле (рис. 4.7).

Более удобный вариант работы с сервисом **www.cloudmqtt.com** — использование Android-приложения IoT MQTT Dashboard, позволяющего задействовать смартфон

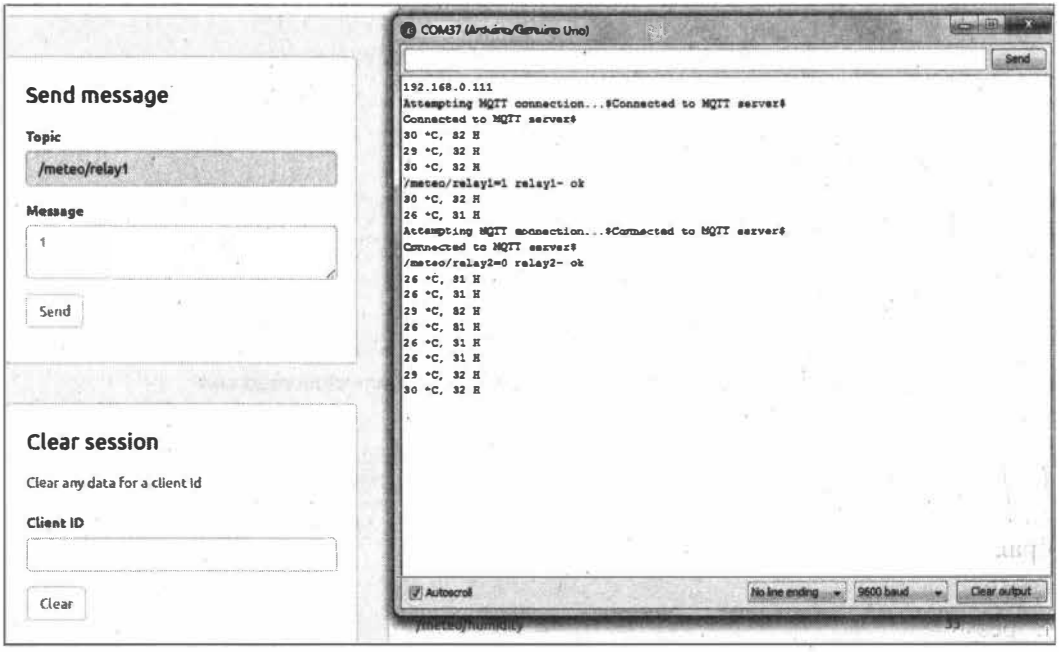


Рис. 4.7. Получение команд от брокера

на Android в качестве пульта управления, а также для отображения информации с датчиков, подключенных к плате Arduino (например, с домашней метеостанции). Приложение представляет собой готовый MQTT-клиент с небольшим количеством очень удобных виджетов.

Итак, скачиваем приложение на смартфон, создаем новое соединение и прописываем в нем данные из нашего профиля на **www.cloudmqtt.com** (рис. 4.8).

Рис. 4.8. Создание соединения

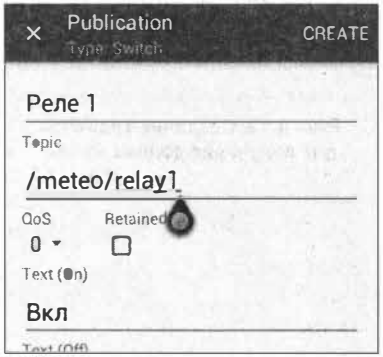


Рис. 4.9. Создание виджетов (Switch) для отправки данных в темы: этап 1

Выбираем созданное соединение и на вкладке **PUBLISH** создаем виджеты (Switch) для отправки данных в темы: `/meteo/relay1` и `/meteo/relay2` (рис. 4.9–4.11).

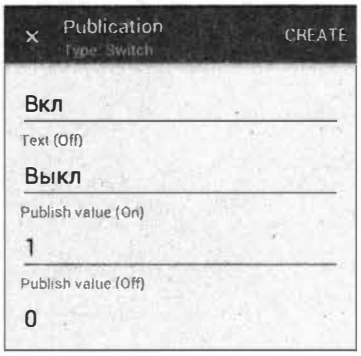


Рис. 4.10. Создание виджетов (Switch) для отправки данных в темы: этап 2

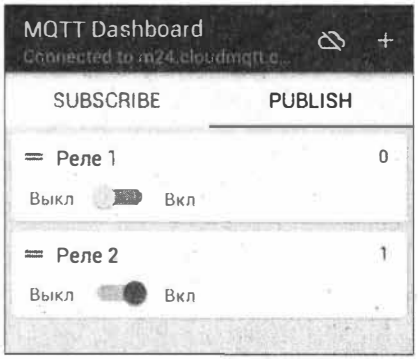


Рис. 4.11. Создание виджетов (Switch) для отправки данных в темы: этап 3

Приложение позволяет также настроить отображение данных, которые отправляются брокеру. Для этого на вкладке **SUBSCRIBE** мы создаем виджеты для отображения температуры и влажности, где необходимо подписаться на топики `/meteo/humidity` и `/meteo/temperature` (рис. 4.12).

В результате мы получим на экране смартфона табло с отображением данных (рис. 4.13).

Рис. 4.12. Создание виджетов для получения данных из тем

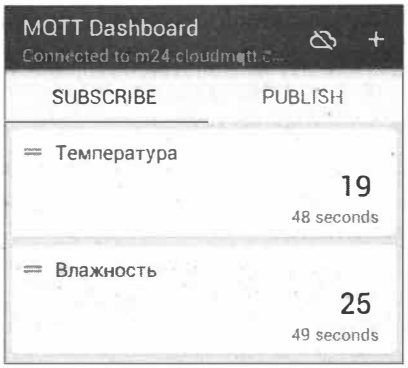
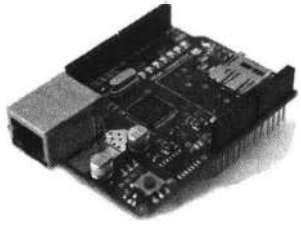


Рис. 4.13. Отображение на экране смартфона данных температуры и влажности, отправленных с датчика DHT11 в сервис www.cloudmqtt.com

ГЛАВА 5



Облачные платформы для устройств Интернета вещей

5.1. Arduino IoT Cloud

Arduino IoT Cloud — платформа для IoT-приложений. Она позволяет просто и быстро разрабатывать и развертывать приложения для различных устройств.

Мы уже давно научились получать данные от датчиков, подключенных к официальной или совместимой плате Arduino, загружать их в облачные сервисы и наблюдать за результатами через удобный веб-интерфейс. Но при этом нам приходится полагаться на сторонние сервисы — такие, как ThingSpeak, Adafruit.io, Amazon Web Services и другие. В 2019 году компания Arduino объявила, что бета-версия их собственного облачного сервиса Arduino IoT становится общедоступной. С запуском Arduino IoT Cloud Arduino теперь предоставляет миллионам своих пользователей полный комплексный подход к Интернету вещей, который включает аппаратное обеспечение, встроенное ПО, облачные сервисы и знания.

5.1.1. Регистрация и подключение устройств к Arduino IoT Cloud

Для начала необходимо создать учетную запись или войти в Arduino IoT Cloud на странице **https://create.arduino.cc/iot** (рис. 5.1).

В зависимости от того, чего хочет достичь пользователь, для приложения IoT потребуется несколько основных компонентов:

- устройства для сбора данных или управления чем-либо;
- программное обеспечение для определения поведения оборудования;
- облачное приложение для хранения данных или удаленного управления оборудованием.

Устройства — это аппаратное обеспечение (аппаратная плата), которое запускает программное обеспечение, считывает датчики, управляет исполнительными механизмами и связывается с Arduino IoT Cloud.

Для начала необходимо подключить и сконфигурировать свою плату. Перейдите на вкладку **Devices** и в открывшемся окне (рис. 5.2) выберите **Set up an Arduino**

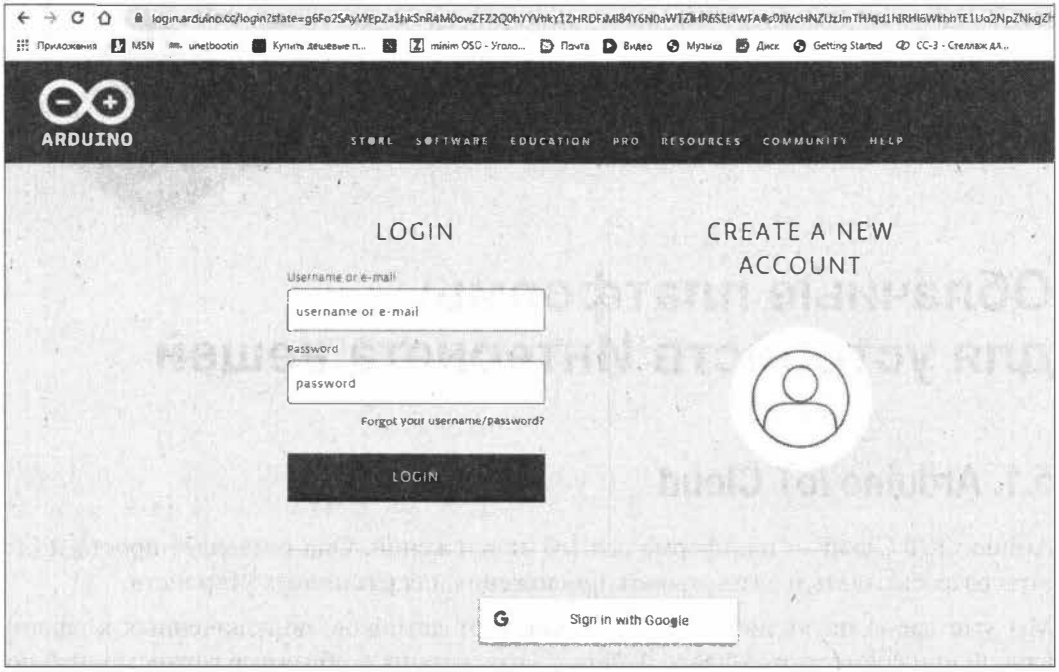


Рис. 5.1. Страница входа в Arduino IoT Cloud

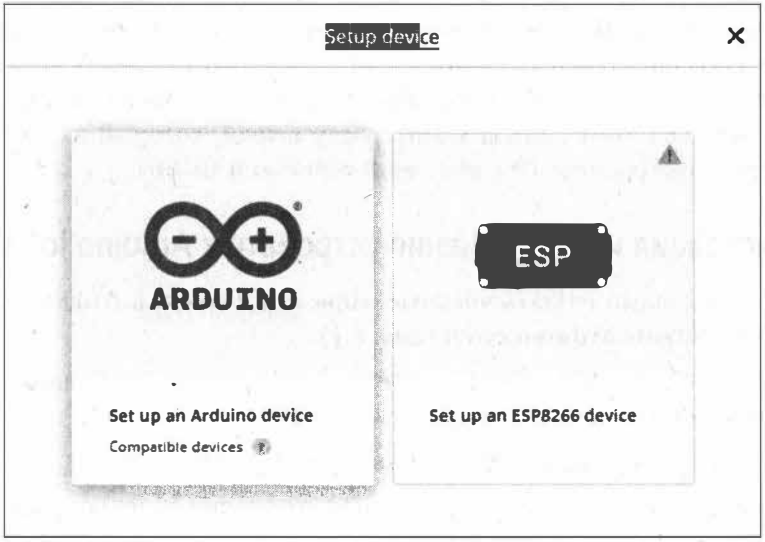


Рис. 5.2. Выбор платы в Arduino IoT Cloud

device. На момент подготовки книги поддерживались платы Arduino MKR1000, Arduino MKR1010, Arduino Nano 33 IoT.

Далее потребуется скачать и установить плагин Arduino Create Agent (рис. 5.3).

После установки Arduino Create Agent подключите плату Arduino к компьютеру по USB. Через некоторое время плата будет определена (рис. 5.4).

Рис. 5.3. Требование установить плагин Arduino Create Agent

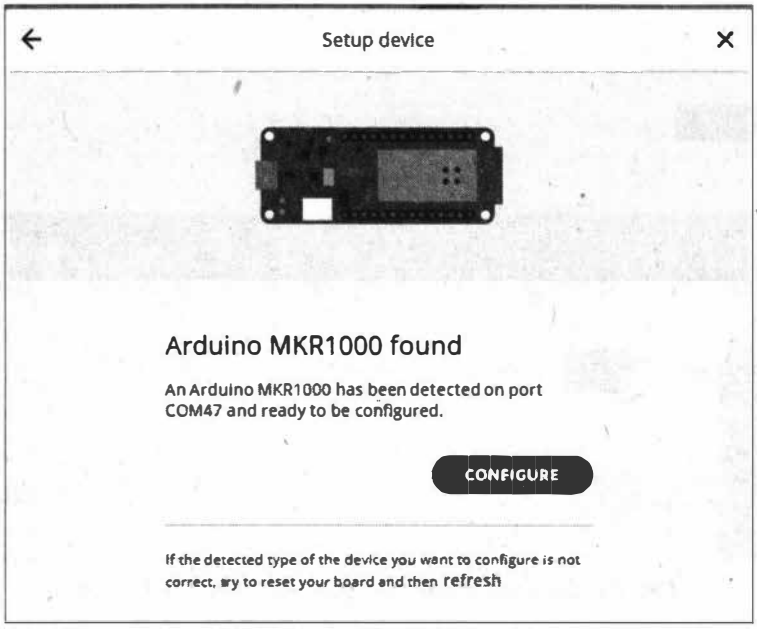


Рис. 5.4. Плата Arduino MKR1000 определена

Далее плату необходимо сконфигурировать — нажмите на кнопку **Configure**, и она появится в списке **Devices** (рис. 5.5).

Теперь мы можем создавать свои IoT-устройства — перейдите на вкладку **Things** и нажмите на кнопку **Add New Thing** (рис. 5.6).

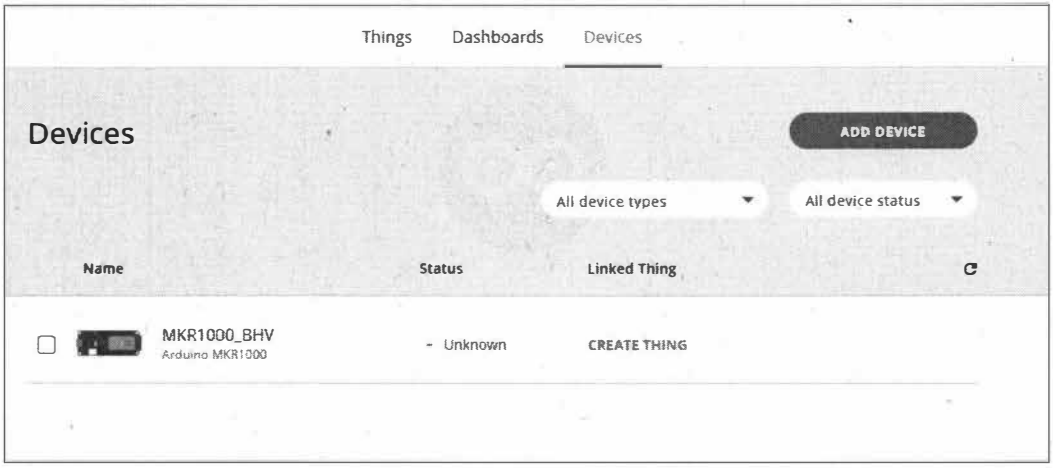


Рис. 5.5. Ваша плата в списке **Devices**

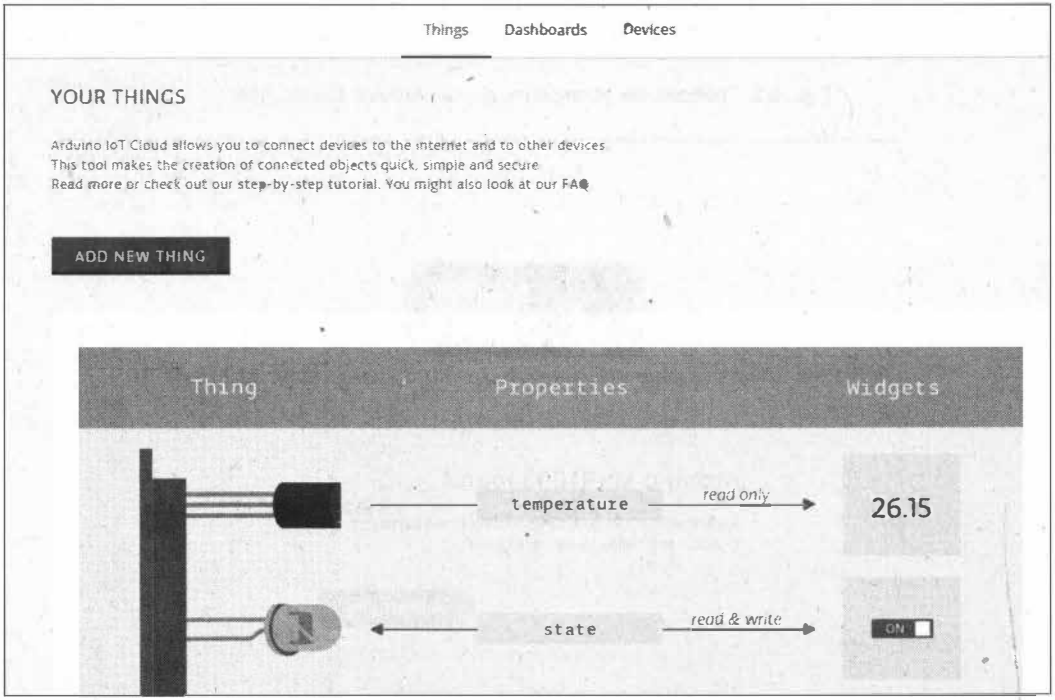


Рис. 5.6. Добавление нового устройства в Arduino IoT Cloud

Введите название устройства (здесь: `ArduinoMKR1000_BHV01`), выберите плату, на основе которой оно будет создано (здесь: **MKR 1000**), и нажмите на кнопку **Create** (рис. 5.7).

Каждое устройство имеет набор свойств **Properties**. Что такое *свойство* и как их добавлять, мы рассмотрим в следующем разделе.

Рис. 5.7. Выбор названия и платы для устройства в Arduino IoT Cloud

5.1.2. Отправка данных с устройства в Arduino IoT Cloud

Создадим на основе платы Arduino MKR1000 устройство, которое будет отправлять в облачный сервис Arduino IoT Cloud данные с потенциометра. Монтажная схема соединений показана на рис. 5.8.

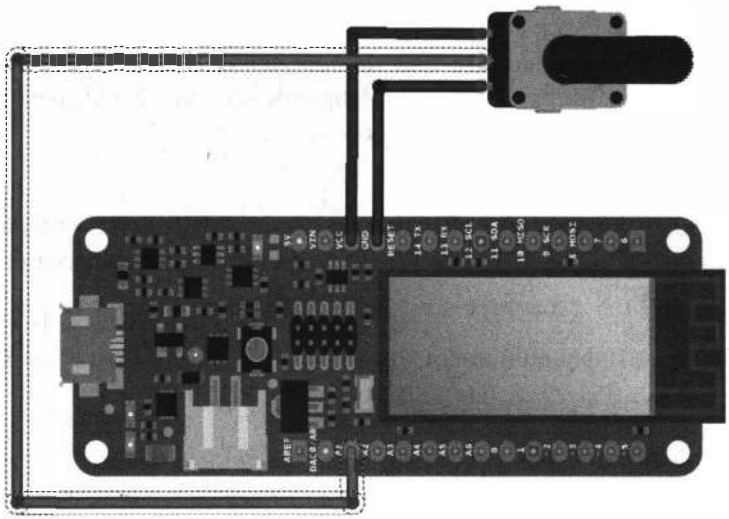


Рис. 5.8. Монтажная схема подключения потенциометра к плате MKR1000

Рис. 5.9. Устройство ArduinoMKR1000_BHV01 в Arduino IoT Cloud

Теперь перейдите в Arduino IoT Cloud и откройте ранее созданное устройство **ArduinoMKR1000_BHV01** (рис. 5.9).

Настало время задать *свойства* устройства, которые определяют его параметры. Это могут быть данные датчиков устройства или состояния подключенных к плате актуаторов. Для рассматриваемого случая добавьте свойство, определяющее данные подключенного к плате Arduino потенциометра, — нажмите на кнопку **Add property** (см. рис. 5.9), задайте его данные (рис. 5.10) и нажмите на кнопку **Add property** — свойство для устройства добавлено (рис. 5.11).

Переходим теперь к разработке кода — нажмите на кнопку **Edit sketch**, и откроется страница Arduino Create, где уже сгенерирован шаблон скетча для нашего устройства.

Внесите на вкладке **Secret** данные для подключения платы Arduno к сети Wi-Fi (рис. 5.12) и перейдите на вкладку **thingProperties.h** (рис. 5.13), уже содержащую заготовку кода. Рассмотрим этот код подробнее:

```
#include <ArduinoIoTCloud.h>
```

Приведенная строка импортирует библиотеку ArduinoIoTCloud, которая необходима для синхронизации локальных переменных скетча с их свойствами в IoT Cloud.

```
const char THING_ID[] = "4bae94ca-81c3-428a-899c-36ef564cc604";
```

Это уникальный идентификационный код `THING_ID[]` устройства в IoT Cloud.

```
#include <Arduino_ConnectionHandler.h>
const char SSID[]     = SECRET_SSID;
const char PASS[]     = SECRET_PASS;
```

Для управления подключением Wi-Fi используется Wi-Fi Connection Manager. Значения SSID сети и пароля берутся из вкладки **Secret**.

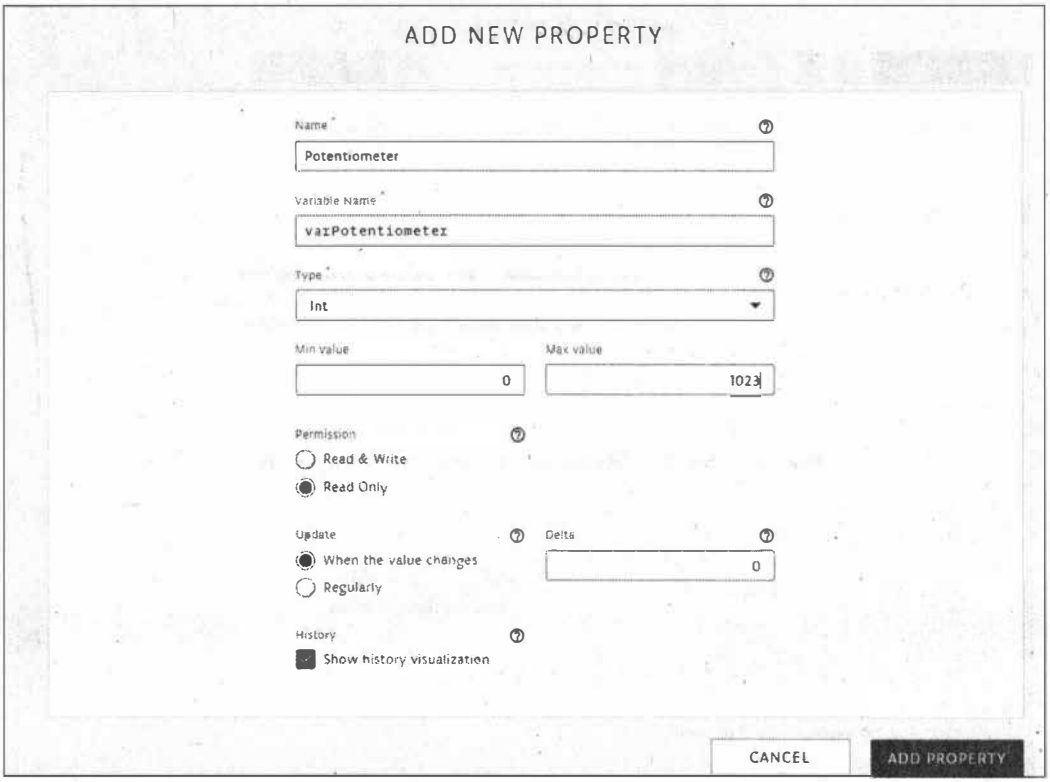


Рис. 5.10. Создание свойства Potentiometer для устройства ArduinoMKR1000_BHV01

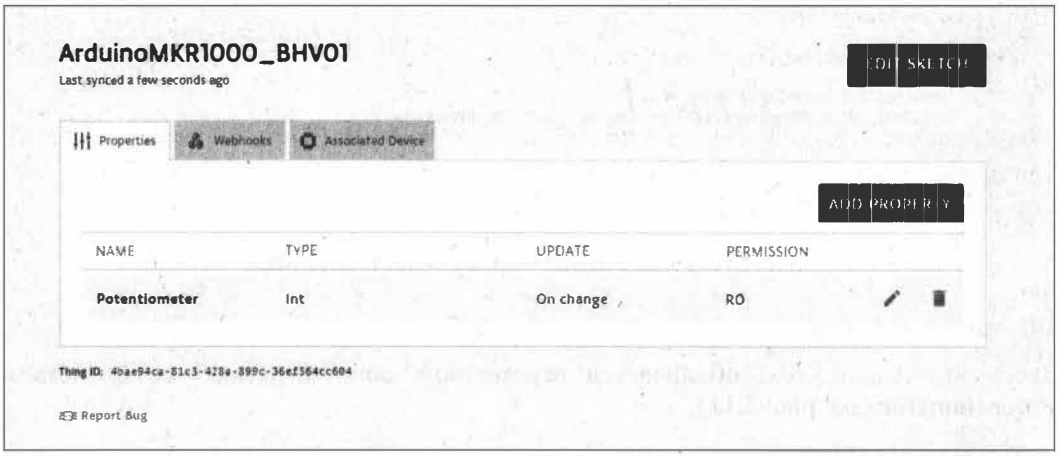


Рис. 5.11. Устройство ArduinoMKR1000_BHV01 с добавленным свойством Potentiometer

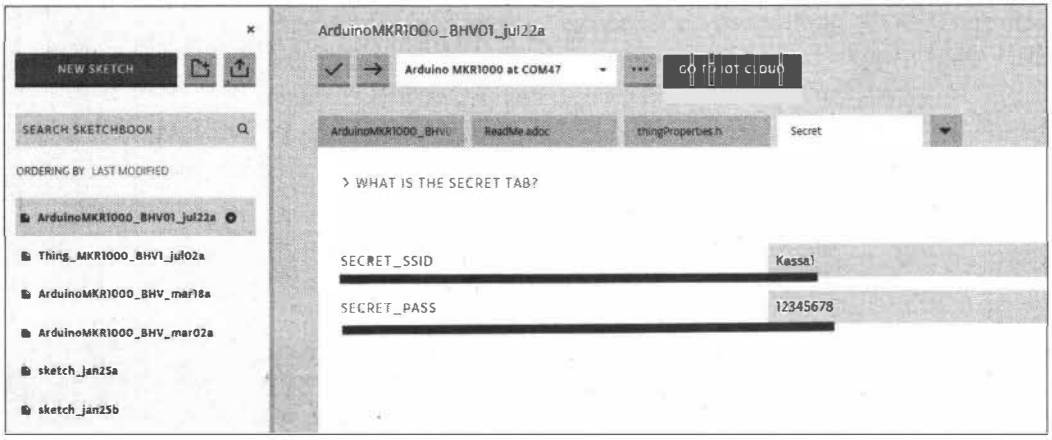


Рис. 5.12. Вкладка **Secret**: данные для подключения к Wi-Fi

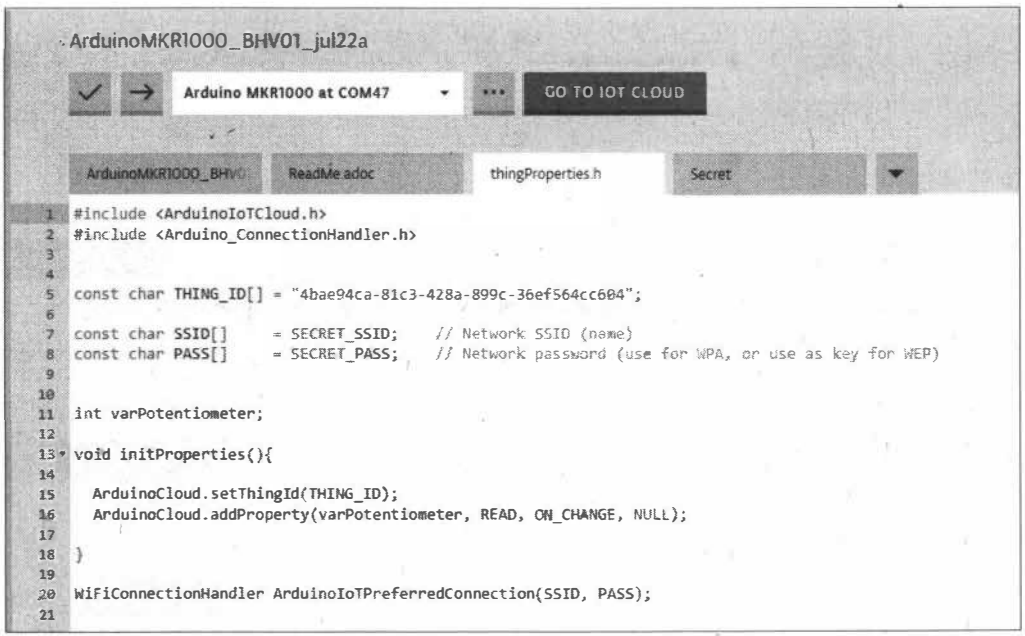


Рис. 5.13. Вкладка **thingProperties.h** с заготовкой кода

```
int varPotentiometer;
```

Здесь `varPotentiometer` объявляется переменной, сопоставленной со свойством **Potentiometer** (см. рис. 5.11).

```
void initProperties(){
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(varPotentiometer, READ, ON_CHANGE, NULL);
}
```

Это функция инициализации свойств устройства. Она сообщает, к какому устройству подсоединиться (`THING_ID`) и как обрабатывать переменную `varPotentiometer`.

```
WiFiConnectionHandler ArduinoIoTPreferredConnection(ssid, pass);
```

Эта строка инициализирует диспетчер соединений, используя имя точки доступа Wi-Fi (`SECRET_SSID`) и пароль (`SECRET_PASS`), которые мы установили на вкладке **Secret**.

Перейдите на основную вкладку скетча (рис. 5.14). Здесь в функции `setup()` мы запускаем монитор последовательного порта:

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

Инициализируем свойства, определенные на вкладке **thingProperties.h**:

```
initProperties();
```

Соединяемся с Arduino Iot Cloud, используя Connection Manager:

```
ArduinoCloud.begin(ArduinoIoTPreferredConnection);
```

Рис. 5.14. Основная вкладка скетча

В процедуре `loop()` считываем данные с потенциометра каждые 500 мс и отправляем их для отладки в монитор последовательного порта:

```
if (millis() - lastConnectionTime > 500) {
  varPotentiometer = analogRead(A1);
```

```
    Serial.println(Pot);
    Serial.println(varPotentiometer);
}
```

Далее функция

```
ArduinoCloud.update();
```

включает синхронизацию значений свойств между облаком и платой, проверку соединения в сети и облаке и другую логику.

Загрузите скетч в плату, откройте монитор последовательного порта, и вы увидите вывод показаний с потенциометра (рис. 5.15).

Рис. 5.15. Скетч загружен, данные отправляются

В Arduino IoT Cloud можно для устройства создать страницу визуализации данных, поступающих с устройства, или страницу отправки данных на устройство.

Вернитесь в Arduino IoT Cloud и перейдите на вкладку **Dashboard**, где можно добавлять на страницу виджеты (рис. 5.16).

Для отображения на виджете значений свойства необходимо привязать к виджету соответствующую свойству переменную — нажмите на кнопку **Link Property** и выберите нужную переменную (рис. 5.17).

На рис. 5.18 показана веб-страница, на двух виджетах отображающая данные, поступающие с потенциометра.

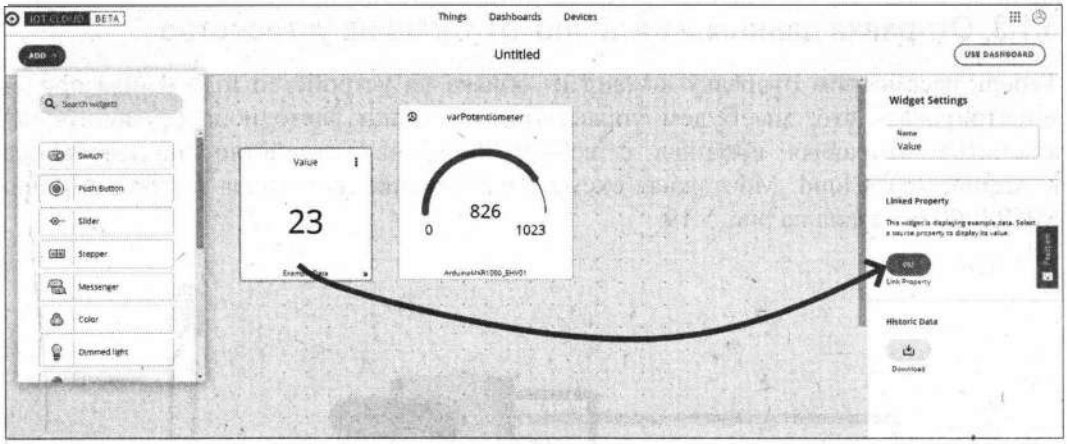


Рис. 5.16. Вкладка Dashboard: добавление виджетов

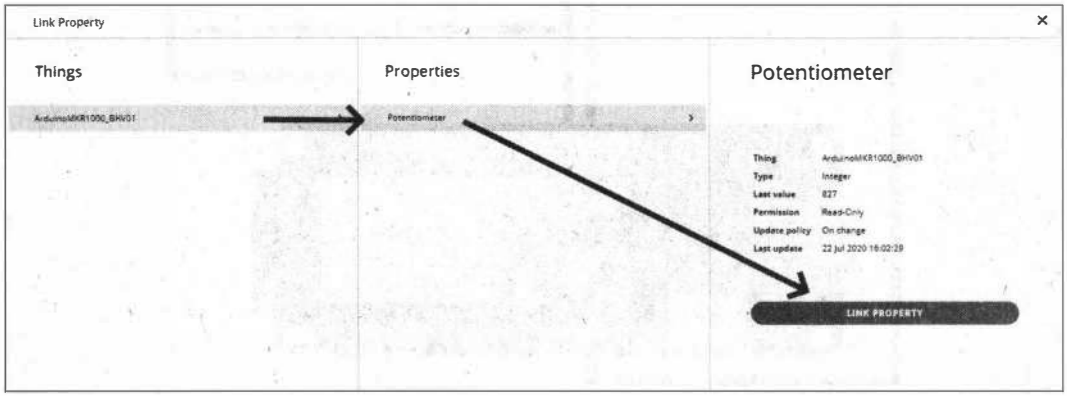


Рис. 5.17. Привязка переменной к виджету

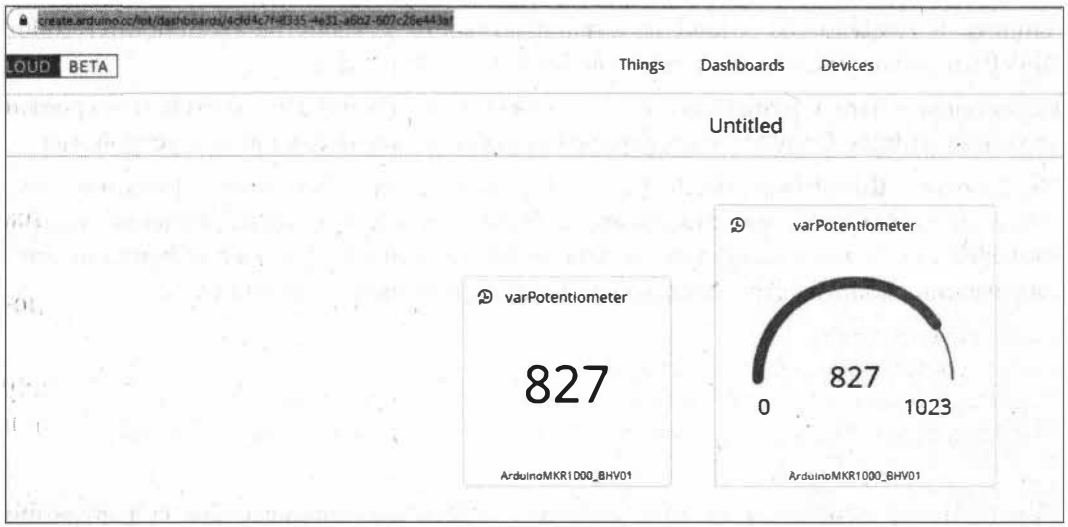


Рис. 5.18. Веб-страница, отображающая на двух виджетах данные, поступающие с потенциометра

5.1.3. Отправка данных из Arduino IoT Cloud на устройство

Теперь рассмотрим отправку команд из облака на устройство IoT. Чтобы продемонстрировать это, мы будем управлять состоянием светодиода (включить/выключить), отправляя команды с веб-страницы на устройство, подключенное к Arduino IoT Cloud. Монтажная схема подключения светодиода к плате Arduino MKR1000 показана на рис. 5.19.

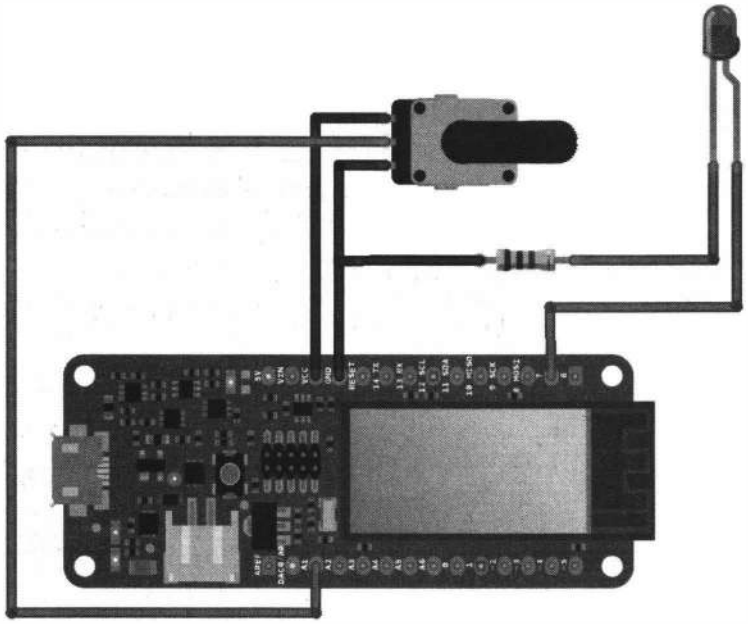


Рис. 5.19. Монтажная схема подключения светодиода к плате MKR1000

Зайдите в Arduino IoT Cloud на страницу своего устройства ArduinoMKR1000_BHV01 и добавьте ему новое свойство **led** (рис. 5.20 и 5.21).

Переходим теперь к разработке кода — нажмите на кнопку **Edit sketch**, и откроется страница Arduino Create, где сгенерирован шаблон скетча для нашего устройства.

На вкладке **thingProperties.h** (рис. 5.13) добавится объявление функции `void onVarLedChange()`, которая станет вызываться каждый раз, когда значение нашего свойства `led` будет изменяться на панели инструментов. А также в функции инициализации свойств устройства добавится инициализация свойства `led`:

```
void initProperties(){
  ArduinoCloud.setThingId(THING_ID);
  ArduinoCloud.addProperty(varPotentiometer, READ, ON_CHANGE, NULL);
  ArduinoCloud.addProperty(varLed, READWRITE, ON_CHANGE, onVarLedChange);
}
```

Перейдите на основную вкладку скетча — здесь мы прописываем код функции `onVarLedChange()`:

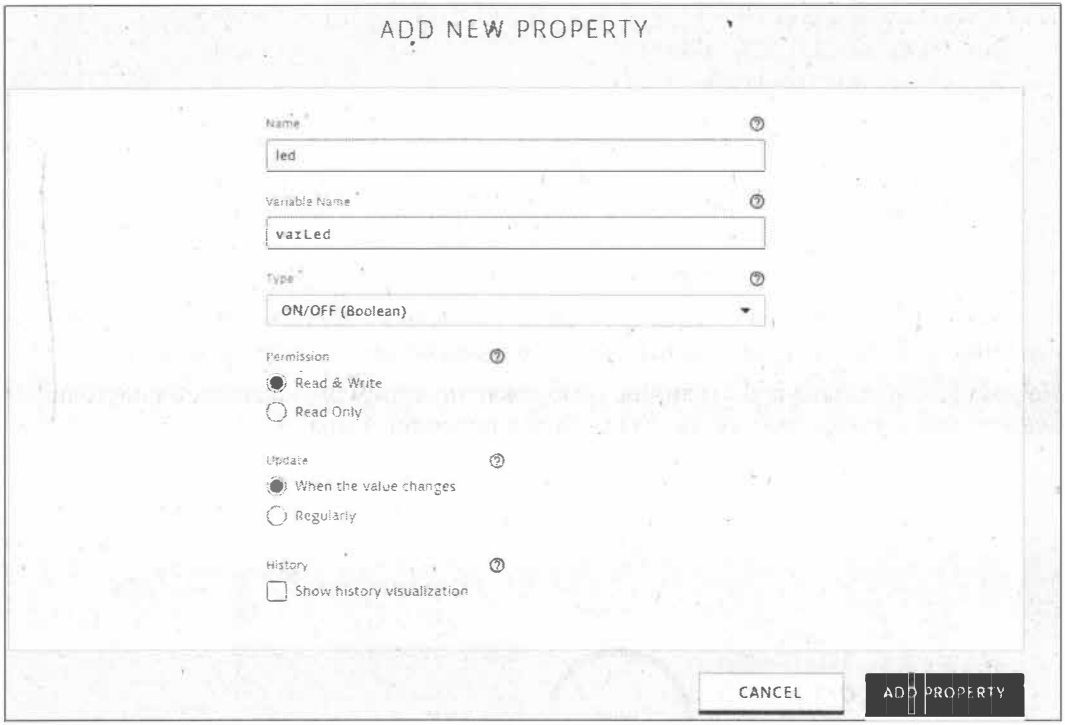


Рис. 5.20. Создание свойства **led** для устройства ArduinoMKR1000_BHV01: шаг 1

Рис. 5.21. Создание свойства **led** для устройства ArduinoMKR1000_BHV01: шаг 2

```
void onVarLedChange() {
    digitalWrite(LED_PIN, light);
    Serial.print("The light is ");
    if (light) {
        Serial.println("ON");
    } else {
        Serial.println("OFF");
    }
}
```

Вернитесь в Arduino IoT Cloud, перейдите на вкладку **Dashboard** и на своей веб-странице добавьте виджет для переключения состояния светодиода (рис. 5.22).

На рис. 5.23 показана веб-страница, отображающая элемент управления состоянием светодиода, а также данные, поступающие с потенциометра.

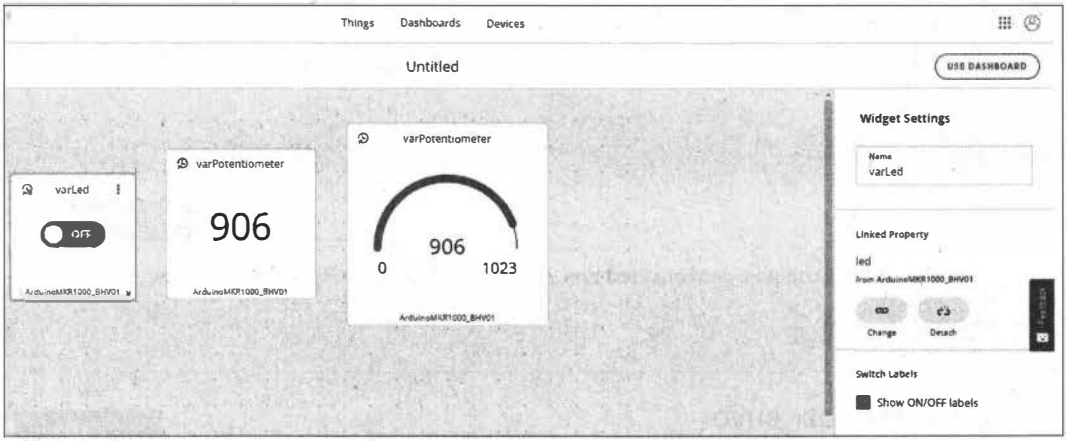


Рис. 5.22. Добавление виджета для переключения светодиода

Рис. 5.23. Веб-страница, отображающая элемент управления состоянием светодиода, а также данные, поступающие с потенциометра

5.2. Сервис «Народный мониторинг»

На сервисе «Народный мониторинг» собираются и отображаются на карте мира практически в реальном времени показания от различных датчиков среды, установленных как на улице, так и в помещениях.

Чтобы стать участником проекта, необходимо зарегистрироваться. Зайдите на сайт **http://www.narodmon.ru** и выберите пункт меню **Вход | Стать участником проекта**. В открывшейся форме введите адрес электронной почты, куда будут отправлены логин и пароль для входа в профиль (рис. 5.24).

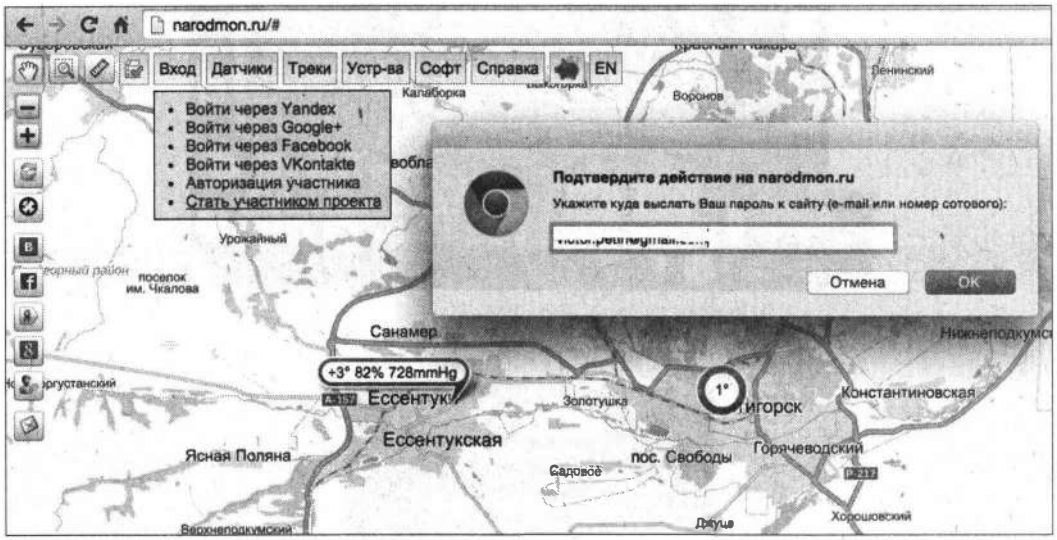


Рис. 5.24. Регистрация в сервисе «Народный мониторинг»

После регистрации можно добавлять на карту сервиса «Народный мониторинг» свои устройства.

В качестве устройства мы воспользуемся домашней метеостанцией, построенной на плате Arduino+WiFi и датчиках BMP280 и DHT11. Для добавления такой домашней метеостанции на карту сервиса «Народный мониторинг» необходимо подключить ее к сети Интернет и настроить передачу показаний датчиков на сайт **narodmon.ru** с интервалом от 5 до 15 минут.

5.2.1. Подготовка платы Arduino+WiFi

Контроллером метеостанции, как уже было отмечено ранее, будет служить плата Arduino+WiFi от компании RobotDyn (рис. 5.25), в которой интегрированы плата Arduino Uno R3 и модуль Wi-Fi ESP8266. Оба эти компонента платы могут работать вместе или каждый в отдельности. Необходимый режим работы платы можно установить с помощью находящихся на ней переключателей (рис. 5.26). Плата Arduino+WiFi представляет собой решение, весьма удобное для разработки новых

проектов, требующих возможностей платы Arduino Uno и наличия Wi-Fi. Встроенный в плату конвертер USB-Serial CH340G позволяет обновлять через порт USB скетчи и прошивки как для ATmega328, так и для ESP8266.

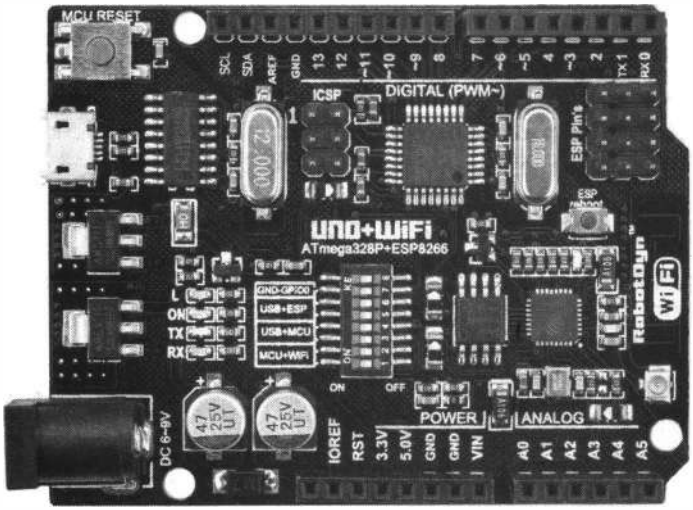


Рис. 5.25. Плата Arduino+WiFi от компании RobotDyn

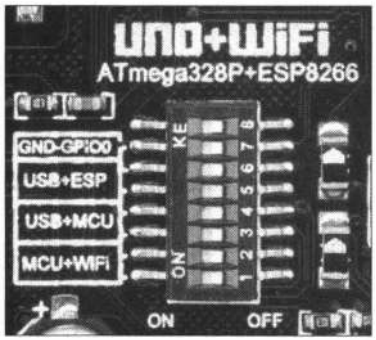


Рис. 5.26. Переключатели режимов работы на плате Arduino+WiFi

В табл. 5.1 представлены все возможные режимы работы платы и показаны все возможные положения переключателей на ней (переключатель 8 не используется).

***Таблица 5.1.** Режимы работы платы Arduino+WiFi*

Режим	Переключатель						
	1	2	3	4	5	6	7
ATmega328 ←→ ESP8266	ON	ON	OFF	OFF	OFF	OFF	OFF
USB ←→ ATmega328	OFF	OFF	ON	ON	OFF	OFF	OFF
USB ←→ ESP8266 (обновление прошивки или эскиз)	OFF	OFF	OFF	OFF	ON	ON	ON

Таблица 5.1 (окончание)

Режим	Переключатель						
	1	2	3	4	5	6	7
USB ←→ ESP8266 (сообщение)	OFF	OFF	OFF	OFF	ON	ON	OFF
Все независимые	OFF	OFF	OFF	OFF	OFF	OFF	OFF

Как видно из таблицы, если установить переключатели 3 и 4 в положение ON, а остальные — в положение OFF, то мы получим обычную Arduino Uno.

Монтажная схема соединений домашней метеостанции на датчиках BMP280 (температура и атмосферное давление) и DHT11 (влажность) показана на рис. 5.27.

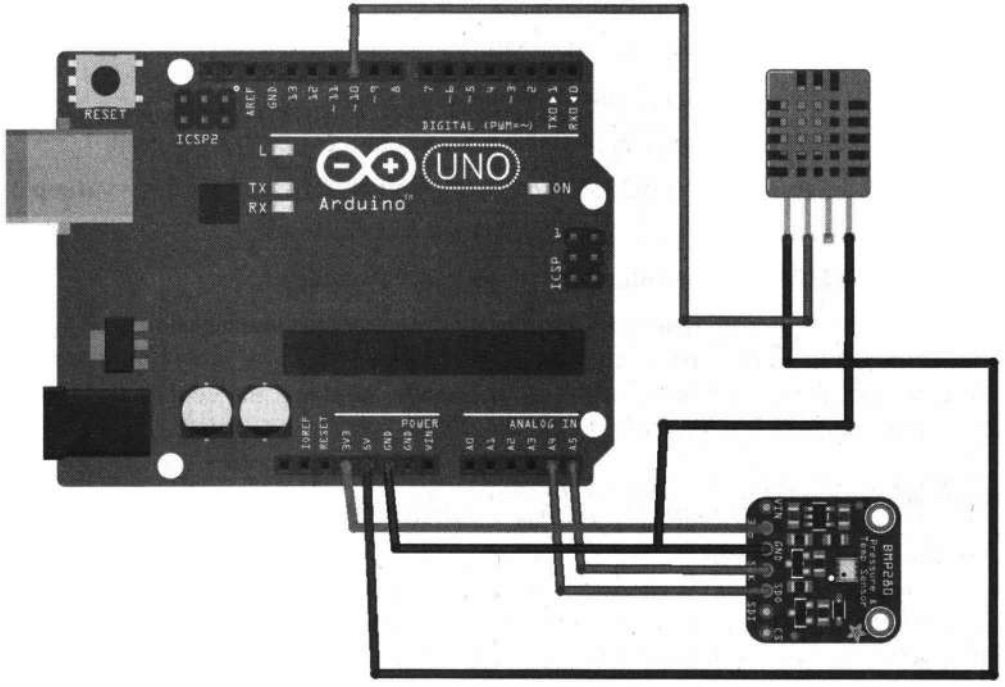


Рис. 5.27. Монтажная схема соединений для метеостанции на датчиках BMP280 и DHT11

5.2.2. Подключение устройства к сервису «Народный мониторинг»

Как уже отмечалось ранее, для добавления домашней метеостанции на карту сервиса «Народный мониторинг» надо подключить ее к сети Интернет и организовать передачу показаний датчиков на сайт **narodmon.ru** с интервалом от 5 до 15 минут. Однако сначала необходимо настроить передачу и отображение на карте сервиса показаний всех датчиков устройства.

Для передачи данных на сайт мы воспользуемся самым простым протоколом передачи данных — HTTP GET. Для этого необходимо подключить модуль ESP8266 по

Wi-Fi к сети Интернет и обратиться к сервису «Народный мониторинг» по адресу следующего вида:

http://narodmon.ru/get?ID=MAC&mac1=value1&...&macN=valueN

где:

- ID — уникальный адрес своего устройства;
- **mac1, ... macnN** — названия датчиков устройства;
- **value1, ... valueN** — показания датчиков.

В качестве **ID** передадим число, отражающее географические координаты нашей метеостанции, например: **ID= 440365430747** (широта: 44.0365, долгота: 43.0747).

Для названия датчиков используем следующие значения:

- **H1** — для относительной влажности (датчик DHT11);
- **T1** — для температуры (датчик BMP280);
- **P1** — для атмосферного давления (датчик BMP280).

Тогда адрес отправки данных для нашей метеостанции будет следующий:

http://narodmon.ru/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep

Строку

/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep

модуль ESP8266 будет получать из Arduino по последовательному порту. Для активации устройства используем произвольные данные. Скетч получения модулем данных из последовательного порта и отправки данных на сервис «Народный мониторинг» приведен в листинге 5.1.

Листинг 5.1

```
// подключение библиотек
#include <ESP8266WiFi.h>
// данные SSID и пароль точки доступа
const char* ssid     = "Kassa1";
const char* password = "12345678";
// создание объекта клиента
WiFiClient client;
 // данные, пришедшие из последовательного порта
String inputString = "";
String inputString1 = "";
// строка пришла
boolean stringComplete = false;
// сервер Народного мониторинга
char server[] = "www.narodmon.ru";
// массив для получения данных от сервера
char response[40];
```

```
void setup(void){
  delay(5000);
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  Serial.print("*");

  // Подсоединение к точке доступа
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop(void){
    serialEvent1();
    if (stringComplete) {
       // отправить данные на сайт
       sendDataNarodmon(inputString);
       // очистить строку
       inputString = "";
       stringComplete = false;
    }
}
// получение строки по Serial
void serialEvent1() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    Serial.write(inChar);
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}

// отправка данных на сайт Народного мониторинга
void sendDataNarodmon(String str) {
```

```
  if (client.connect(server, 80)) {
     //// отправка данных на сервер narodmon.ru
     Serial.print(str);
     client.println("GET "+str+" HTTP/1.1");
     client.println("Host: www.narodmon.ru");
     client.println("Connection: close");
     client.println();
     // получение ответа
     unsigned long previos=millis();
     for(int i=0;i<40;i++)
        response[i]=0;
     int x=0;int f=0;
     do{
        if(client.available() > 0) {
           // получать данные из ESP8266
           char s = client.read();
           if(s=='#')
              f=1;
           if(f==1) {
              response[x]=s;
              Serial.print(response[x]);
              x++;
           }
        Serial.write(s);
        }
     }
     while((millis() - previos) < 5000);
     Serial.println(response);
     client.stop();
  }
  else {
    // нет соединения
    Serial.println("connection failed");
    client.stop();
  }

}
```

Электронный архив

Скетч, соответствующий листингу 5.1, можно найти в папке *examples\05\05_01* сопровождающего книгу электронного архива (см. *приложение*).

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

и загрузить скетч в модуль ESP8266.

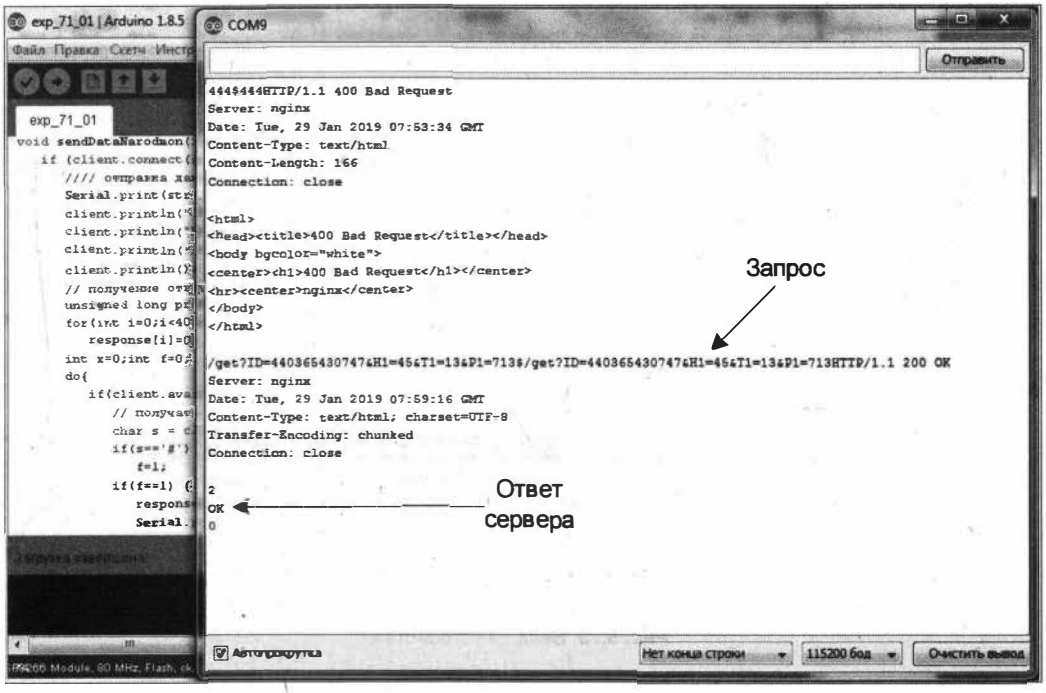


Рис. 5.28. Получение данных по последовательному порту и отправка на сайт «Народного мониторинга»

Откройте теперь монитор последовательного порта и попробуйте отправлять данные (рис. 5.28).

Как можно видеть, при отправке строки

```
/get?ID=440365430747&H1=45&T1=13&P1=713$/get?ID=440365430747&H1=45&T1=13&P1=713
```

приходит ответ **OK**. Значит, данные успешно отправлены.

Переходим к следующему этапу — добавлению устройства в свой профиль для отображения на карте «Народного мониторинга».

Зайдите на сайт **http://narodmon.ru**, авторизуйтесь и откройте пункт меню **Датчики | Мои Датчики | Добавить устройство**, где необходимо ввести ID твоего устройства (рис. 5.29).

В открывшемся окне (рис. 5.30) выберите тип данных и название для каждого из датчиков, установите доступ к показаниям для каждого датчика (публичный или приватный) и выполните привязку к карте, указав полный адрес его размещения или геокоординаты. После этого устройство появится на карте (рис. 5.31).

> ***Внимание!***
>
> Установка публичного доступа возможна только для уличных датчиков.

В следующем эксперименте мы организуем отправку на сервис «Народный мониторинг» реальных данных домашней метеостанции.

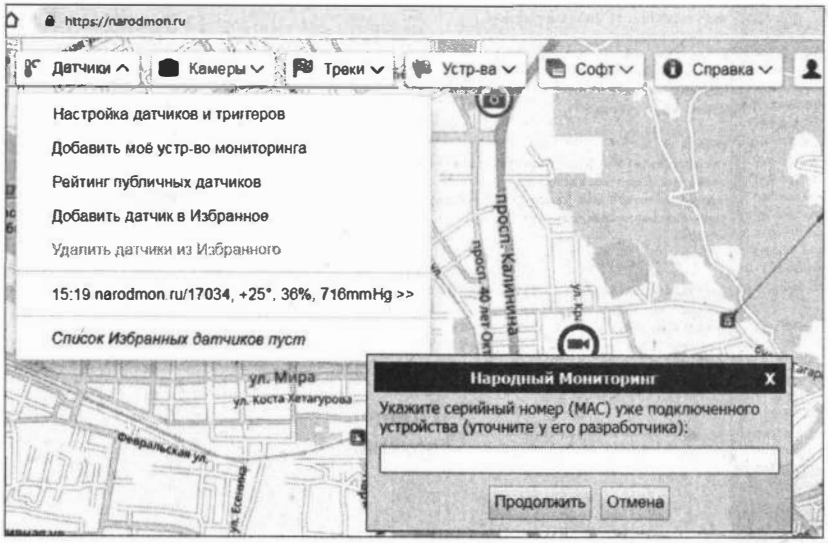


Рис. 5.29. Ввод ID устройства

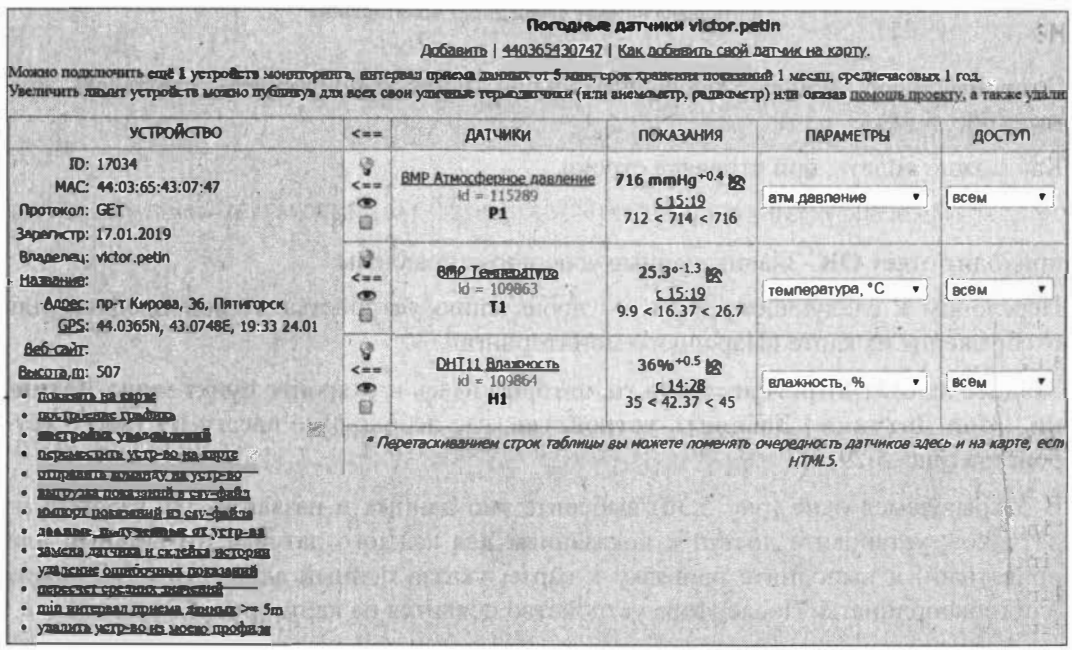


Рис. 5.30. Редактирование данных устройства

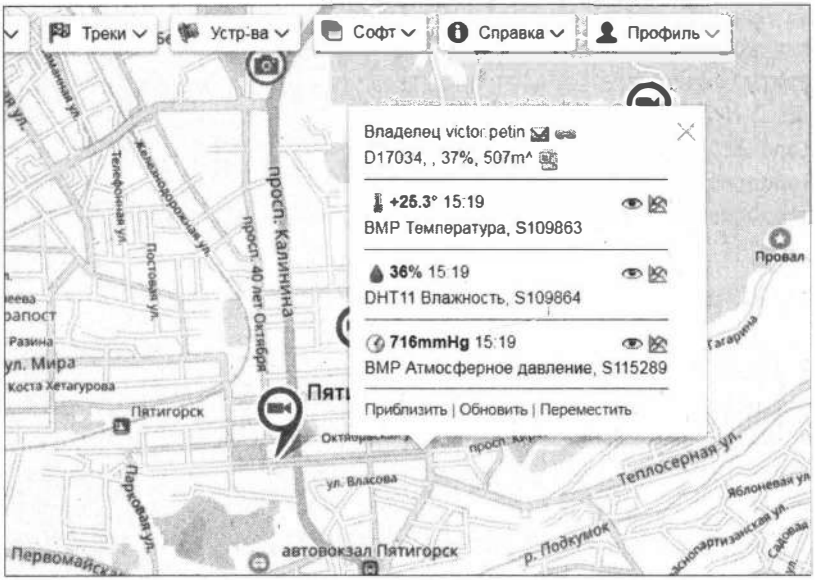


Рис. 5.31. Устройство на карте «Народного мониторинга»

5.2.3. Отправка данных датчиков домашней метеостанции на сервис «Народный мониторинг»

Соберите схему, показанную на рис. 5.27, и загрузите в Arduino скетч (листинг 5.2), который будет каждые 5 минут получать показания с датчиков BMP280 и DHT11, формировать строку вида

/get?ID=440365430747&H1=valueh&T1=valuet&P1=valuep*

и отправлять эту строку по последовательному порту в модуль ESP8266. Для получения данных с датчиков BMP280 и DHT11 нам понадобится подключить Arduino-библиотеки Adafruit_BMP280 и DHT, а также и другие библиотеки, упомянутые в начальном разделе листинга 5.2.

Листинг 5.2

```
// подключение библиотек
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
// пин для подключения датчика DHT
#define DHTPIN 10
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
```

```
// создание экземпляров
Adafruit_BMP280 bmp;
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millissend=0;
unsigned long pauseseconds=10;
// ID - устройства в Народном мониторинге
String IDstr="441369430175";

void setup()
{
  // подключение последовательного порта
  Serial.begin(115200);
  // запуск датчиков DHT,BMP280
  dht.begin();
  bmp.begin();
}

void loop() {
  // ждем время паузы
  if(millis()-millissend>=1000) {
     pauseseconds--;
     //Serial.print("time to send - ");Serial.println(pauseseconds);
     if(pauseseconds==0) {
        // получение данных
        int h = dht.readHumidity();
        int t = bmp.readTemperature();
        float p = bmp.readPressure()/133.32;
        // формирование строки
        String str="/get?ID="+IDstr;
        str=str+"&H1="+String(h);
        str=str+="&T1="+String(t);
        str=str+="&P1="+String(p);
        str=str+="*";
        // отправка в последовательный порт
        Serial.print(str);
        //
        pauseseconds=300;
        }

     millissend=millis();
  }
}
```

Электронный архив

Скетч, соответствующий листингу 5.2, можно найти в папке *examples\05\05_02* сопровождающего книгу электронного архива (см. *приложение*).

Электронный архив

Библиотеки DHT, Adafruit_Sensor и Adafruit_BMP280 размещены в каталоге *libraries* сопровождающего книгу электронного архива (см. *приложение*).

Переключатели на плате Arduino+WiFi необходимо установить следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

и загрузить скетч в плату Arduino.

Откройте монитор последовательного порта, и вы увидите отправку в последовательный порт данных каждые 5 минут (первая отправка через 10 секунд) (рис. 5.32).

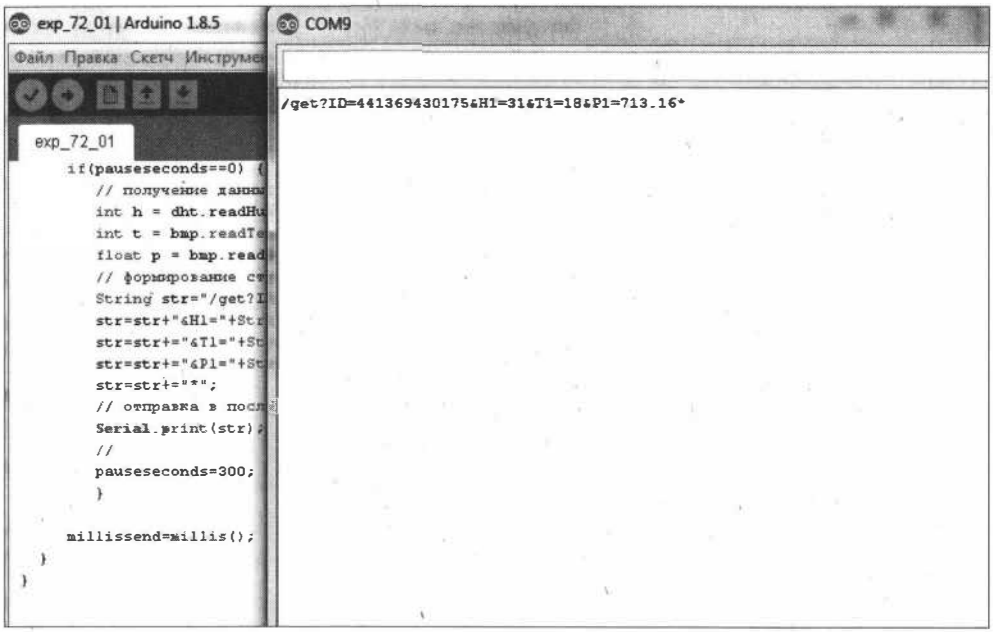


Рис. 5.32. Отправка данных по последовательному порту

Далее надо перевести плату в режим ATmega328 ←→ ESP8266, для чего переключатели необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту.

Через некоторое время проверьте отправку данных на сервис «Народный мониторинг». Для этого зайдите на сайте сервиса в меню **Профиль | Мои Датчики** и в открывшемся окне перейдите по ссылке **данные полученные от устройства** (рис. 5.33).

Полученные показания датчиков / Latest Sensors Readings, UTC+3, find=''

```
2019-01-30 11:26:35  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.50&P1=716.65
2019-01-30 11:21:30  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.00&P1=716.59
2019-01-30 11:16:24  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.10&P1=716.65
2019-01-30 11:11:19  185.75.6.102  GET  ID=440365430747&H1=45&T1=12.30&P1=716.68
2019-01-30 11:06:13  185.75.6.102  GET  ID=440365430747&H1=45&T1=11.90&P1=716.71
2019-01-30 11:01:08  185.75.6.102  GET  ID=440365430747&H1=45&T1=11.70&P1=716.80
```

[Вывести данные за предыдущий час >>]

Рис. 5.33. Данные, полученные от устройства

Рис. 5.34. Отображение данных, полученных от устройства, на карте сервиса «Народный мониторинг»

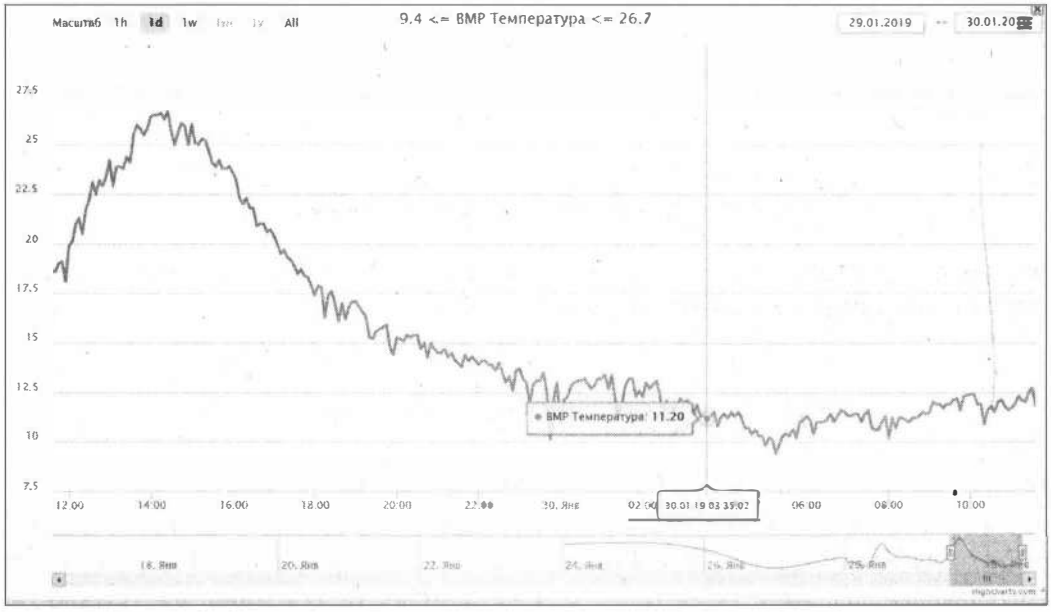


Рис. 5.35. График данных датчика температуры

Данные отображаются и на карте (рис. 5.34), вы также можете посмотреть и графики по каждому датчику (рис. 5.35).

5.2.4. Прием на устройстве команд, отправленных из сервиса «Народный мониторинг»

Сервис «Народный мониторинг» предоставляет возможность отправлять данные с его сайта на устройство, что позволяет подключить к домашней метеостанции с помощью реле исполнительные устройства и управлять ими через Интернет.

Сначала определим список команд, которые будем отправлять с сервиса «Народный мониторинг», например:

- **relay1=1** — включить реле 1;
- **relay1=0** — выключить реле 1;
- **relay2=1** — включить реле 2;
- **relay2=0** — выключить реле 2.

Отправить эти команды можно из своего профиля (пункт меню **Профиль | Мои датчики**), щелкнув на ссылке **Отправить команду на устр-во**. Откроется панель, в которую введите нужную команду (рис. 5.36). Команда появится в очереди на исполнение (рис. 5.37), и при очередном получении данных от датчиков будет отправлена как ответ сервера.

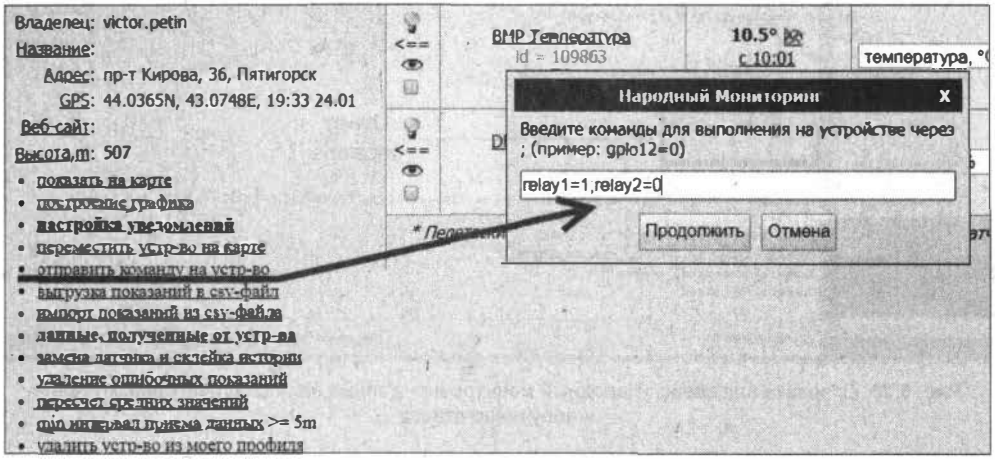


Рис. 5.36. Ввод списка команд для отправки на устройство

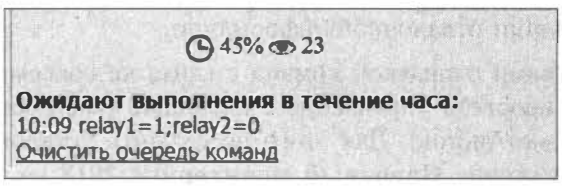


Рис. 5.37. Очередь команд

Проверим получение ответа от сервера при отправке данных. Для этого установите переключатели на плате Arduino+WiFi следующим образом:

1	2	3	4	5	6	7
OFF	OFF	OFF	OFF	ON	ON	ON

и загрузите в модуль ESP8266 скетч из листинга 5.3.

На сервисе «Народный мониторинг» введите список команд для отправки на устройство (см. рис. 5.36), откройте монитор последовательного порта и попробуйте отправлять данные. Наберите строку

```
/get?ID=440365430747&H1=45&T1=13&P1=713$
```

При этом на сервер отправляются введенные нами данные, а мы смотрим ответ сервера в мониторе последовательного порта (рис. 5.38).

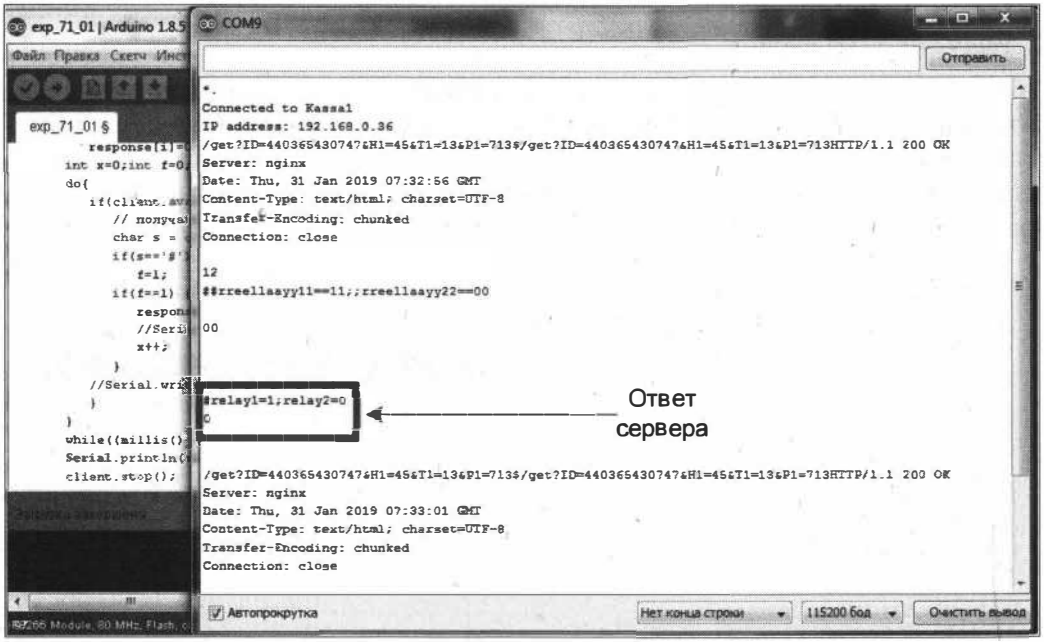


Рис. 5.38. Отправка на сервис «Народный мониторинг» данных по последовательному порту и получение ответа

Этот ответ от сервера, добавив символ `'$'`, мы будем отправлять по последовательному порту на Arduino. В скетче листинга 5.1 при этом небходимо убрать вывод в последовательный порт отладочной информации.

Управлять устройствами отправкой команд с сайта не совсем удобно. К счастью, есть возможность упростить управление с помощью смартфона или планшета на операционной системе Android. Для этого необходить установить на свой смартфон (планшет) приложение **Народный мониторинг 2018** из Google Play Маркет (рис. 5.39).

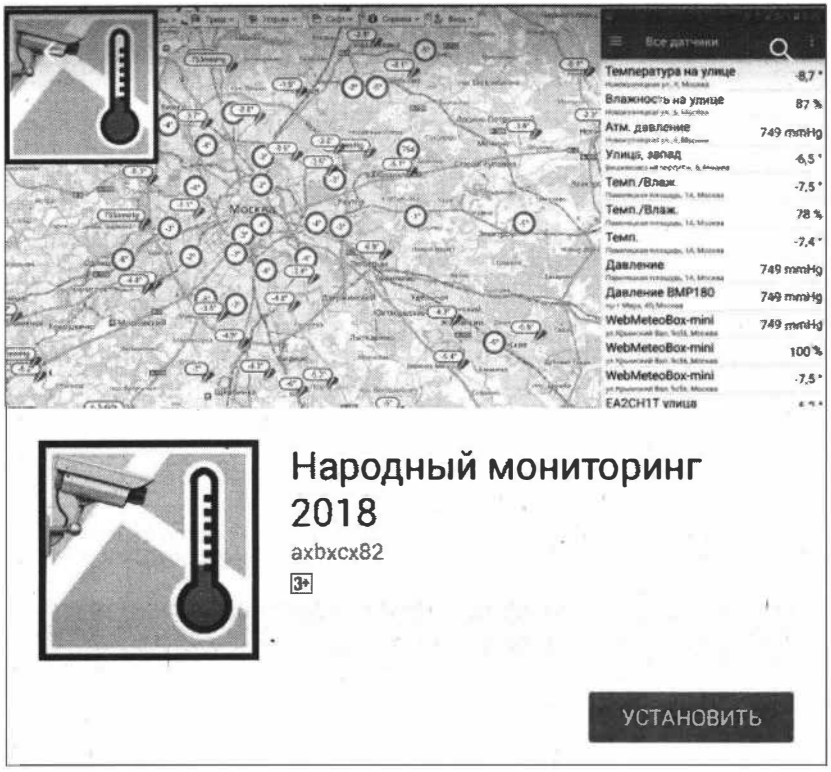


Рис. 5.39. Приложение **Народный мониторинг 2018** в Google Play Маркет

Установив приложение, запустите его, авторизуйтесь с данными своего профиля на сервисе «Народный мониторинг», и через некоторое время вы получите доступ к данным датчиков своей домашней метеостанции (рис. 5.40 и 5.41).

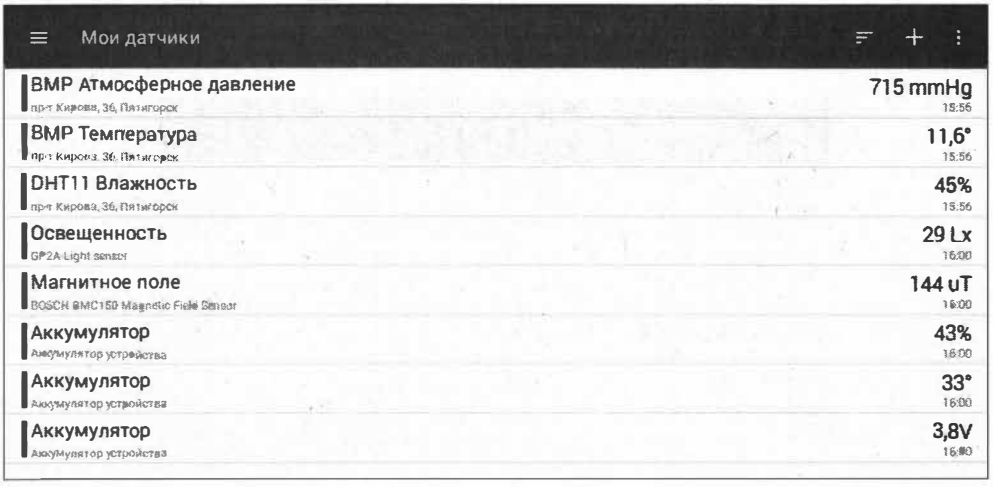


Рис. 5.40. Данные домашней метеостанции в мобильном приложении Народный мониторинг 2018

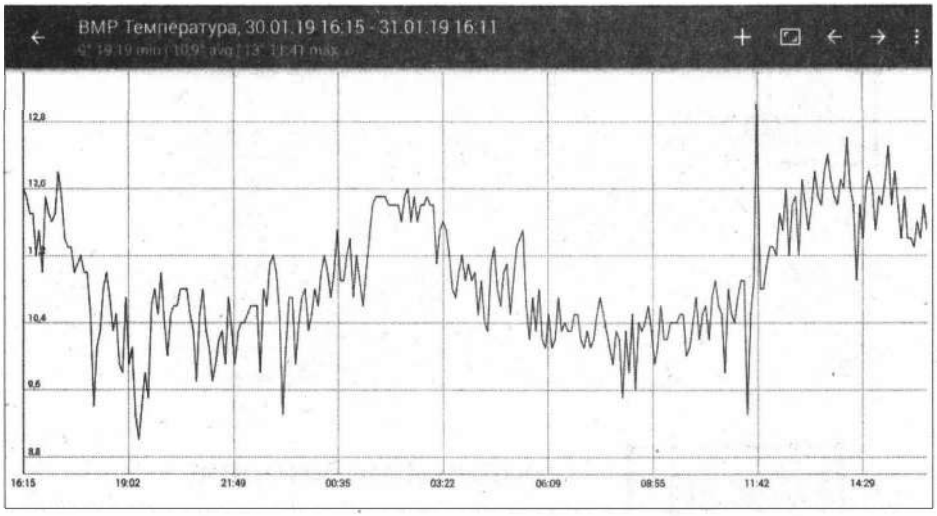


Рис. 5.41. График данных температуры с домашней метеостанции в мобильном приложении Народный мониторинг 2018

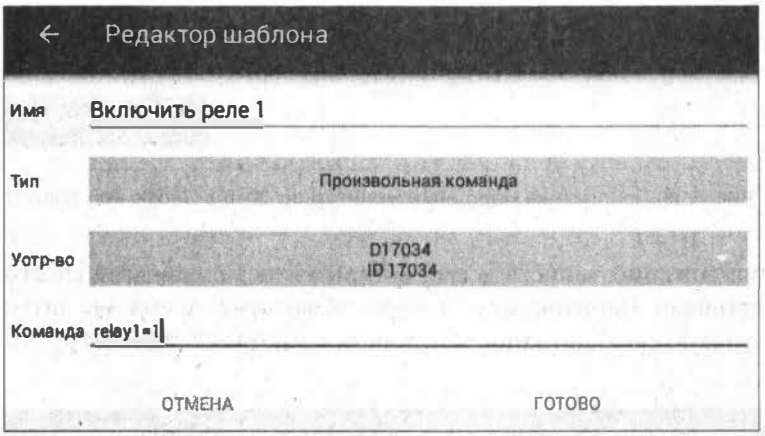


Рис. 5.42. Создание кнопок интерфейса для отправки команд: шаг 1

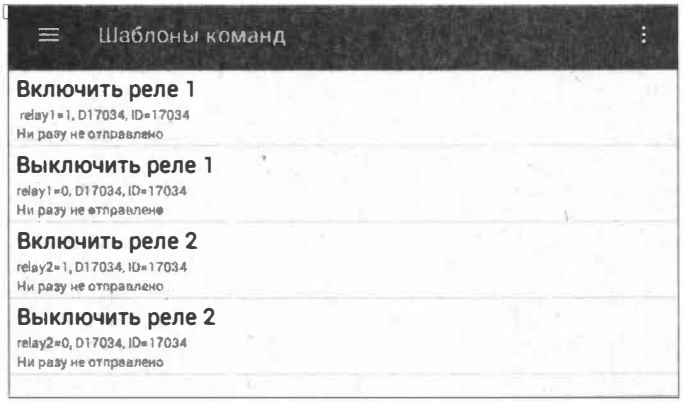


Рис. 5.43. Создание кнопок интерфейса для отправки команд: шаг 2

Рассмотрим, как организовать в приложении отправку команд управления. Выберите пункт меню **Управление** и создайте кнопки для отправки команд (рис. 5.42 и 5.43). Теперь вы сможете управлять контактами реле домашней метеостанции со смартфона.

5.2.5. Обработка и исполнение команд, полученных от сервиса «Народный мониторинг»

Добавим в домашнюю метеостанцию модуль управления реле Relay Shield, к которому можно потом подключить исполнительные устройства — например, вентилятор или лампу освещения. Монтажная схема соединений показана на рис. 5.44.

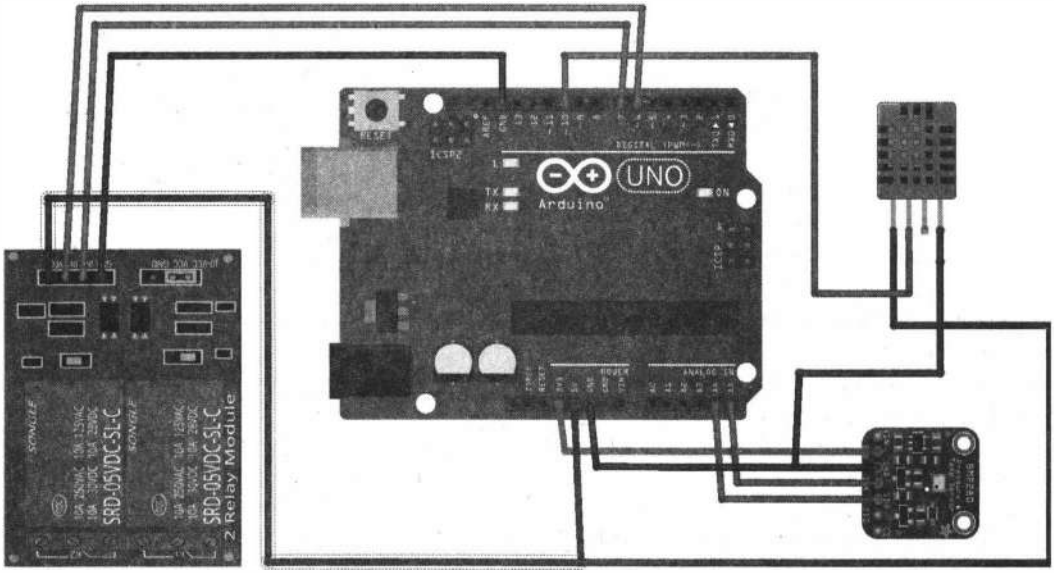


Рис. 5.44. Монтажная схема соединений для метеостанции на датчиках BMP280 и DHT11 с добавлением модуля управления реле Relay Shield

При этом на Arduino необходимо организовать прием и анализ команд, приходящих по последовательному порту. Добавим в строковую переменную `inputString` скетча (листинг 5.3) сбор приходящих команд. По получении символа `'$'` строка считается скомплектованой, и вызывается функция `command()` для поиска ожидаемых команд:

- **relay1=1** — включить реле 1;
- **relay1=0** — выключить реле 1;
- **relay2=1** — включить реле 2;
- **relay2=0** — выключить реле 2.

Листинг 5.3

```
// подключение библиотек
#include "DHT.h"
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BMP280.h>
// пин для подключения датчика DHT
#define DHTPIN 10
// тип датчика DHT
#define DHTTYPE DHT11   // DHT 11
// создание экземпляров
Adafruit_BMP280 bmp;
DHT dht(DHTPIN, DHTTYPE);
// для опроса
unsigned long millissend=0;
unsigned long pauseseconds=10;
// ID - устройства в Народном мониторинге
String IDstr="441369430175";
// контакты подключения реле
const int pinRelay1=7;
const int pinRelay2=6;
// данные, пришедшие из последовательного порта
String inputString = "";
// строка пришла
boolean stringComplete = false;

void setup()
{
  // подключение последовательного порта
  Serial.begin(115200);
   // конфигурация пинов
   pinMode(pinRelay1,OUTPUT);
   pinMode(pinRelay2,OUTPUT);
   // начальная установка
   digitalWrite(pinRelay1,HIGH);
   digitalWrite(pinRelay2,HIGH);

  // запуск датчиков DHT,BMP280
  dht.begin();
  bmp.begin();
}

void loop() {
    // проверка прихода строки из последовательного порта
    if (stringComplete) {
       command();
```

```
      // очистить строку
      inputString = "";
      stringComplete = false;
    }

  // ждем время паузы
  if(millis()-millissend>=1000) {
     pauseseconds--;
     if(pauseseconds==0) {
        // получение данных
        int h = dht.readHumidity();
        int t = bmp.readTemperature();
        float p = bmp.readPressure()/133.32;
        // формирование строки
        String str="/get?ID="+IDstr;
        str=str+"&H1="+String(h);
        str=str+="&T1="+String(t);
        str=str+="&P1="+String(p);
        str=str+="*";
        // отправка в последовательный порт
        Serial.print(str);
        //
        pauseseconds=300;
        }

     millissend=millis();
  }
}
// получение данных по последовательному порту
void serialEvent() {
  boolean flag1=false;

  while (Serial.available() && flag1==false) {
    // получить байт:
    char inChar = (char)Serial.read();
    if (inChar == '$') {
      stringComplete = true;
      flag1=true;
    }
    else   // добавление в строку
      inputString += inChar;
  }
}
// проверка пришедших с сервера команд
void command() {
     //
    if(inputString.indexOf("relay1=1")!=-1) {
       digitalWrite(pinRelay1,LOW);
    }
```

```
    if(inputString.indexOf("relay1=0")!=-1) {
       digitalWrite(pinRelay1,HIGH);
    }
    if(inputString.indexOf("relay2=1")!=-1) {
       digitalWrite(pinRelay2,LOW);
    }
    if(inputString.indexOf("relay2=0")!=-1) {
       digitalWrite(pinRelay2,HIGH);
    }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.3, можно найти в папке examples\05\05_03 сопровождающего книгу электронного архива (см. *приложение*).

Установите переключатели на плате Arduino+WiFi следующим образом:

1	2	3	4	5	6	7
OFF	OFF	ON	ON	OFF	OFF	OFF

и загрузите скетч в Arduino.

Откройте монитор последовательного порта, где можно видеть отправку в последовательный порт данных каждые 5 минут. Попробуйте теперь отправлять команды управления реле (рис. 5.45).

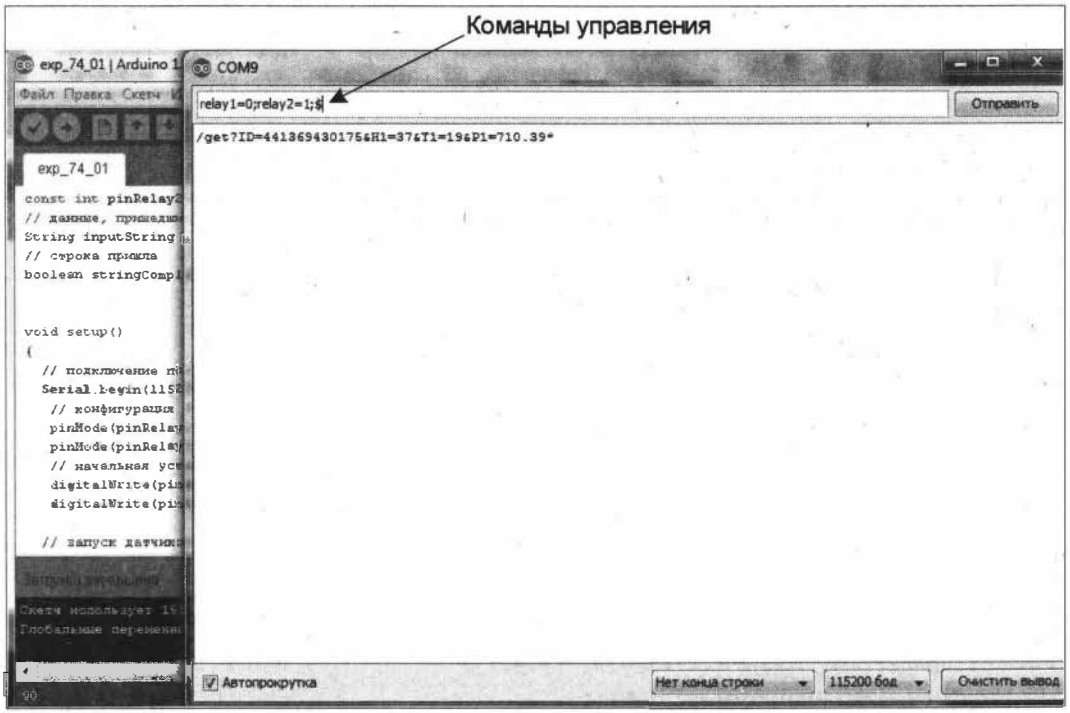


Рис. 5.45. Получение команд управления по последовательному порту

Далее надо перевести плату в режим ATmega328 ←→ ESP8266. Переключатели для этого необходимо установить следующим образом:

1	2	3	4	5	6	7
ON	ON	OFF	OFF	OFF	OFF	OFF

Теперь Arduino и ESP8266 соединены по последовательному порту. Вы можете попробовать отправлять команды управления реле с сервиса «Народный мониторинг» и из мобильного приложения Народный мониторинг 2018.

5.3. Сервис ThingSpeak

Сервис ThingSpeak (**https://thingspeak.com**) — открытая платформа данных для проектов Internet of Things. ThingSpeak обеспечивает сбор данных с датчиков в реальном времени, обработку этих данных, их визуализацию и использование данных в приложениях и плагинах. Перед началом работы с ThingSpeak необходимо зарегистрироваться, нажав в его стартовом окне на кнопку **Get Started For Free**. Чтобы

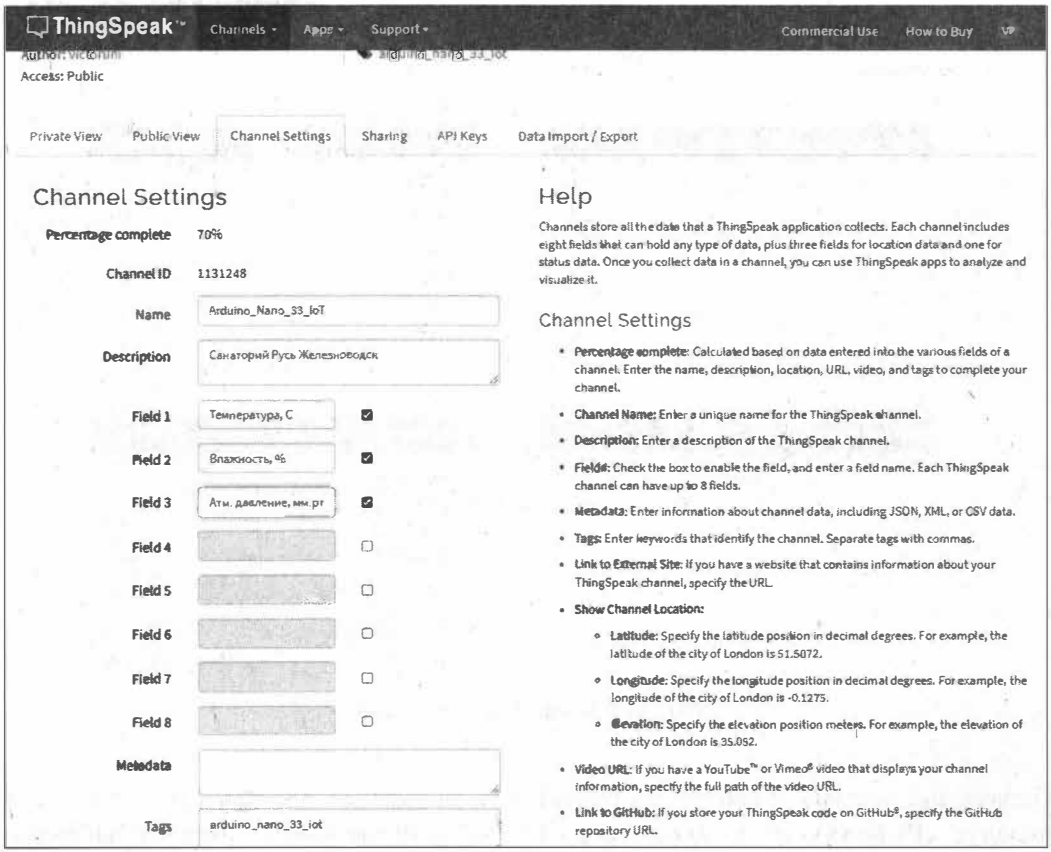


Рис. 5.46. Заполнение полей канала ThingSpeak

использовать ThingSpeak, вы должны войти в свою существующую учетную запись или создать новую. Некоммерческие пользователи могут использовать ThingSpeak бесплатно, однако бесплатные аккаунты предлагают ограничения на определенные функции.

Зарегистрированный пользователь ThingSpeak должен создать для себя канал (Channel), где будут храниться его данные. Каждый канал включает восемь полей для любого типа данных, три поля местоположения и одно поле состояния. Таким образом, один канал можно использовать для отправки и хранения данных с одного устройства, имеющего не более восьми датчиков. Для создания канала нажмите на кнопку **New**, заполните поля открывшейся формы (рис. 5.46) и сохраните канал, нажав на кнопку **Save Channel**. В результате канал будет создан (рис. 5.47).

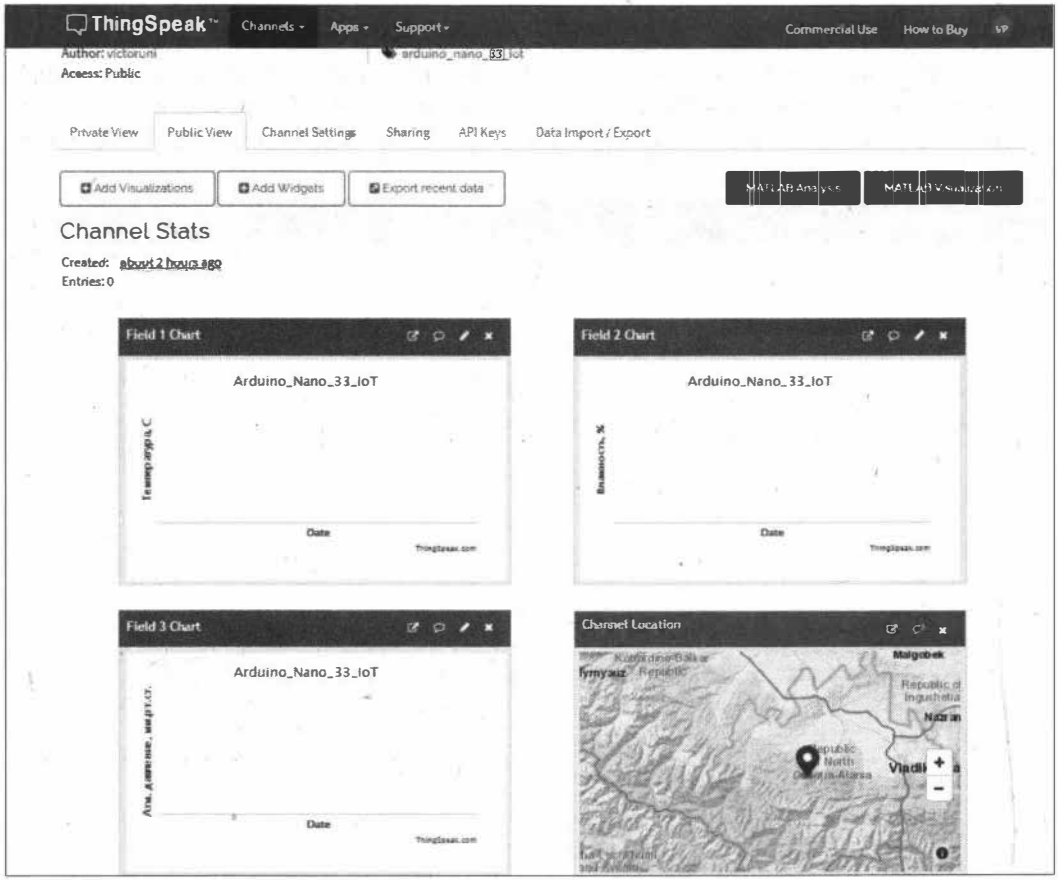


Рис. 5.47. Канал ThingSpeak создан

Параметры доступа к каналу настраиваются на вкладке **Sharing** (рис. 5.48). А на вкладке **API Keys** (рис. 5.49) находятся ключи, которые необходимы для настройки отправки и получения данных.

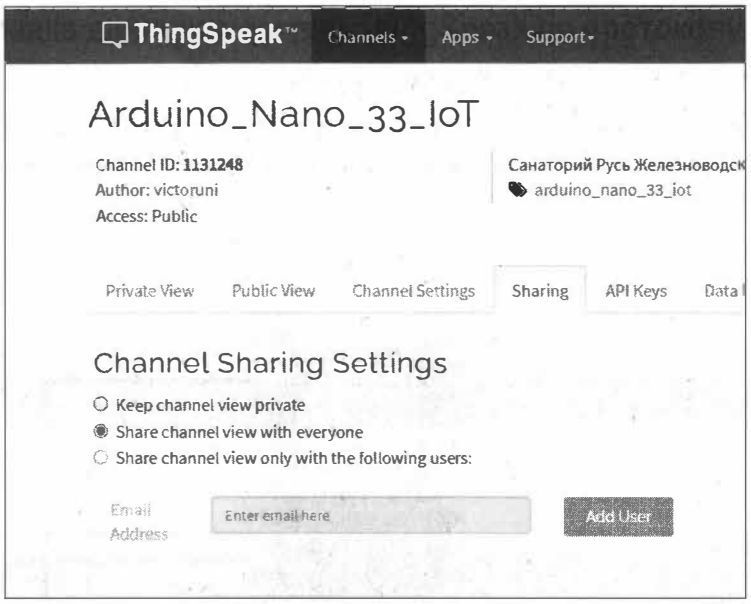


Рис. 5.48. Настройка доступа к каналу ThingSpeak

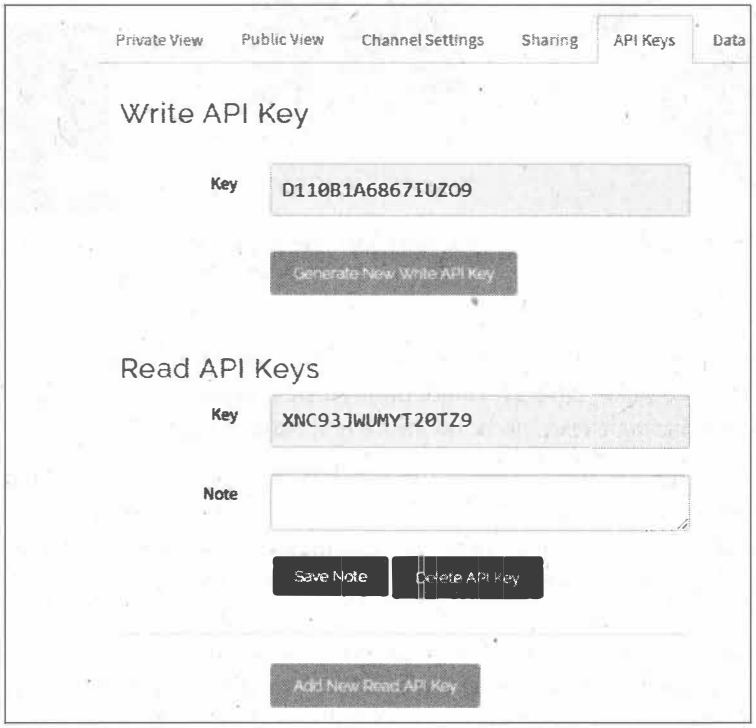


Рис. 5.49. Ключи API Keys для отправки и получения данных

5.3.1. Подготовка устройства IoT на плате Arduino Nano 33 IoT для подключения к сервису ThingSpeak

Создадим IoT-устройство на плате Arduino Nano 33 IoT, для чего подключим к плате датчик BME280 фирмы Bosch Sensortec (рис. 5.50), предназначенный для измерения атмосферного давления, температуры и влажности.

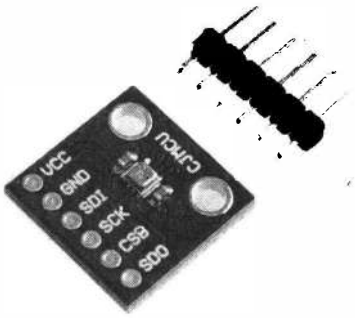


Рис. 5.50. Датчик BME280

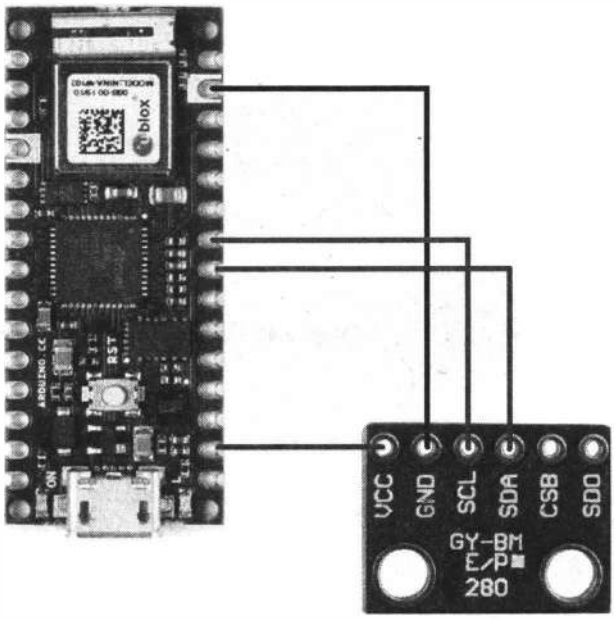


Рис. 5.51. Монтажная схема подключения датчика BME280 к плате Arduino Nano 33 IoT

Датчик поддерживает два интерфейса: I^2C и SPI, поэтому подключать его модуль можно двумя способами. Мы здесь воспользуемся вариантом подключения по протоколу I^2C. Монтажная схема подключения показана на рис. 5.51.

Для работы с датчиком необходимо установить две библиотеки: Adafruit BME280 Library и Adafruit Sensor (напомню, что мы устанавливали их для работы с сервисом «Народный мониторинг» в *разд. 5.2.3*, однако, если они по каким-либо причинам еще не установлены, скачайте и установите их). Чтобы работать с датчиком по протоколу I^2C, необходимо определить I^2C-адрес датчика, т. к. он может иметь адрес `0x76` или `0x77`. Определение адреса датчика осуществляется с помощью скетча I^2C-сканер, приведенного в листинге 5.4, — загрузите скетч в плату Arduino и запустите монитор последовательного порта, где будут отображаться адреса всех подключенных к шине I^2C-устройств (рис. 5.52).

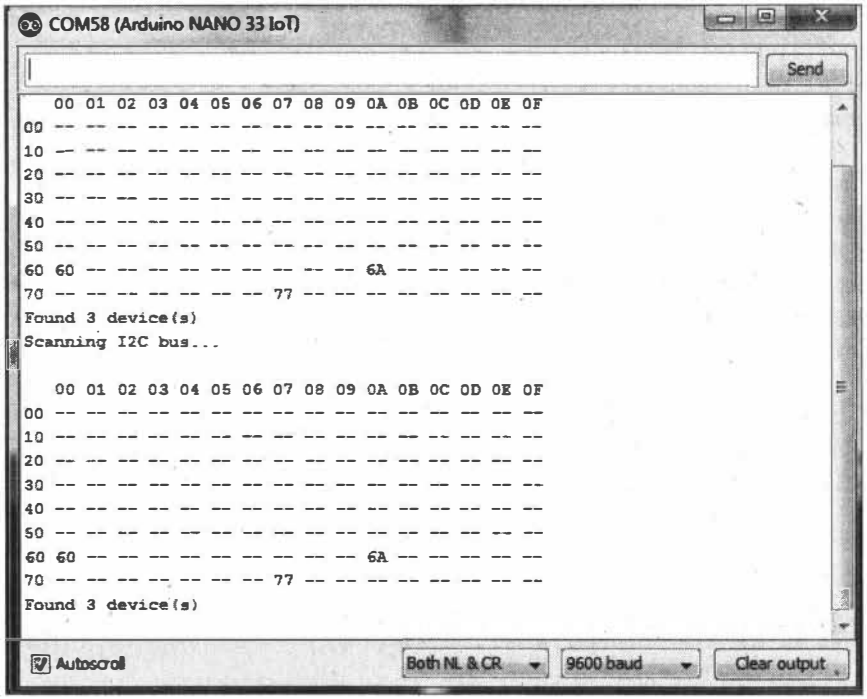


Рис. 5.52. Поиск I²C-устройств, подключенных к плате Arduino

Листинг 5.4

```
#include "Wire.h"
void setup()
  {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("\nI2C Scanner");
  Wire.begin();
  }

void loop()
  {
  int nDevices;
  byte error, address;
  Serial.println("Scanning I2C bus...\n");
  nDevices = 0;
  Serial.print("   00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F");

  for(address = 0; address < 128; address++ )
  {
    if((address % 0x10) == 0)
    {
      Serial.println();
```

```
    if(address < 16)
      Serial.print('0');
    Serial.print(address, 16);
    Serial.print(" ");
  }
  // The i2c_scanner uses the return value of
  // the Write.endTransmisstion to see if
  // a device did acknowledge to the address.
  Wire.beginTransmission(address);
  error = Wire.endTransmission();
  if (error == 0)
  {
    if (address<16)
       Serial.print("0");
    Serial.print(address, HEX);

    nDevices++;
  }
  else
  {
    Serial.print("--");
  }
  Serial.print(" ");
  delay(1);
}
Serial.println();
if (nDevices == 0)
   Serial.println("No I2C devices found\n");
 else
 {
   Serial.print("Found ");
   Serial.print(nDevices);
   Serial.println(" device(s) ");
 }
delay(5000);           // wait 5 seconds for next scan
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.4, можно найти в папке *examples\05\05_04* сопровождающего книгу электронного архива (см. *приложение*).

Если окажется, что I^2C-адрес датчика равен `0x76`, внесите соответствующее изменение в файл Adafruit BME280.h библиотеки Adafruit BME280 Library (рис. 5.53).

ЭЛЕКТРОННЫЙ АРХИВ

Библиотека Adafruit_BME280 размещена в каталоге *libraries* сопровождающего книгу электронного архива (см. *приложение*).

```
  Designed specifically to work with the Adafruit BME280 Breakout
  ----> http://www.adafruit.com/products/2650

  These sensors use I2C or SPI to communicate, 2 or 4 pins are required
  to interface.

  Adafruit invests time and resources providing this open source code,
  please support Adafruit andopen-source hardware by purchasing products
  from Adafruit!

  Written by Limor Fried & Kevin Townsend for Adafruit Industries.
  BSD license, all text above must be included in any redistribution
 ***************************************************************************/
#ifndef __BME280_H__
#define __BME280_H__

#if (ARDUINO >= 100)
 #include "Arduino.h"
#else
 #include "WProgram.h"
#endif

#include <Adafruit_Sensor.h>
#include <Wire.h>

/*=========================================================================
    I2C ADDRESS/BITS
    -----------------------------------------------------------------------*/
    #define BME280_ADDRESS                (0x77)
/*=========================================================================*/

/*=========================================================================
```

Исправьте этот адрес на адрес вашей платы

Рис. 5.53. Внесение изменения в файл Adafruit BME280.h

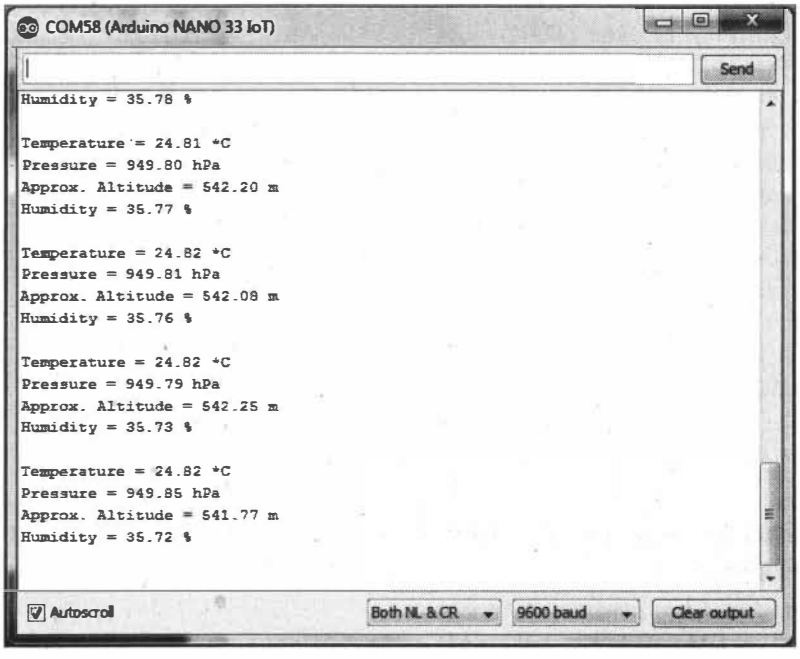


Рис. 5.54. Вывод данных с датчика BME280 в монитор последовательного порта

Загрузите в плату пример BME280test из библиотеки Adafruit BME280 Library, откройте монитор последовательного порта, и вы увидите вывод значений атмосферного давления, температуры и влажности (рис. 5.54).

5.3.2. Отправка данных в сервис ThingSpeak по протоколам HTTP GET и POST

Самый простой способ отправки данных с устройства в сервис ThingSpeak — HTTP GET:

```
https://api.thingspeak.com/update?api_key=<Your API Key>&field1=<data field1>
```

По протоколу HTTP POST отправка данных в канал ThingSpeak осуществляется следующим образом:

```
URL: http://api.thingspeak.com/update
Content Type: application/x-www-form-urlencoded
Content: key=<Your API Key>&field1=<data field1>
```

Напишем скетч для отправки данных датчика BME280 в канал ThingSpeak по протоколу HTTP POST. Данные отправляем с интервалом

```
unsigned long updateThingSpeakInterval = 16 * 1000;
```

Пытаемся подключиться к сервису ThingSpeak

```
char thingSpeakAddress[] = "api.thingspeak.com";
```

В случае десяти неудачных попыток необходимо заново подключиться к сети:

```
if (failedCounter > 10 )
  {startWiFi();}
void startWiFi() {
  status = WL_IDLE_STATUS;
  client.stop();
  // подключение к точке доступаа WiFi
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);
    // ждем 5 сек:
    delay(5000);
  }
  printWifiData();
}
```

Подключившись к серверу, получаем данные с датчика BME280 и отправляем их в канал ThingSpeak:

```
  // отправка данных в канал ThingSpeak
  if(!client.connected() && (millis() - lastConnectionTime >
                                                updateThingSpeakInterval))
```

```
{
// получение данных bme280
t=bme.readTemperature();
h=bme.readHumidity();
p=bme.readPressure()/133.322;
Serial.print("Temperature = ");
Serial.print(t);
Serial.println(" *C");
Serial.print("Pressure = ");
Serial.print(p);
Serial.println(" mmHg");
Serial.print("Humidity = ");
Serial.print(h);
Serial.println(" %");

String data = "field1="+ String(t)+"&field2="+ String(h)+"&field3="+
        String(p);
updateThingSpeak(data);
}
```

Содержание функции отправки данных `updateThingSpeak()` **показано в листинге** 5.5.

Листинг 5.5

```
void updateThingSpeak(String tsData)
  {
  if (client.connect(thingSpeakAddress, 80))
    {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: "+writeAPIKey+"\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(tsData.length());
    client.print("\n\n");

    client.print(tsData);
    lastConnectionTime = millis();
    if (client.connected())
      {
      Serial.println("Connecting to ThingSpeak...");
      Serial.println();

      failedCounter = 0;
      }
    else
      {
      failedCounter++;
```

```
      Serial.println("Connection to ThingSpeak failed ("+String(failedCounter,
                                                              DEC)+")");
      Serial.println();
      }
    }
  else
    {
    // увеличение счетчика неуспешных попыток отправки данных
    failedCounter++;
    Serial.println("Connection to ThingSpeak Failed
       ("+String(failedCounter, DEC)+")");
    Serial.println();
    lastConnectionTime = millis();
    }
}
```

Электронный архив

Полный скетч отправки данных в сервис ThingSpeak можно найти в папке *examples\05\05_05* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату Arduino, и через некоторое время вы увидите графики отправляемых в канал ThingSpeak значений (рис. 5.55).

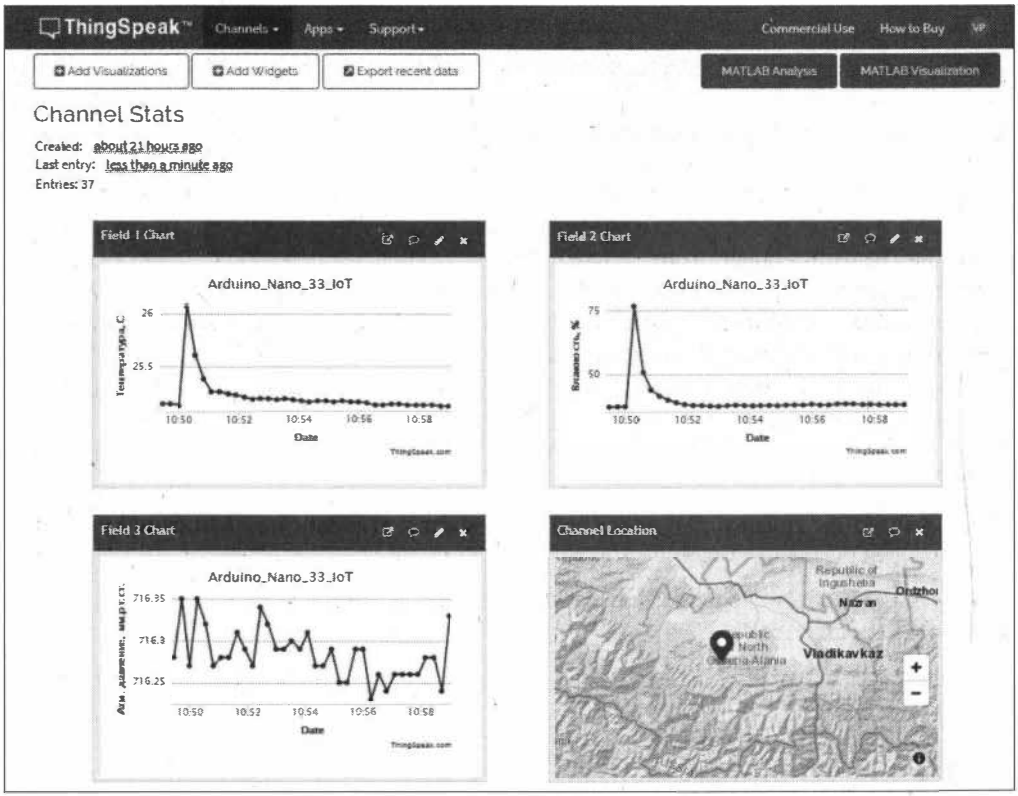


Рис. 5.55. Отображение в канале ThingSpeak графиков поступивших из платы Arduino данных

5.3.3. Отправка данных в сервис ThingSpeak по протоколу MQTT

Рассмотрим отправку данных в сервис ThingSpeak с использованием протокола MQTT. Распространенные MQTT-брокеры имеют один недостаток — у них нет встроенной базы данных, т. е. они только принимают и передают сообщения. Сервис ThingSpeak выгодно отличается в этом плане, поскольку поддерживает работу в качестве MQTT-брокера и позволяет сохранять данные в своей СУБД.

Напишем скетч для отправки данных датчика BME280 в канал ThingSpeak по протоколу MQTT.

Подключаем необходимые библиотеки:

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

Данные для подключения платы Arduino к точке доступа Wi-Fi:

```
char ssid[] = "Kassa1";
char pass[] = "12345678";
int status = WL_IDLE_STATUS;
```

Данные, необходимые для отправки данных в канал:

```
char mqttUserName[] = "TSArduinoMQTTDemo";
char mqttPass[] = "0AJ9UGHFF763MPCI";
char writeAPIKey[] = "D110B1A6867IUZO9";
long channelID = 1131248;
```

Значение параметра `mqttPass` получаем на странице профиля **My Profile** из поля **MQTT API Key** (рис. 5.56).

Создаем экземпляр `PubSubClient` и определяем брокера ThingSpeak MQTT:

```
PubSubClient mqttClient(client);
const char* server = "mqtt.thingspeak.com";
```

В разделе `setup()` после подключения к беспроводной сети установим данные брокера MQTT:

```
mqttClient.setServer(server, 1883);
```

В разделе `loop()` устанавливаем соединение MQTT и публикуем данные в канале через заданный интервал времени:

```
void loop()  {
  // Проверяем соединение MQTT
  if (!mqttClient.connected())
  {
    reconnect();
  }
```

```
  // Вызываем цикл непрерывно для установки соединения с сервером.
  mqttClient.loop();
  // публикация данных в ThingSpeak.
  if (millis() - lastConnectionTime > updateThingSpeakInterval)
  {
    mqttPublishFeed();
  }
}
```

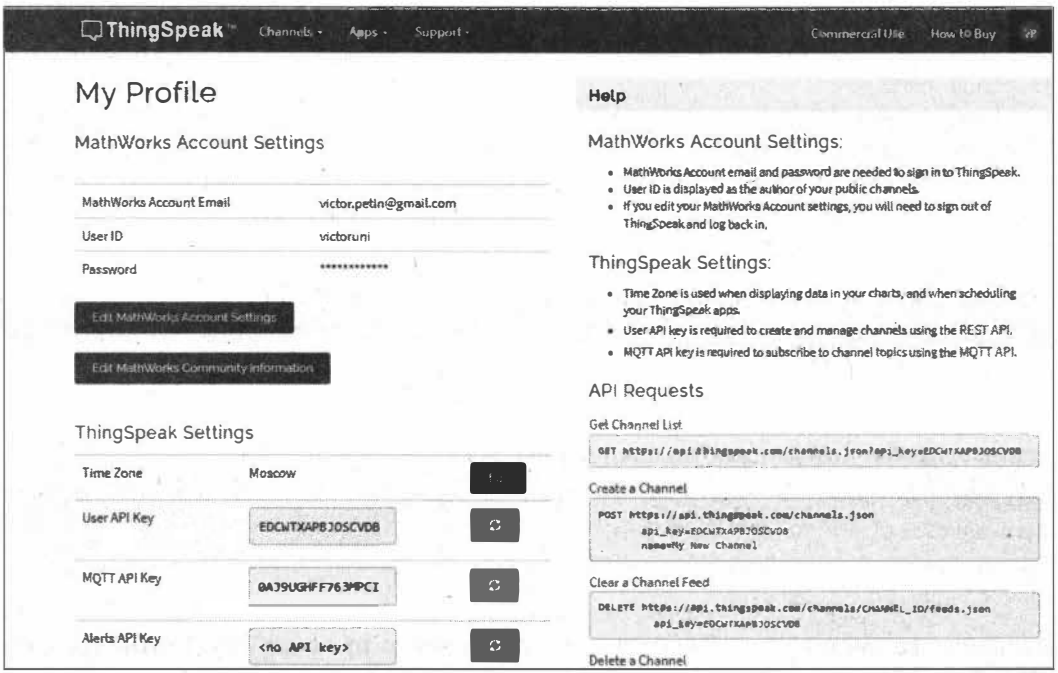


Рис. 5.56. Значение параметра `mqttPass` в поле MQTT API Key

Функция `reconnect()` для подключения клиента Arduino к брокеру MQTT:

```
void reconnect()
{
  char clientID[9];

  // цикл подключения
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Генерация ClientID
    for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
    }
    clientID[8]='\0';
```

```
    // Подключение к брокеру
    if (mqttClient.connect(clientID,mqttUserName,mqttPass))
    {
      Serial.println("connected");
    } else
    {
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Функция `mqttPublishFeed()` **предназначена для публикации данных датчика в канале ThingSpeak. С помощью этой функции вы можете публиковать сразу в несколько полей. В следующем примере выполняется публикация в поля** `field1`, `field2` **и** `field3` **канала:**

```
// отправка данных в канал ThingSpeak
void mqttPublishFeed() {
    t=bme.readTemperature();
    h=bme.readHumidity();
    p=bme.readPressure()/133.322;
    Serial.print("Temperature = ");
    Serial.print(t);
    Serial.println(" *C");
    Serial.print("Pressure = ");
    Serial.print(p);
    Serial.println(" mmHg");
    Serial.print("Humidity = ");
    Serial.print(h);
    Serial.println(" %");

  // Формирование строки для отправки в ThingSpeak.
  String data = String("field1=") + String(t, DEC) + "&field2=" +
              String(h, DEC) + "&field3=" + String(p, DEC);
  int length = data.length();
  const char *msgBuffer;
  msgBuffer=data.c_str();
  Serial.println(msgBuffer);

  // Создание топика и публикация в канал.
  String topicString = "channels/" + String( channelID ) +
          "/publish/"+String(writeAPIKey);
  length = topicString.length();
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
```

```
  mqttClient.publish( topicBuffer, msgBuffer );
  lastConnectionTime = millis();
}
```

Электронный архив

Полный скетч отправки данных в сервис ThingSpeak по протоколу MQTT можно найти в папке *examples\05\05_06* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату Arduino и откройте монитор последовательного порта (рис. 5.57). Через некоторое время вы также увидите графики отправляемых значений в канале ThingSpeak.

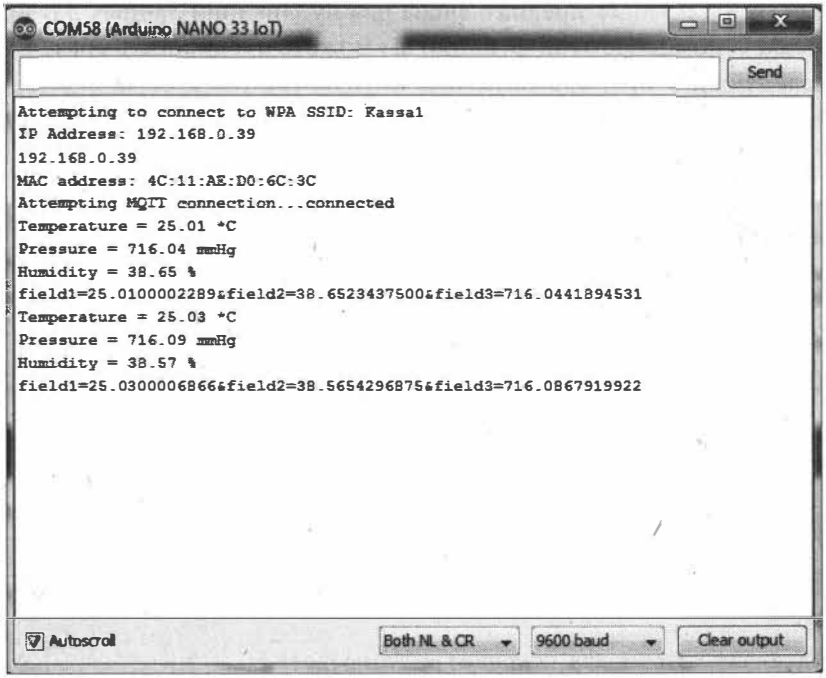


Рис. 5.57. Просмотр в последовательном порту данных, отправляемых в канал ThingSpeak

5.3.4. Добавление виджетов в канал ThingSpeak

Для более удобной визуализации данных мы можем добавлять в канал ThingSpeak виджеты. Добавим для примера виджет шкалы отображения текущей температуры: перейдите в канал, на вкладке **Public View** (или **Private View**) нажмите на кнопку **Add Widgets,** в открывшемся окне выберите необходимый виджет (рис. 5.58) и нажмите на кнопку **Next**.

В следующем открывшемся окне (рис. 5.59) настройте параметры виджета: выберите поле канала для отображения, интервалы значений, интервал обновления и

сохраните введенные значения, нажав на кнопку **Create**. Виджет появится в канале (рис. 5.60).

Если виджет создан в **Public View**, можно предоставить доступ для его просмотра другим пользователям. По ссылке вида **https://www.thingspeak.com/channels/<channelID>/widgets/<widgetID>** все желающие увидят страницу с виджетом, показанную на рис. 5.61.

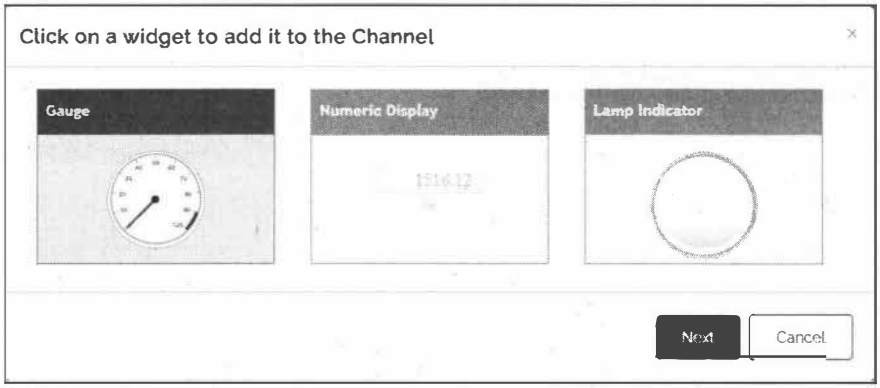


Рис. 5.58. Добавление виджета в канал ThingSpeak

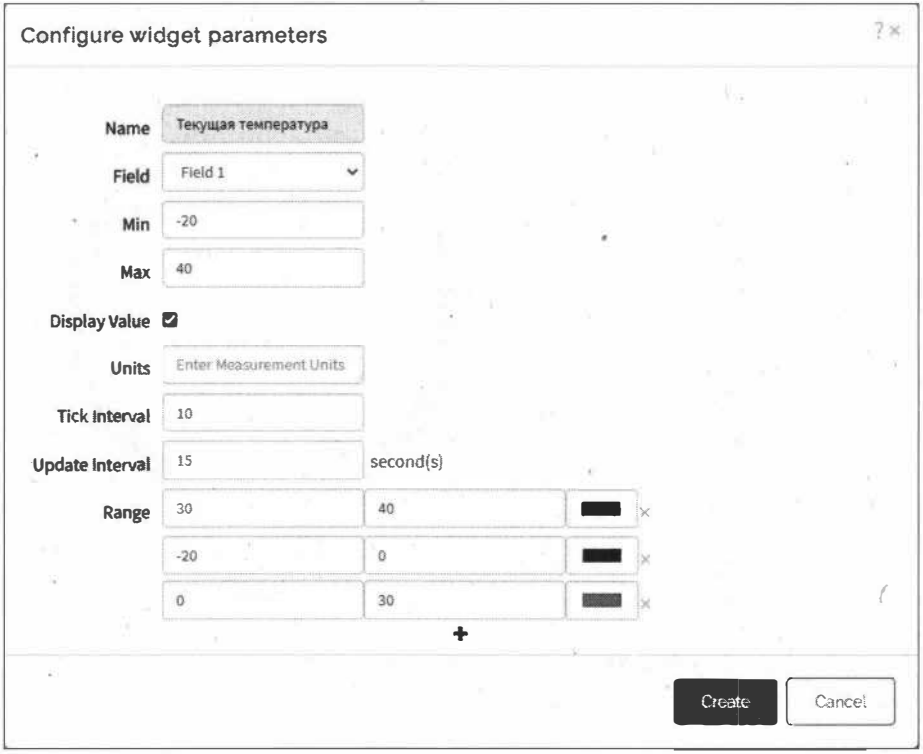


Рис. 5.59. Настройка параметров виджета

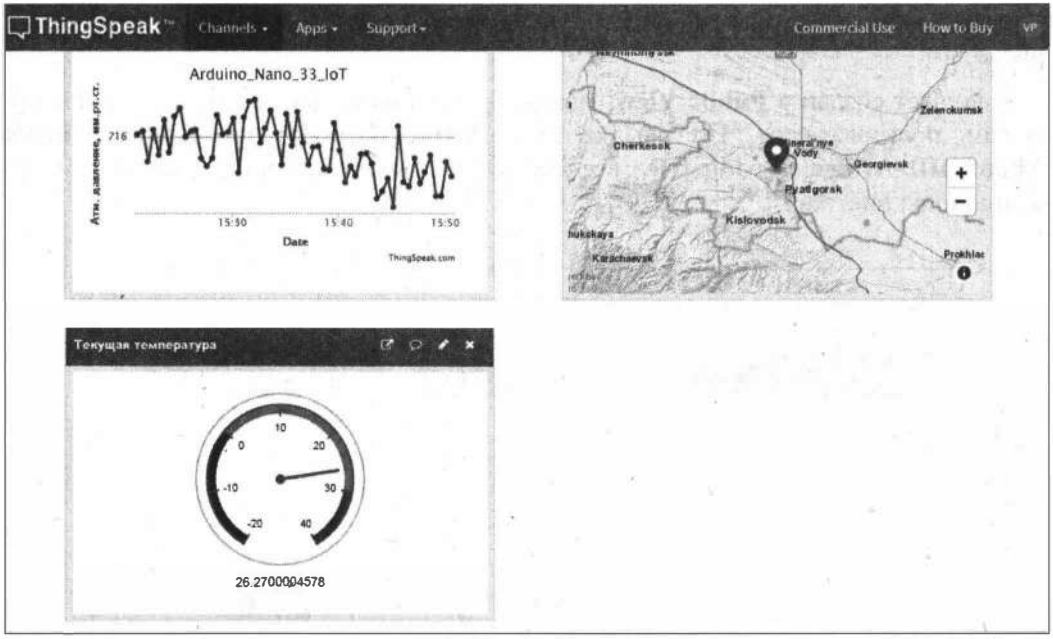


Рис. 5.60. Виджет шкалы отображения текущей температуры добавлен в канал ThingSpeak

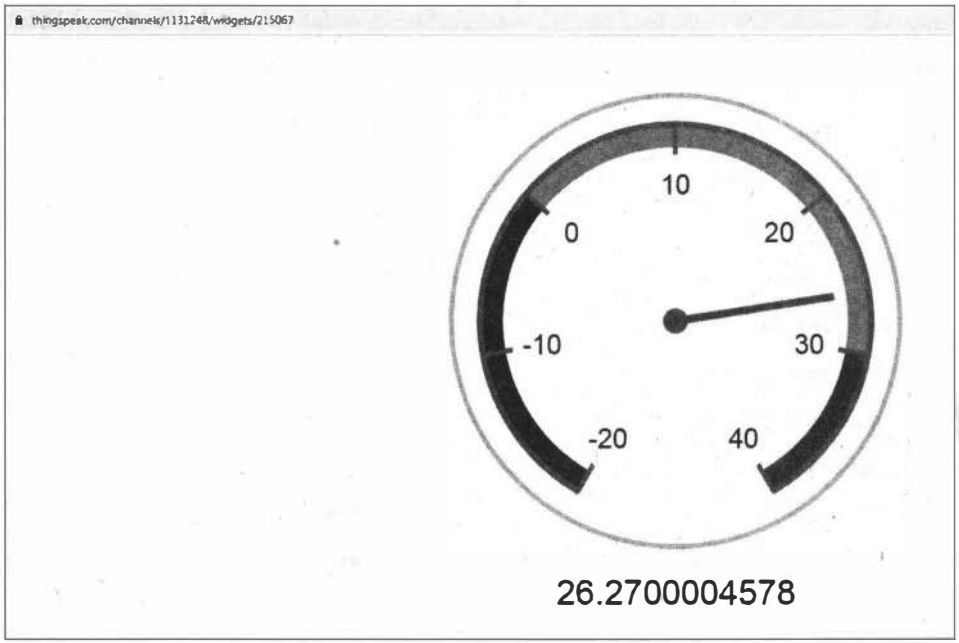


Рис. 5.61. Страница для просмотра виджета

5.3.5. Преобразование и визуализация данных, поступивших в ThingSpeak

Рассмотрим преобразование данных, поступивших в ThingSpeak. Для этого используются инструменты MATLAB. Создадим для примера канал, который будет выводить данные температуры в градусах шкалы Фаренгейта.

Сценарий анализа MATLAB создается из кода шаблона: на странице канала ThingSpeak нажмите на кнопку **Matlab Analysis,** выберите шаблон **Custom (no starter code)** и нажмите на кнопку **Create** (рис. 5.62).

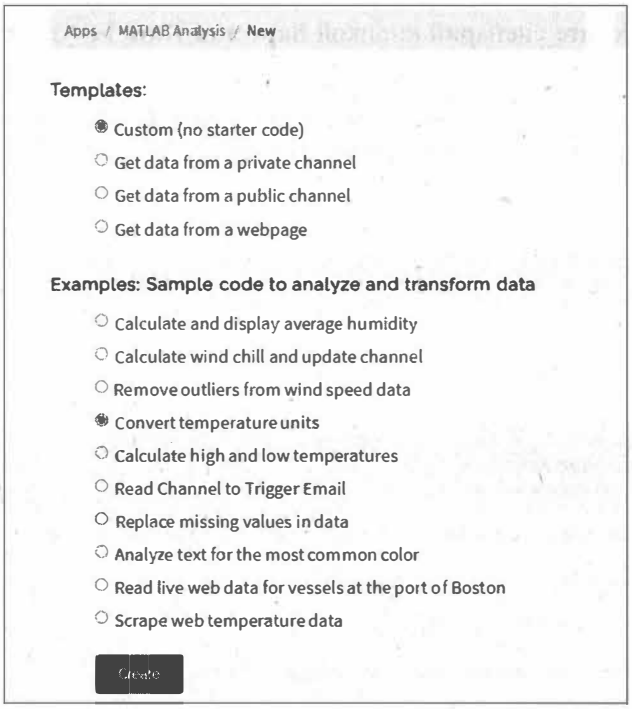


Рис. 5.62. Создание сценария MATLAB на основе шаблона Convert temperature units

В открывшееся окно **MATLAB Code** введите следующий код.

Объявляем переменные:

```
readChannelID = 1131248;
temperatureFieldID = 1;
readAPIKey = '';
```

где:

- `readChannelID` — ID канала для чтения данных;
- `temperatureFieldID` — номер поля данных температуры (`Field1`);
- `readAPIKey` — для публичного канала оставляем пустое значение. Если чтение данных будет производиться из приватного канала, необходимо внести значение поля **Read API Keys** с вкладки **API keys**.

Получаем последние данные поля `temperatureFieldID` (`Field1`) из канала `readChannelID`:

```
tempF =  thingSpeakRead(readChannelID,'Fields', temperatureFieldID,
        'ReadKey', readAPIKey);
```

Преобразовываем полученные данные:

```
tempF = tempC*9/5+32;
```

и выводим результат в окно:

```
display(tempF,'Temperature in F');
```

Сохраните и запустите сценарий кнопкой **Save and Run**. Результат преобразования показан на рис. 5.63.

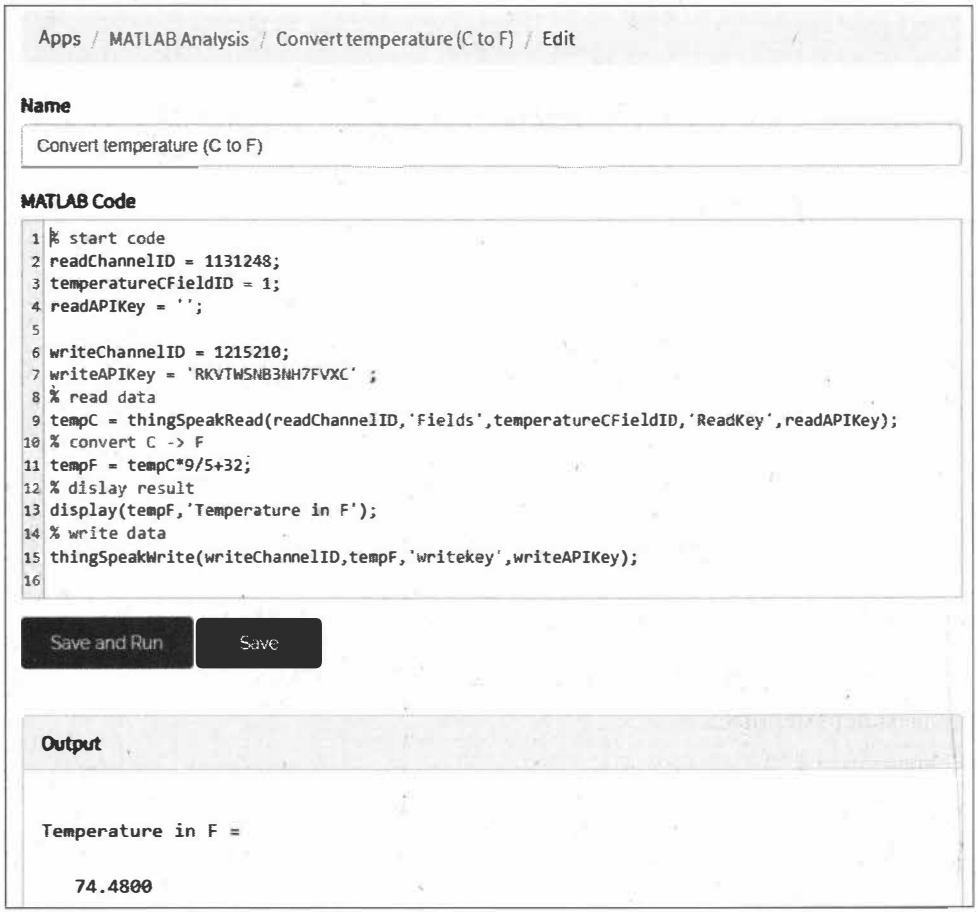


Рис. 5.63. Результат выполнения сценария MATLAB

Теперь добавим отправку конвертированных данных в другой канал. Создадим для этого еще один канал: **Arduino_Nano_33_IoT_1** и активируем одно поле: **Field1** — температура в F (рис. 5.64).

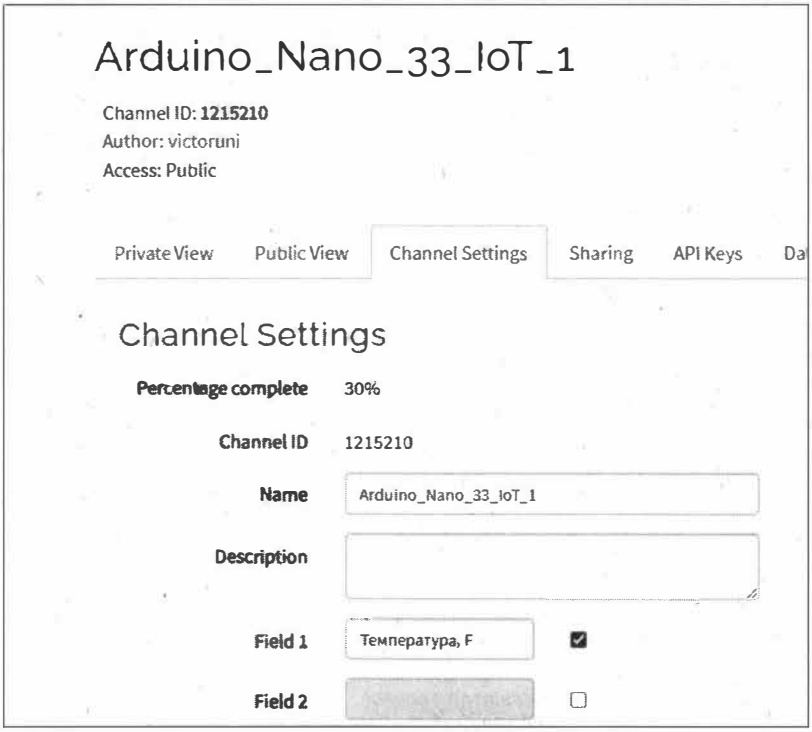


Рис. 5.64. Новый канал ThingSpeak

Объявляем переменные для канала Arduino_Nano_33_IoT_1:

```
writeChannelID = 1215210;
writeAPIKey = 'RKVTWSNB3NH7FVXC' ;
```

и отправляем полученное значение `tempF` в канал Arduino_Nano_33_IoT_1:

```
thingSpeakWrite(writeChannelID,tempF,'writekey',writeAPIKey);
```

Сохраняем и запускаем сценарий — произойдет однократная отправка данных в канал Arduino_Nano_33_IoT_1.

Теперь необходимо настроить периодичность отправки данных (периодичность запуска сценария). Командой **Apps | TimeControl | New TimeControl** создайте новое значение **NewTimeControl** и задайте для него периодичность запуска сценария. Затем в поле **Code to execute** выберите наш сценарий **Convert temperature (C to F)** и сохраните настройки, нажав на кнопку **Save TimeControl** (рис. 5.65).

Теперь данные в канале Arduino_Nano_33_IoT_1 будут появляться каждые 5 минут (рис. 5.66).

Это очень простой пример создания сценария аналитики данных, поступающих в канал ThingSpeak. В *разд 6.9* мы рассмотрим более сложный пример — предсказание погоды по данным, поступающим в канал ThingSpeak из метеостанции.

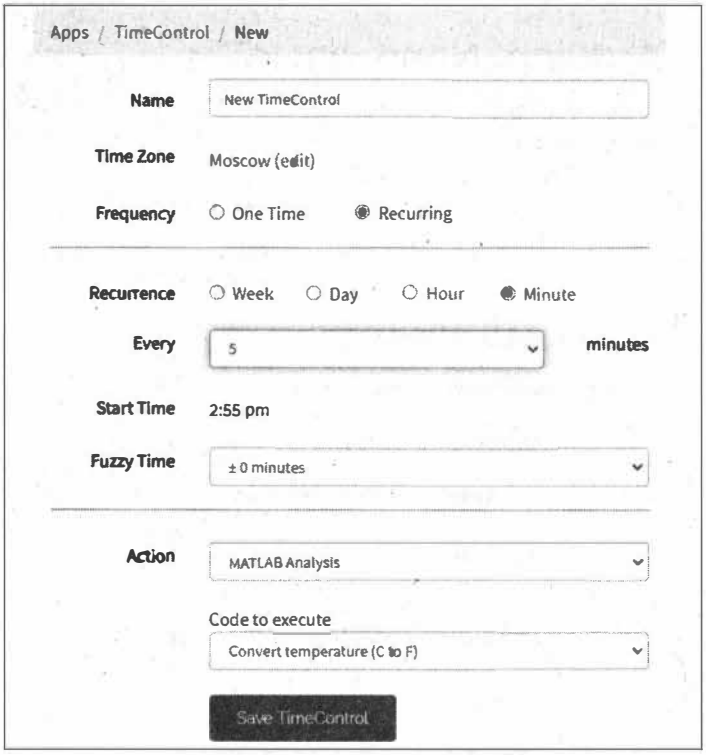


Рис. 5.65. Создание периодичности запуска сценария

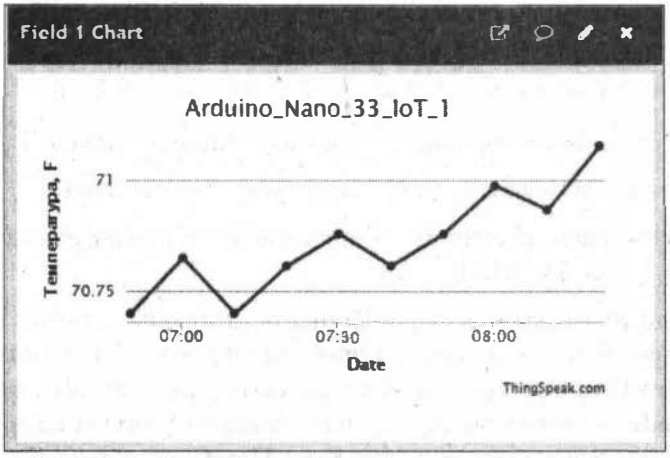


Рис. 5.66. Данные в канале Arduino_Nano_33_IoT_1

5.3.6. Визуализация данных: MATLAB Visualization

Приложение MATLAB Visualization позволяет использовать интерактивные графики ThingSpeak MATLAB для визуализации и изучения данных, собранных в канале. Рассмотрим простой пример визуализации — отображение данных температуры и

относительной влажности воздуха, приходящих в канал ThingSpeak на одном графике. Добавим для этого приложение MATLAB Visualization (**Apps | MATLAB Visualizations | New**), выберите шаблон **Custom** и нажмите на кнопку **Create**.

В открывшееся окно **MATLAB Code** введите следующий код.

Объявляем переменные:

```
readChannelID = 1131248;
readAPIKey = '';
```

где:

- `readChannelID` — ID канала;
- `readAPIKey` — для публичного канала оставляем пустое значение. Если чтение данных будет производиться из приватного канала, необходимо внести значение поля **Read API Keys** с вкладки **API keys**.

Используем функцию `thingSpeakRead()` для получения 10 точек данных о температуре и влажности:

```
[data,timeStamps] = thingSpeakRead(readChannelID, 'NumPoints', 10,
                'ReadKey', readAPIKey);
```

Извлекаем данные о температуре и влажности:

```
temperatureData = data(:, 1);
humidityData = data(:, 2);
```

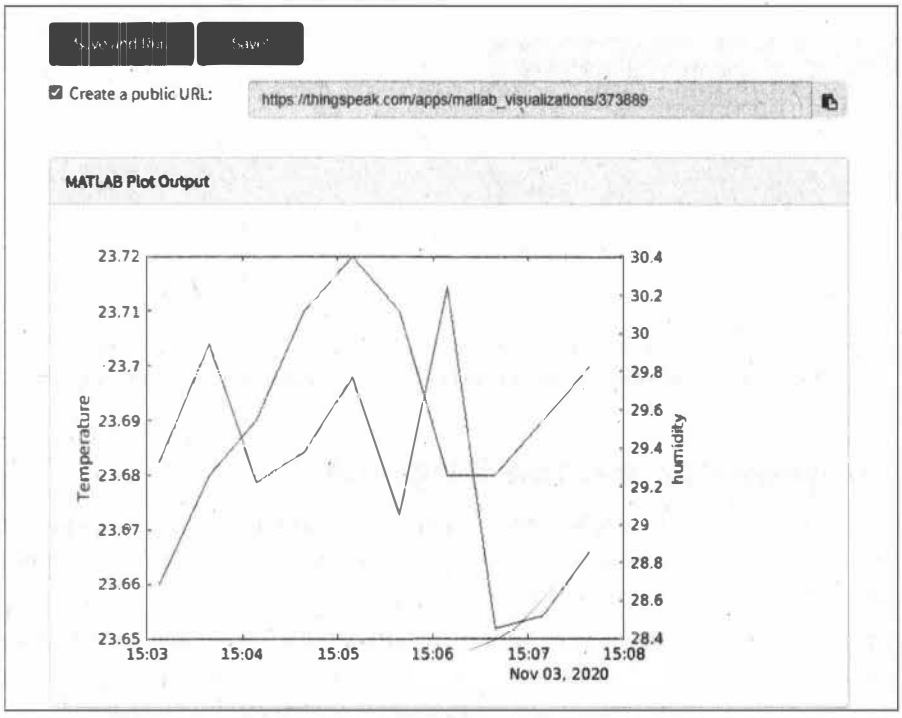


Рис. 5.67. График MATLAB Visualization

Задействуем функции `plot()` и `yyaxis()` для создания двухосного графика. Для установки меток оси Y с каждой стороны применим функцию `ylabel()`:

```
yyaxis left
plot(timeStamps, temperatureData);
ylabel('Temperature');

yyaxis right
plot(timeStamps, humidityData);
ylabel('humidity');
```

Кнопкой **Save and Run** сохраняем и запускаем сценарий. Результат его выполнения показан на рис. 5.67. Данные также будут отображаться и в канале (рис. 5.68).

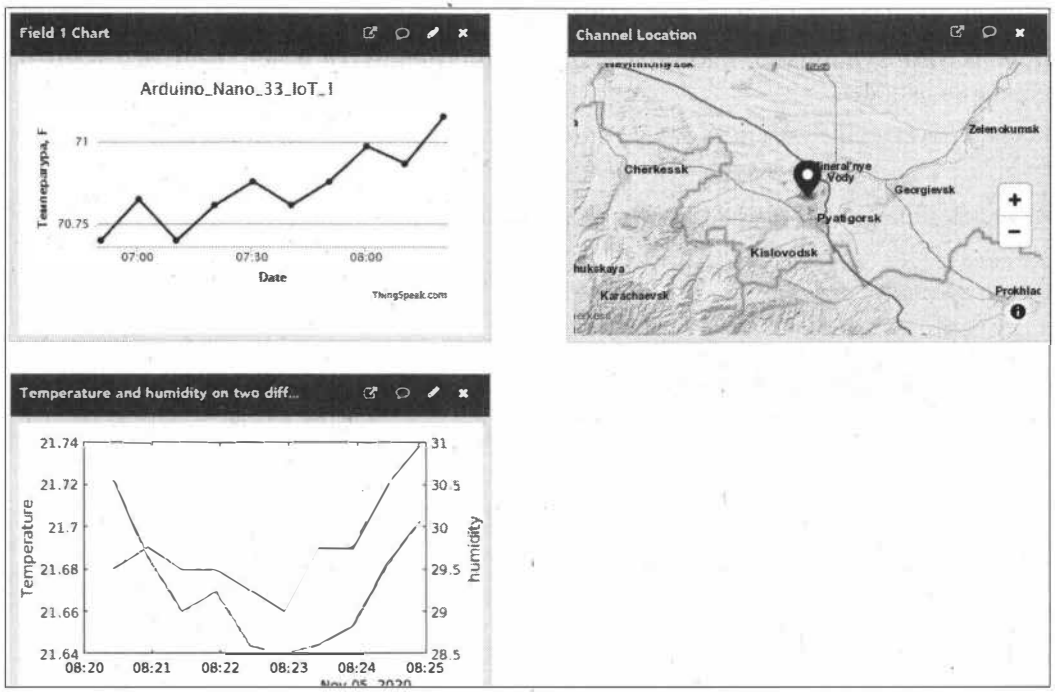


Рис. 5.68. Отображение графика MATLAB Visualization в канале ThingSpeak

5.3.7. Чтение данных из канала ThingSpeak

Используя HTTP-вызовы и REST API, вы можете читать данные из своих каналов ThingSpeak. Вы также можете использовать метод подписки MQTT для получения сообщений при каждом обновлении канала.

Рассмотрим чтение данных из всех или отдельных полей в канале с помощью HTTP GET.

Чтение данных *из всех полей* в канале с помощью HTTP GET обеспечивает функция `Read Data`: **https://api.thingspeak.com/channels/<channel_id>/feeds.<format>**.

Функция имеет два обязательных параметра:

- **<channel_id>** — ID канала;
- **<format>** — формат ответа HTTP (json, xml или csv).

Чтение данных *из одного поля* в канале с помощью HTTP GET обеспечивает функция `Read Field`: **https://api.thingspeak.com/channels/<channel_id>/fields/<field_id>.<format>**.

- Функция имеет три обязательных параметра:
- **<channel_id>** — ID канала;
- **<field_id>** — ID поля для выбранног канала;
- **<format>** — формат ответа HTTP (JSON, XML или CSV).

Кроме обязательных параметров, функции `Read Data` и `Read Field` имеют множество необязательных параметров, позволяющих гибко настраивать получаемые данные. Список дополнительных параметров функции `ReadData` представлен в табл. 5.2

***Таблица 5.2.** Список параметров запроса функции ReadData*

Название	Описание
api_key	Ключ API чтения для приватных каналов (Read API Key)
results	Количество извлекаемых записей. Максимальное количество: 8000
days	Количество 24-часовых периодов до настоящего момента для включения в ответ. По умолчанию 1
minutes	Количество 60-секундных периодов до настоящего момента для включения в ответ. По умолчанию 1440
start	Дата начала в формате ГГГГ-ММ-ДД -ЧЧ: НН: СС
end	Дата окончания в формате ГГГГ-ММ-ДД% 20ЧЧ: НН: СС
timezone	Идентификатор из справочника часовых поясов для этого запроса
offset	Смещение часового пояса, в котором отображаются результаты
status	Включить обновления статуса в ленту, установив status = true
metadata	Включить метаданные для канала, установив metadata = true
location	Включить в ответ широту, долготу и высоту, установив location = true
min	Минимальное значение для включения в ответ
max	Максимальное значение, которое нужно включить в ответ
round	Округлить до указанного количества десятичных знаков
timescale	Получить первое значение за выбранное количество минут. Допустимые значения: 10, 15, 20, 30, 60, 240, 720, 1440, daily.
sum	Получить сумму выбранного количества минут. Допустимые значения: 10, 15, 20, 30, 60, 240, 720, 1440, daily
average	Получить среднее значение количества минут. Допустимые значения: 10, 15, 20, 30, 60, 240, 720, 1440, daily

Таблица 5.2 (окончание)

Название	Описание
median	Получить медианное значение количества минут. Допустимые значения: 10, 15, 20, 30, 60, 240, 720, 1440, daily

Рассмотрим пример получения данных из канала ThingSpeak. В качестве IoT-устройства мы воспользуемся платой ESP32 или ESP8266 — например, FireBeetle ESP32 (с ее подключением мы познакомились в *разд. 3.5*), и загрузим в нее скетч из листинга 5.7 для получения пяти последних данных из канала `channellId` поля `fielld1` в формате XML. Загрузив скетч в плату, откройте монитор последовательного порта (рис. 5.69).

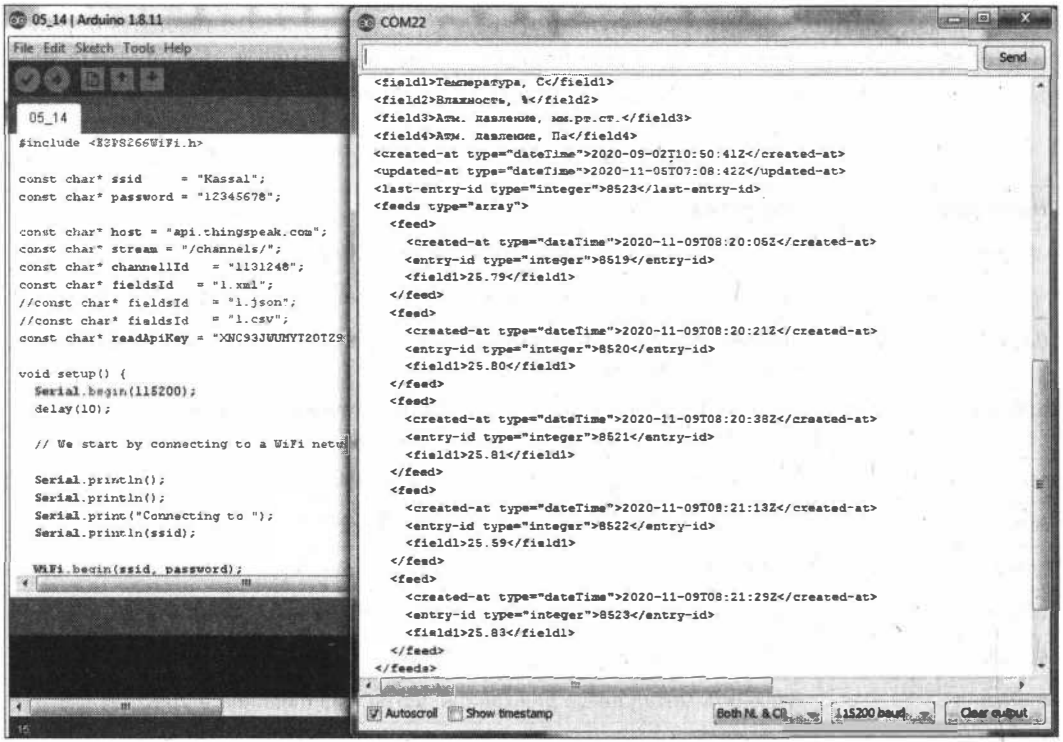


Рис. 5.69. Просмотр в последовательном порту данных, полученных из канала ThingSpeak в формате XML

Листинг 5.7

```
#include <ESP8266WiFi.h>
// ssid и пароль точки доступа Wi-Fi
const char* ssid     = "Kassa1";
const char* password = "12345678";
// данные канала ThingSpeak
const char* host = "api.thingspeak.com";
```

```
const char* stream = "/channels/";
const char* channelId   = "1131248";
const char* fieldsId    = "1.xml";
//const char* fieldsId   = "1.json";
//const char* fieldsId   = "1.csv";
const char* readApiKey = "XNC93JWUMYT20TZ9";

void setup() {
  Serial.begin(115200);
  delay(10);
  // подключение к сети Wi-Fi
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  // вывод IP-адреса при подключении
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

int value = 0;

void loop() {
  delay(20000);
  ++value;

  Serial.print("connecting to ");
  Serial.println(host);

  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }

  // формирование URI запроса
  String url = "";
  url += stream;
  url += channelId;
```

```
  url += "/fields/";
  url += fieldsId;
  url += "?results=5&round=2&api_key=";
  url += readApiKey;

  Serial.print("Requesting URL: ");
  Serial.println(url);

  // отправка запроса на сервер ThingSpeak
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }

  // получение ответа от сервера ThingSpeak
  while(client.available()){
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }
  Serial.println();
  Serial.println("closing connection");
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.7, можно найти в папке *examples\05\05_07* сопровождающего книгу электронного архива (см. *приложение*).

Для получения данных в формате JSON или CSV надо в скетче раскомментировать нужное значение для переменной `fieldsId` (и закомментировать ненужное):

```
const char* fieldsId   = "1.xml";
//const char* fieldsId   = "1.json";
//const char* fieldsId   = "1.csv";
```

5.3.8. Отображение данных из канала ThingSpeak на дисплее светодиодной матрицы

Создадим пример получения данных из канала ThingSpeak с выводом их на дисплей. Для плат FireBeetle ESP32 и ESP8266 в качестве такого дисплея можно использовать светодиодную матрицу FireBeetle Covers-24x8 LED Matrix (рис. 5.70).

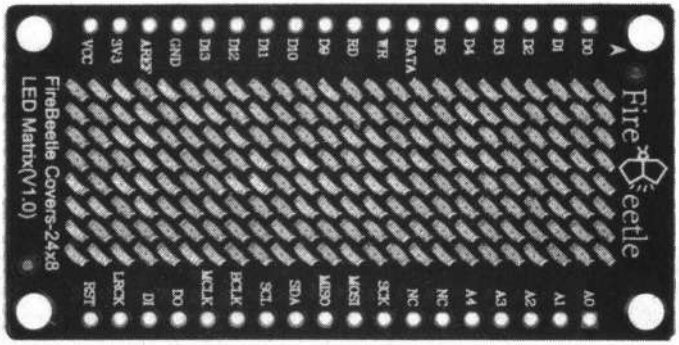


Рис. 5.70. Светодиодная матрица FireBeetle Covers-24x8 LED Matrix

Светодиодная матрица FireBeetle Covers-24x8 LED Matrix имеет четыре контакта CS, и выбрать нужный контакт CS можно с помощью переключателя набора номера (рис. 5.71). Таким образом, одна плата FireBeetle ESP32 может управлять четырьмя светодиодными матрицами одновременно.

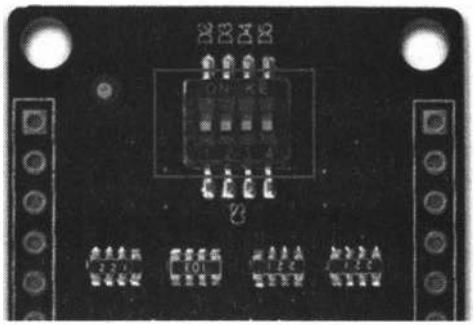


Рис. 5.71. Выбор контакта CS на плате FireBeetle Covers-24x8 LED Matrix

Назначение выводов платы FireBeetle ESP32, используемых для управления матрицей, показано в табл. 5.3.

***Таблица 5.3.** Назначение выводов платы FireBeetle ESP32, используемых для управления матрицей*

Номер	Метка	Описание
1	DATA	DATA <——> D6
2	WR	WR <——> D7 (Generally Not Used)
3	CS	CS <——> D2, D3, D4, D5 selectable
4	RD	RD <——> D8
5	VCC	5V <——> USB; 3.7V <——> LiPo Battery

Подключите матрицу к плате FireBeetle ESP32 или ESP8266 и загрузите в плату скетч, представленный в листинге 5.8. В результате его выполнения мы получаем

последние данные из канала channelId в формате CSV, округленные до одного десятичного знака. Из ответа сервера выделяем значения температуры и относительной влажности и попеременно выводим их на экран (рис. 5.72). Запрос к серверу повторяется каждые 20 секунд. В этом скетче задействуется библиотека DFRobot_HT1632C.

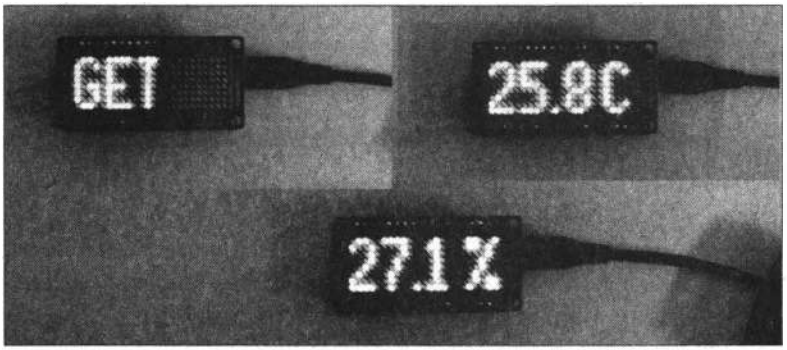


Рис. 5.72. Вывод на светодиодный дисплей данных, получаемых из сервиса ThingSpeak

Листинг 5.8

```
// подключение библиотек
#include <ESP8266WiFi.h>
#include "DFRobot_HT1632C.h"
// ssid и пароль точки доступа Wi-Fi
const char* ssid     = "Kassa1";
const char* password = "12345678";
// данные канала ThingSpeak
const char* host = "api.thingspeak.com";
const char* stream = "/channels/";
const char* channelId   = "1131248";
const char* readApiKey = "XNC93JWUMYT20TZ9";
// матрица
#define DATA D6
#define CS D2
#define WR D7
DFRobot_HT1632C ht1632c = DFRobot_HT1632C(DATA, WR,CS);
// служебные переменные
String temp="";
String humidity="";

void setup() {
  Serial.begin(115200);
  delay(10);

  // матрица
  ht1632c.begin();
```

```
  ht1632c.isLedOn(true);
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  ht1632c.print("WiFi");
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  // подключение к Wi-Fi
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  ht1632c.print("get");

  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
  // формирование URI для запроса
  String url = "";
  url += stream;
  url += channellId;
  url += "/feeds.csv";
  url += "?results=1&round=1&api_key=";
  url += readApiKey;
  Serial.print("Requesting URL: ");
  Serial.println(url);
  // отправка запроса на сервер
  client.print(String("GET ") + url + " HTTP/1.1\r\n" +
               "Host: " + host + "\r\n" +
               "Connection: close\r\n\r\n");
  unsigned long timeout = millis();
```

```
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Client Timeout !");
      client.stop();
      return;
    }
  }
  // ответ с сервера
  char json[600];
  while(client.available()){
    String line = client.readStringUntil('\r');
    if (line.indexOf("UTC") != -1) {
        //Serial.println(line);
        String str1=line.substring(line.indexOf("UTC")+5,
                    line.length()-1);
        String str2=str1.substring(str1.indexOf(",")+1,str1.length()-1);
        temp=str2.substring(0,str2.indexOf(","));
        Serial.print("t=");Serial.println(temp);
        String str3=str2.substring(str2.indexOf(",")+1,str2.length()-1);
        humidity=str3.substring(0,str3.indexOf(","));
        Serial.print("h=");Serial.println(humidity);
    }
  }
  Serial.println();
  Serial.println("closing connection");
  // вывод температуры
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  char c[6];
  temp.toCharArray(c, sizeof(c));
  ht1632c.print(c);
  ht1632c.setCursor(18,0);
  ht1632c.print("C");
  delay(10000);
  // вывод влажности
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  humidity.toCharArray(c, sizeof(c));
  ht1632c.print(c);
  ht1632c.setCursor(18,0);
  ht1632c.print("%");
  delay(10000);
}
```

Электронный архив

Скетч, соответствующий листингу 5.8, можно найти в папке *examples\05\05_08* сопровождающего книгу электронного архива (см. *приложение*).

ЭЛЕКТРОННЫЙ АРХИВ

Библиотека DFRobot_HT1632C размещена в каталоге *libraries* сопровождающего книгу электронного архива (см. *приложение*).

5.4. Проект Blynk: управление с планшета

Проект Blynk (**http://www.blynk.io**) в начале 2015 года успешно профинансировался на Kickstarter почти на 500%. Проект позволяет установить в приложении на смартфоне (планшете) различные виджеты: кнопки, слайдеры, дисплеи, графики и пр. и с их помощью управлять IoT-устройствами или получать данные с IoT-устройств. Проект уже поддерживает более 400 плат, включая Arduino, Particle, ARM Mbed, TI Energia, MicroPython, Node.js, OpenWRT и многие одноплатные компьютеры. Blynk работает через Интернет и может взаимодействовать с любым оборудованием через Ethernet, Wi-Fi или GSM, 2G, 3G, LTE и т. д. К нему также можно легко добавить свои собственные типы подключения. Проект предлагает обширный API аппаратного облачного приложения — вы можете выбрать C++, JS, Python или HTTP. Blynk Server развертывается за считаные минуты. Он работает в режиме реального времени и готов управлять миллиардами запросов от ваших устройств.

Приложения Blynk iOS и Android подключаются к облаку Blynk Cloud по умолчанию — доступ свободный для каждого пользователя Blynk. Сервер, впрочем, можно установить и локально.

5.4.1. Начало работы

Прежде всего необходимо выполнить следующие шаги:

- скачать и установить для Android или iOS бесплатное приложение из Play Маркет или App Store;
- запустить скачанное приложение Blynk и зарегистрироваться в сервисе Blynk;
- создать приложение для своего устройства, добавив на экран необходимые виджеты;
- скачать и установить библиотеку для своего устройства (Arduino, ESP, Raspberry Pi), загрузить скетч примера для него.

После этого запустить приложение на смартфоне, управлять выводами Arduino и отображать полученные данные.

Для скачивания приложения для Android заходим в Play Маркет и находим по поиску приложение Blynk (рис. 5.73). Скачиваем его и устанавливаем.

Далее необходимо создать аккаунт, указав адрес своей электронной почты и пароль (рис. 5.74). Войдя в свой аккаунт, создайте проект, нажав на кнопку +. Введите имя проекта и выберите тип устройства (рис. 5.75). Как уже отмечалось ранее, сервис поддерживает множество устройств, включая несколько плат Arduino, Raspberry Pi, ESP8266 и другие. Список устройств создатели сервиса обещают пополнять. Нажмите на кнопку **Create**, и проект будет создан.

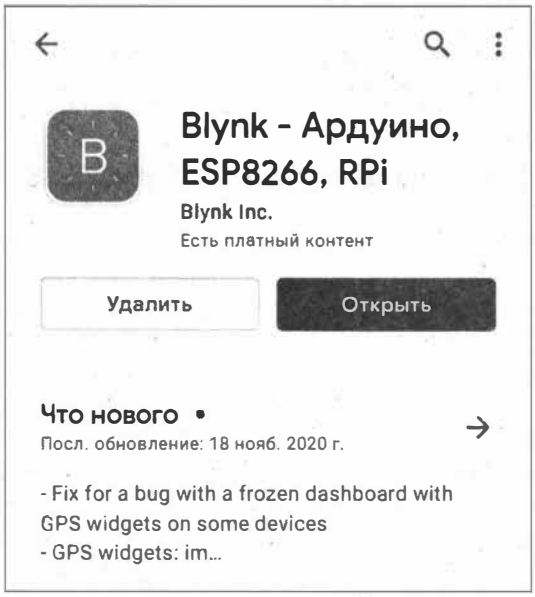


Рис. 5.73. Andriod-приложение Blynk в Play Маркет

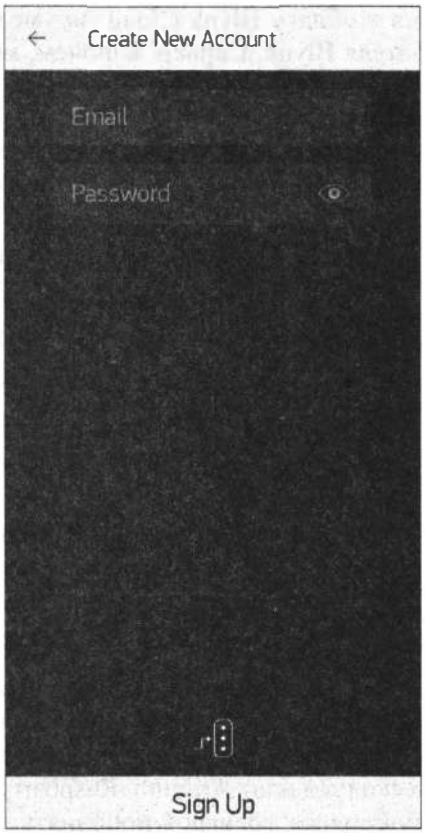


Рис. 5.74. Создание аккаунта в Blynk

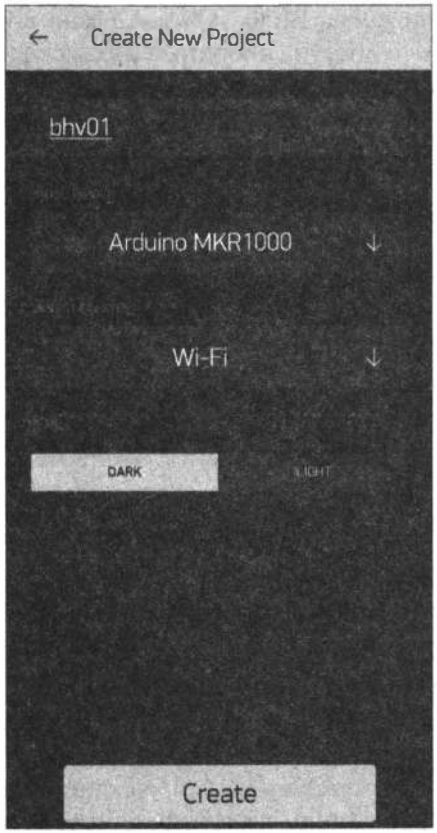


Рис. 5.75. Создание нового проекта в Blynk

При создании проекта автоматически генерируется уникальный *токен авторизации* (Auth Token), который понадобится при написании скетча. Он будет отправлен на указанную вами почту (рис. 5.76).

Рис. 5.76. Отправка токена авторизации для проекта в Blynk в почту пользователя

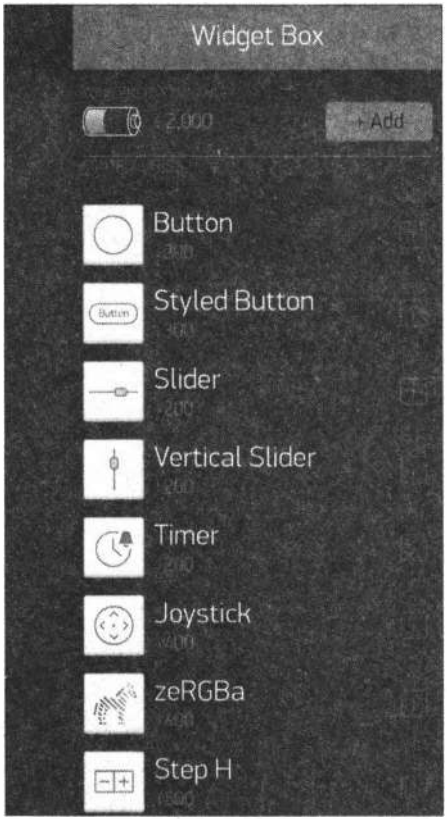


Рис. 5.77. Энергия — валюта для покупки виджетов в Blynk

Теперь в проект можно добавлять виджеты для управления и получения данных. Все элементы управления покупаются за «энергию». Каждый новый пользователь имеет 2000 единиц «энергии» на старте. Для получения большего количества «энергии» придется заплатить (рис. 5.77).

Для добавления виджетов нажимаем кнопку **+Add**. Выбор виджетов просто огромен.

5.4.2. Создание интерфейса на планшете. Добавление виджетов

Для создания интерфейса на экране планшета просто добавляем виджеты в созданный проект.

Виджет *Button*

Виджет **Button** (Кнопка (рис. 5.78) может работать в двух режимах: переключателя (нажатие и отжатие посылает одно сообщение) и в пуш-режиме (нажатие посылает команду, и отжатие посылает команду). Кнопка позволяет послать любое число. По умолчанию она шлет 0 или 1 (LOW или HIGH). Так, в пуш-режиме кнопка шлет 1 (HIGH) при нажатии и 0 (LOW) при отжатии. Для кнопки необходимо выбрать пин (контакт) — реальный или виртуальный, на который отправляются значения HIGH и LOW.

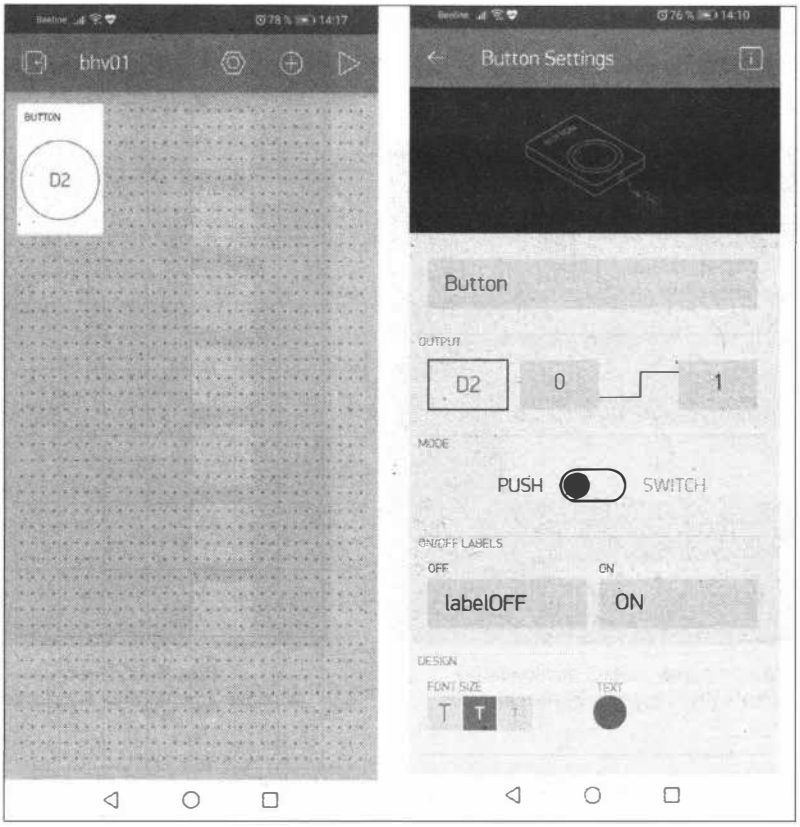


Рис. 5.78. Виджет Button

Состояние кнопки можно менять с микроконтроллера. Например, включить кнопку на пине V1 можно так:

```
Blynk.virtualWrite(V1, HIGH);
```

Так можно поменять тексты в кнопке:

```
Blynk.setProperty(V1, "onLabel", "Вкл");
Blynk.setProperty(V1, "offLabel", "Выкл");
```

сменить название самой кнопки:

```
Blynk.setProperty(V1, "label", "Моя кнопочка");
```

или изменить ее цвет :

```
//#D3435C - Blynk RED
Blynk.setProperty(V1, "color", "#D3435C");
```

В случае, если микроконтроллер был перезагружен, получить последнее состояние кнопки с сервера можно с помощью синхронизации состояния:

```
//как только подключились
BLYNK_CONNECTED() {
  //запросить информацию у сервера о состоянии пина V1
  Blynk.syncVirtual(V1);
}
//этот метод будет вызыван после ответа сервера
BLYNK_WRITE(V1) {
  int buttonState = param.asInt();
}
```

Виджеты *Slider* и *Vertical Slider*

Виджеты **Slider** (Слайдер) (рис. 5.79) и **Vertical Slider** (Вертикальный слайдер) очень похожи на потенциометр. Слайдер позволяет посылать значения в диапазоне

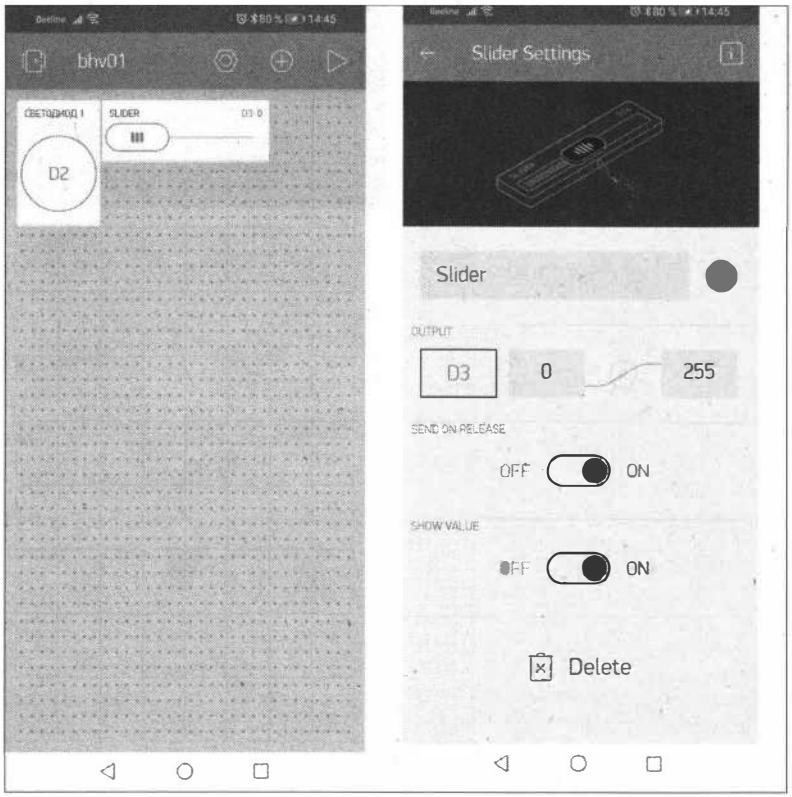


Рис. 5.79. Виджет Slider

от минимального значения к максимальному. Диапазон допустимых максимального и минимального значений определяется в приложении. Слайдер можно использовать для регулирования яркости светодиода или управления скоростью двигателя.

Состояние слайдера можно менять с микроконтроллера. Например, изменить положение ползунка в слайдере можно так:

```
Blynk.virtualWrite(V1, 55);
```

Так можно поменять текст в слайдере:

```
Blynk.setProperty(V1, "label", "Мой слайдерок");
```

или изменить его цвет :

```
//#D3435C - Красный цвет в кодировке RGB
Blynk.setProperty(V1, "color", "#D3435C");
```

Виджет *Value Display*

Виджет **Value Display** (Отображение значений) (рис. 5.80) отображает входящие данные с датчиков или виртуальных пинов.

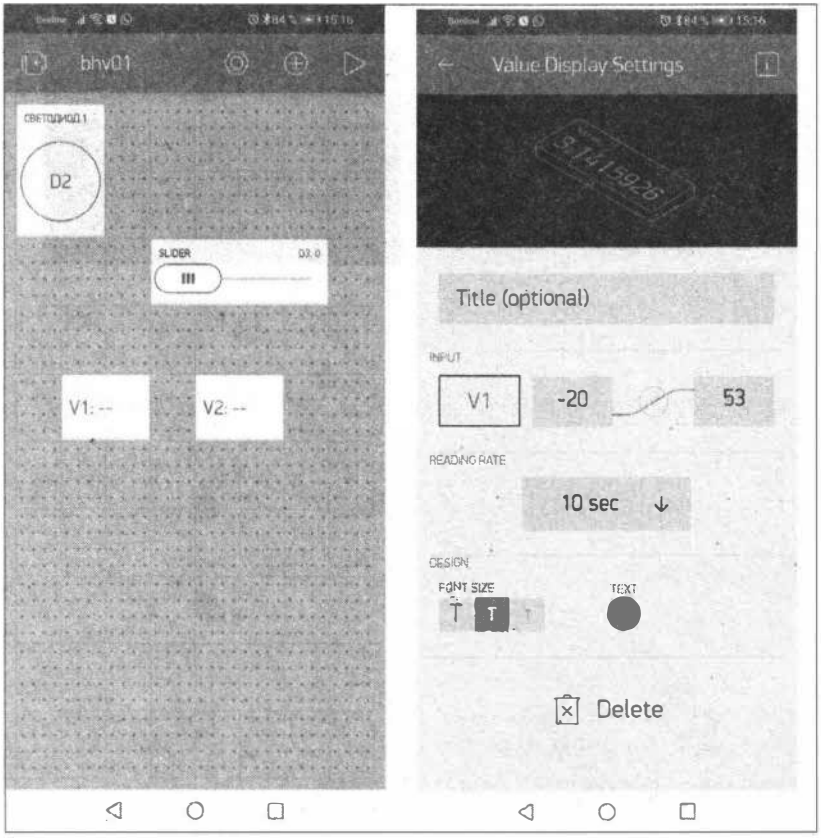


Рис. 5.80. Виджет Value Display

Виджет может работать в двух режимах:

- режим PUSH;
- режим частоты считываний.

В режиме PUSH обновление значения виджета происходит со стороны оборудования с помощью кода:

```
Blynk.virtualWrite(V1, val);
```

В этом режиме каждое сообщение, которое отправляет аппаратное устройство, автоматически сохраняется на сервере. Режим PUSH не требует, чтобы приложение было онлайн или открыто.

В режиме частоты считывания необходимо выбрать интервал обновления данных, и приложение будет запускать события считывания с требуемой периодичностью. Приложение должно быть открыто и запущено для отправки запросов на оборудование. Для аналоговых и цифровых выводов в этом случае код не нужен. Однако для виртуальных выводов необходимо использовать следующий код:

```
//вызывать из приложения
BLYNK_READ(V1)
{
  //отправить в приложение
  Blynk.virtualWrite(V1, val);
}
```

Виджет *Gauge*

Виджет **Gauge** (Указатель) (рис. 5.81) — отличный визуальный способ отображения входящих числовых значений.

Виджет может работать в двух режимах:

- режим PUSH;
- режим частоты считываний.

В режиме PUSH обновление значения виджета происходит со стороны оборудования с помощью кода:

```
Blynk.virtualWrite(V1, val);
```

В этом режиме каждое сообщение, которое отправляет аппаратное устройство, автоматически сохраняется на сервере. Режим PUSH не требует, чтобы приложение было онлайн или открыто.

В режиме частоты считывания необходимо выбрать интервал обновления данных, и приложение будет запускать события считывания с требуемой периодичностью. Приложение должно быть открыто и запущено для отправки запросов на оборудование. Для аналоговых и цифровых выводов в этом случае код не нужен. Однако для виртуальных выводов вам необходимо использовать следующий код:

```
//вызывать из приложения
BLYNK_READ(V1)
```

```
{
  //отправить в приложение
  Blynk.virtualWrite(V1, val);
}
```

Указатель также имеет поле `Label` (Метка), которое позволяет использовать форматирование. Предположим, ваш датчик отправляет число 12.6789 в приложение Blynk. При этом поддерживаются следующие параметры форматирования:

- `/pin/` — отображает значение без форматирования (12.6789);
- `/pin./` — отображает значение без десятичной части (13);
- `/pin.#/` — отображает значение с одним десятичным знаком (12.7);
- `/pin.##/` — отображает значение с двумя десятичными знаками (12.68).

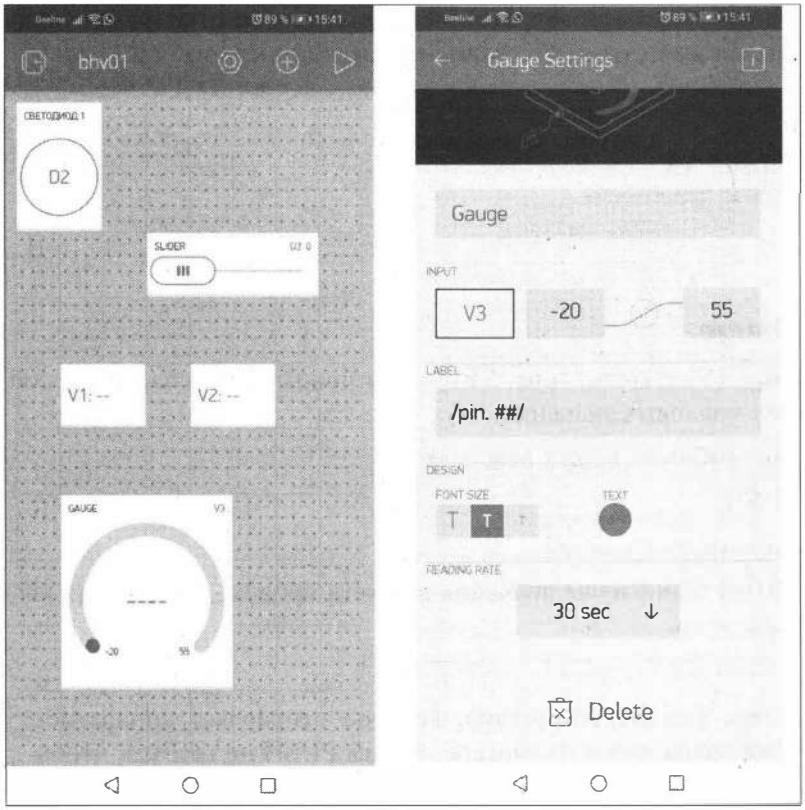


Рис. 5.81. Виджет Gauge

Виджет *Timer*

Виджет **Timer** (Таймер) (рис. 5.82) запускает действия в определенное время, даже если планшет (смартфон) не в сети. По умолчанию время начала отправляет 1 (HIGH), время остановки отправляет 0 (LOW). Можно изменить это поведение на

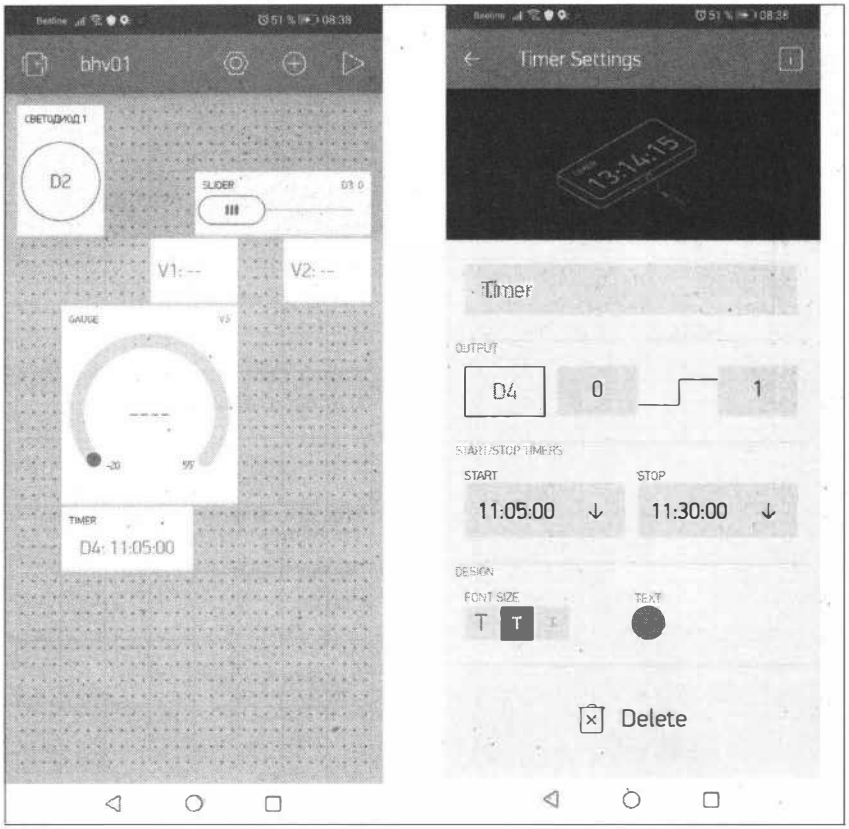


Рис. 5.82. Виджет **Timer**

любые другие значения. В последней версии Android поддерживается улучшенный таймер в составе описанного далее виджета Eventor (Обработчик событий).

Виджет *Eventor*

Виджет **Eventor** (Обработчик событий) (рис. 5.83) позволяет создавать простые правила поведения или события. Например, при изменении значения на назначенном виртуальном пине отправить уведомление по почте. Обработчик событий также поддерживает события таймера. Так, вы можете установить пин V1 ON/HIGH в 21:00:00 каждую пятницу. В обработчике событий можно назначить несколько таймеров на один и тот же пин, отправить любую строку/число, выбрать день и часовой пояс.

Виджет *Webhook*

Еще один очень полезный виджет — **Webhook.** Он предназначен для связи со сторонними сервисами. С помощью виджета Webhook можно отправлять HTTP(S) запросы на любую стороннюю службу или устройство, имеющее HTTP(S) API. Активировать стороннюю службу при этом можно одним нажатием кнопки.

Рис. 5.83. Виджет Eventor

Любая операция `write` со стороны оборудования вызовет запуск виджета Webhook. Вы также можете активировать из приложения Blynk веб-перехватчик, когда виджету приложения назначен тот же пин, что и веб-перехватчику. Например, если вам нужно отправить данные со своего оборудования не только в Blynk, но и в ThingSpeak, потребуется написать длинный код HTTP-запроса, подобный этому:

```
WiFiClient client;
if (client.connect("api.thingspeak.com", 80)) {
    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + apiKeyThingspeak1 + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
}
```

Вместо этого с виджетом Webhook вам нужно будет только заполнить в виджете необходимые поля (рис. 5.84).

Рис. 5.84. Виджет WebHook

И добавить этот код на аппаратной стороне:

```
Blynk.virtualWrite(V0, value);
```

где V0 — пин, назначенный виджету Webhook.

В URL-адресе используются стандартные заполнители Blynk для значения пина, например:

```
https://api.thingspeak.com/update?api_key=xxxxxx&field1=/pin/
```

Когда вам нужно отправить массив значений, вы можете обратиться к определенному индексу значения массива. Blynk Pin может содержать массив максимум из 10 значений:

```
/pin[0]/,/pin[1]/, /pin[2]/
```

Также можно делать от Blynk Server запросы GET и получать ответы непосредственно на свое оборудование. Например, чтобы получить текущую погоду от сторонней службы погоды, использующей URL-адрес, подобный следующему:

http://api.sunrise-sunset.org/json?lat=33.3823&lng=35.1856&date=2020-10-01

нужно будет поместить этот URL-адрес в виджет Webhook и назначить его закрепление пину V0.

А чтобы проанализировать ответ на аппаратной стороне, потребуется следующий код:

```
BLYNK_WRITE(V0){
  String webhookdata = param.asStr();
  Serial.println(webhookdata);
}
```

Теперь каждый раз, когда появляется команда записи для закрепления V0, — например:

```
Blynk.virtualWrite(V0, 1)
```

будет запускаться и обрабатываться конструкция `BLYNK_WRITE(V0)`.

Виджет *SuperChart*

Виджет **SuperChart** (Диаграмма) (рис. 5.85) можно использовать для создания графиков, для живой визуализации и хранения данных, для логгирования данных датчиков, бинарных событий и многого другого.

Чтобы задействовать виджет SuperChart, необходимо передавать данные с оборудования с желаемым интервалом, используя таймеры.

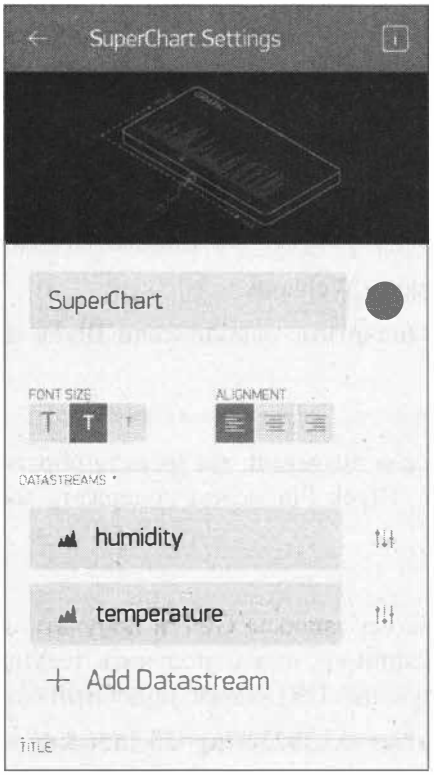


Рис. 5.85. Виджет SuperChart

Виджет поддерживает до четырех потоков данных. Для каждого потока можно настроить вид диаграммы (Line, Area, Bar, Binary). Необходимо также назначить пин, привязанный к потоку, минимальное и максимальное значение по оси Y, округление выводимых данных (рис. 5.86).

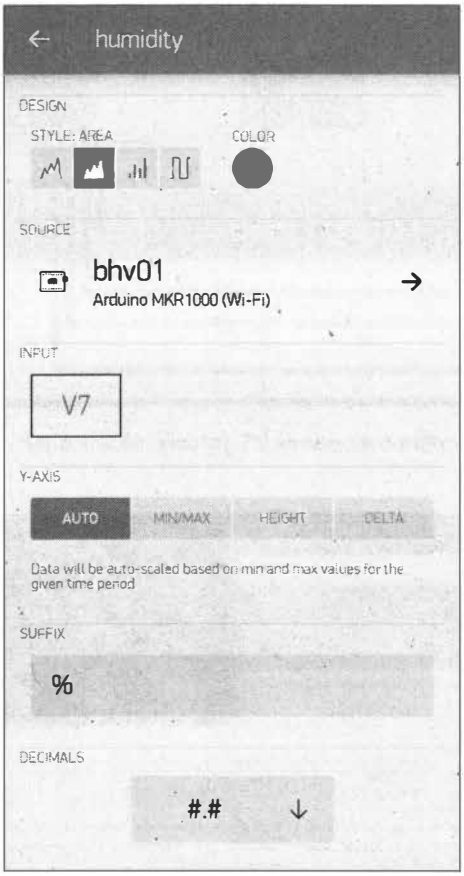


Рис. 5.86. Настройка параметров виджета SuperChart

5.4.3. Создание скетчей на Arduino для устройства IoT Blynk

В качестве IoT-устройства Blynk мы воспользуемся платой Arduino Nano 33 IoT с подключенными к ней датчиками, светодиодами и реле. Монтажная схема соединений показана на рис. 5.87.

Для работы с этим устройством необходимо установить Arduino-библиотеку Blynk. Зайдите в Менеджер библиотек (**Скетч | Подключить библиотеку | Управлять библиотеками**), в поиске найдите **Blynk** и нажмите кнопку **Install** — библиотека установлена (рис. 5.88).

Загрузите в плату Arduino Nano 33 IoT скетч из листинга 5.9. Замените предварительно в скетче данные точки доступа (`ssid[]` и `pass[]`) и токена авторизации `auth[]` на свои.

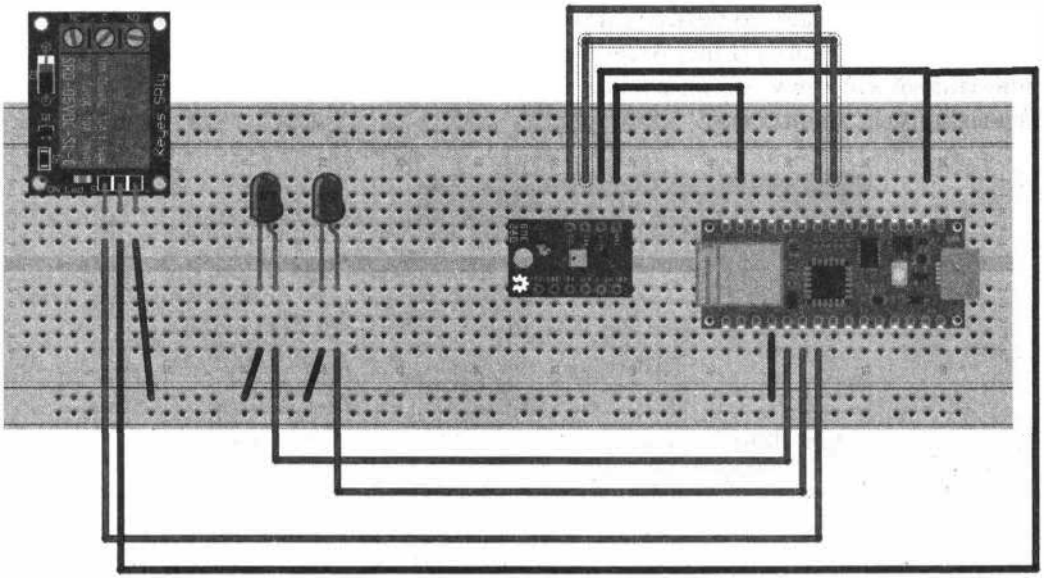


Рис. 5.87. Монтажная схема соединений IoT-устройства для Blynk

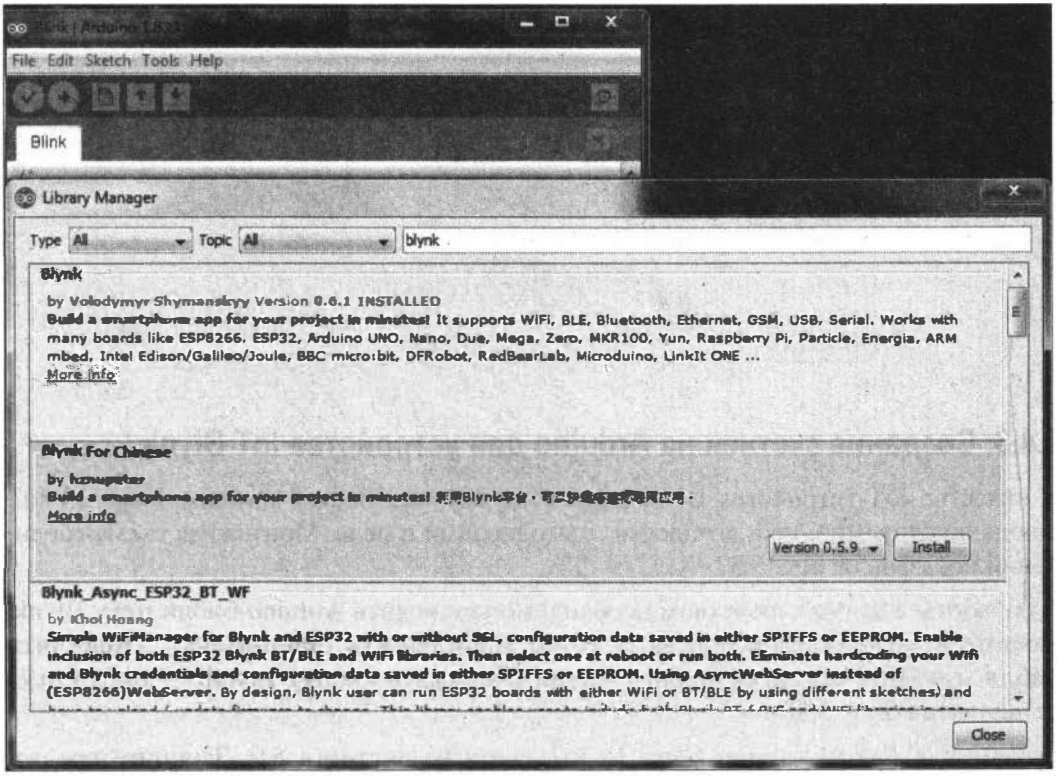


Рис. 5.88. Установка библиотеки Blynk в Arduino IDE

Листинг 5.9

```
// вывод отладки в монитор последовательного порта
#define BLYNK_PRINT Serial

// Подключение библиотек
#include <SPI.h>
#include <WiFiNINA.h>
#include <BlynkSimpleWiFiNINA.h>

// токен авторизации для проекта
char auth[] = "OlFJb_KL_wh8DV0CiPrDuVvUe7_iEc0B";

// данные точки доступа Wi-Fi
char ssid[] = "Kassal";
char pass[] = "12345678";

void setup()
{
  // Запуск последовательного порта
  Serial.begin(9600);
  // Запуск blynk
  Blynk.begin(auth, ssid, pass);
}

void loop()
{
  // Вот и вся программа..
  Blynk.run();
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.9, можно найти в папке *examples\05\05_09* сопровождающего книгу электронного архива (см. *приложение*).

Загрузив скетч, откройте монитор последовательного порта. При подключении платы к серверу **blynk-cloud.com** в мониторе последовательного порта вы увидите вывод, показанный на рис. 5.89.

Запустите на планшете программу, и вы сможете управлять элементами, связанными с реальными пинами: D2 — Button, D3 — Slider и срабатывание таймера на D4. Если виджеты связаны с виртуальными пинами (в нашей программе: V1 — ValueDisplay, V3 — Gauge, V5 — Eventor с отправкой уведомления на e-mail), то в скетч необходимо внести изменения:

- V1 — отображение влажности с датчика BME280;
- V3 — отображение температуры с датчика BME280.

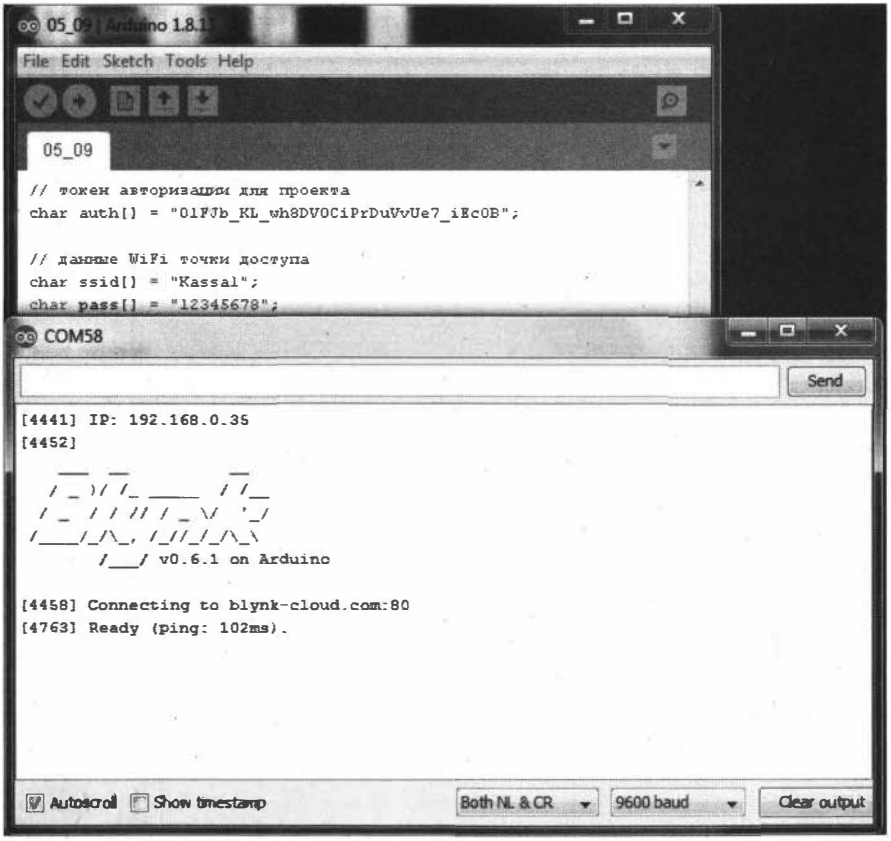


Рис. 5.89. Подключение платы к серверу **blynk-cloud.com**

Загрузите в плату Arduino Nano 33 IoT скетч из листинга 5.10. Замените предварительно в скетче данные точки доступа (`ssid[]` и `pass[]`) и токена авторизации `auth[]` на свои.

Листинг 5.10

```
// вывод отладки в монитор последовательного порта
#define BLYNK_PRINT Serial

// Подключение библиотек
#include <SPI.h>
#include <WiFiNINA.h>
#include <BlynkSimpleWiFiNINA.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme; // I2C
// токен авторизации для проекта
char auth[] = "OlFJb_KL_wh8DV0CiPrDuVvUe7_iEc0B";
```

```
// данные точки доступа Wi-Fi
char ssid[] = "Kassa1";
char pass[] = "12345678";

void setup()
{
  // Запуск последовательного порта
  Serial.begin(9600);

  bool status;
  status = bme.begin();
  if (!status) {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      while (1);
  }
  // Запуск blynk
  Blynk.begin(auth, ssid, pass);
}

BLYNK_READ(V1)
{
  //отправить в приложение
  Blynk.virtualWrite(V1, bme.readHumidity());
}

BLYNK_READ(V3)
{
  //отправить в приложение
  Blynk.virtualWrite(V3, bme.readTemperature());
}

void loop()
{
  // Вот и вся программа..
  Blynk.run();
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.10, можно найти в папке *examples\05\05_10* сопровождающего книгу электронного архива (см. *приложение*).

Вызов функций `BLYNK_READ(V1)` и `BLYNK_READ(V3)` будет осуществляться с периодичностью, установленной в свойствах виджетов Value Display и Gauge. Вид экрана планшета показан на рис. 5.90.

Рассмотрим отправку данных в виджет SuperChart для построения графиков (с настройкой виджета SuperChart мы познакомились в *разд. 5.4.2*). Виджет будет строить графики влажности и температуры по данным, получаемым с датчика BME280. Отправка данных будет производиться по таймеру.

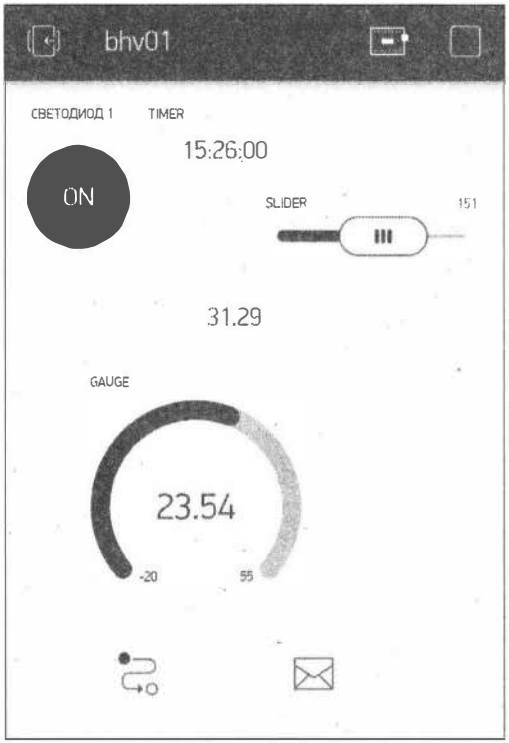


Рис. 5.90. Работа приложения на экране планшета

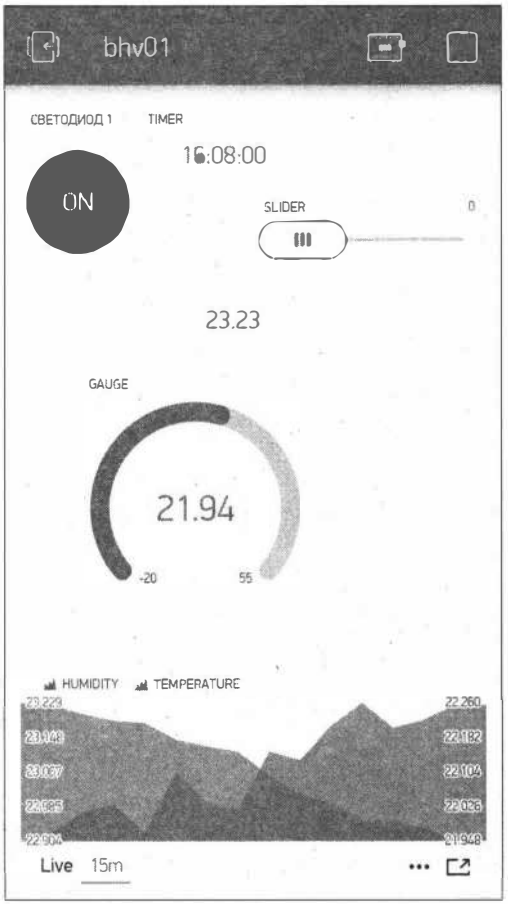


Рис. 5.91. Работа приложения на экране планшета

Если виджеты связаны с виртуальными пинами, то в скетч необходимо внести изменения:

- V7 — отображение влажности с датчика BME280;
- V8 — отображение температуры с датчика BME280.

Загрузите в плату Arduino Nano 33 IoT скетч из листинга 5.11. Замените предварительно в скетче данные точки доступа (`ssid[]` и `pass[]`) и токена авторизации `auth[]` на свои. Вид экрана планшета показан на рис. 5.91.

Листинг 5.11

```
// вывод отладки в монитор последовательного порта
#define BLYNK_PRINT Serial

// Подключение библиотек
#include <SPI.h>
```

```
#include <WiFiNINA.h>
#include <BlynkSimpleWiFiNINA.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme; // I2C
// токен авторизации для проекта
char auth[] = "OlFJb_KL_wh8DV0CiPrDuVvUe7_iEc0B";

// данные точки доступа Wi-Fi
char ssid[] = "Kassa1";
char pass[] = "12345678";

BlynkTimer timer1;

void setup()
{
  // Запуск последовательного порта
  Serial.begin(9600);

  bool status;
  status = bme.begin();
  if (!status) {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      while (1);
  }
  // Запуск blynk
  Blynk.begin(auth, ssid, pass);
  // запуск таймера
  timer1.setInterval(15000L, timer1Event);
}

void timer1Event()
{
  //отправить в приложение
  Blynk.virtualWrite(V7, bme.readHumidity());
  Blynk.virtualWrite(V8, bme.readTemperature());

}
BLYNK_READ(V1)
{
  //отправить в приложение
  Blynk.virtualWrite(V1, bme.readHumidity());
}
```

```
BLYNK_READ(V3)
{
  //отправить в приложение
  Blynk.virtualWrite(V3, bme.readTemperature());
}

void loop()
{
  // Вот и вся программа..
  Blynk.run();
  // инициализация таймера
  timer1.run();
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 5.11, можно найти в папке *examples\05\05_11* сопровождающего книгу электронного архива (см. *приложение*).

А вот пример использования виджета WebHook. Пусть мы отправляем данные температуры с датчика BME280, подключенного к плате Arduino_Nano_33_IoT, в канал ThingSpeak (см. *разд. 5.3*). Отправка идет в поле `field1`. Адрес, который необходимо ввести в поле адреса виджета:

https://api.thingspeak.com/update?api_key=RKVTWSNB3NH7FVXC&field1=/pin/

Выбираем виртуальный пин V8. На IoT-устройстве загружен скетч из листинга 5.11. После загрузки скетча запускаем приложение на смартфоне, через некоторое время заходим на страницу ThingSpeak (канал Arduino_Nano_33_IoT) и видим данные, поступающие с IoT-устройства через приложение Blynk (рис. 5.92).

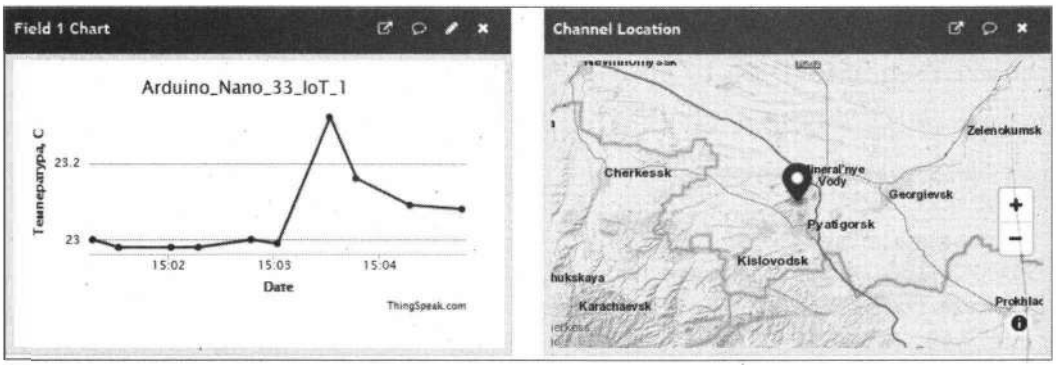


Рис. 5.92. Отправка данных из Blynk в ThingSpeak с помощью виджета WebHook

Здесь мы рассмотрели базовые возможности сервиса Blynk. Вы можете включать в свои проекты и другие виджеты, а подробную документацию по использованию виджетов вы сможете найти на сайте проекта.

5.5. The Things Network (TTN) — открытая LoRaWAN сеть

Сервис The Things Network (**www.thethingsnetwork.org**) — открытая LoRaWAN сеть, поддерживаемая и развиваемая ее участниками. Сервис предоставляет набор открытых инструментов и глобальную открытую сеть для создания IoT-приложений. Благодаря надежному сквозному шифрованию создается защищенная и совместная сеть Internet of Things, охватывающая многие страны мира. Сейчас работают тысячи шлюзов, обеспечивающих покрытие миллионам людей.

Любой желающий может зарегистрировать в сети свое LoRa-устройство как узел (node) — если этот узел может подключиться к существующему шлюзу. Карту шлюзов можно посмотреть на главной странице проекта **www.thethingsnetwork.org/** (рис. 5.93). Если же возможности подключиться к существующему шлюзу нет, необходимо зарегистрировать свой шлюз (gateway).

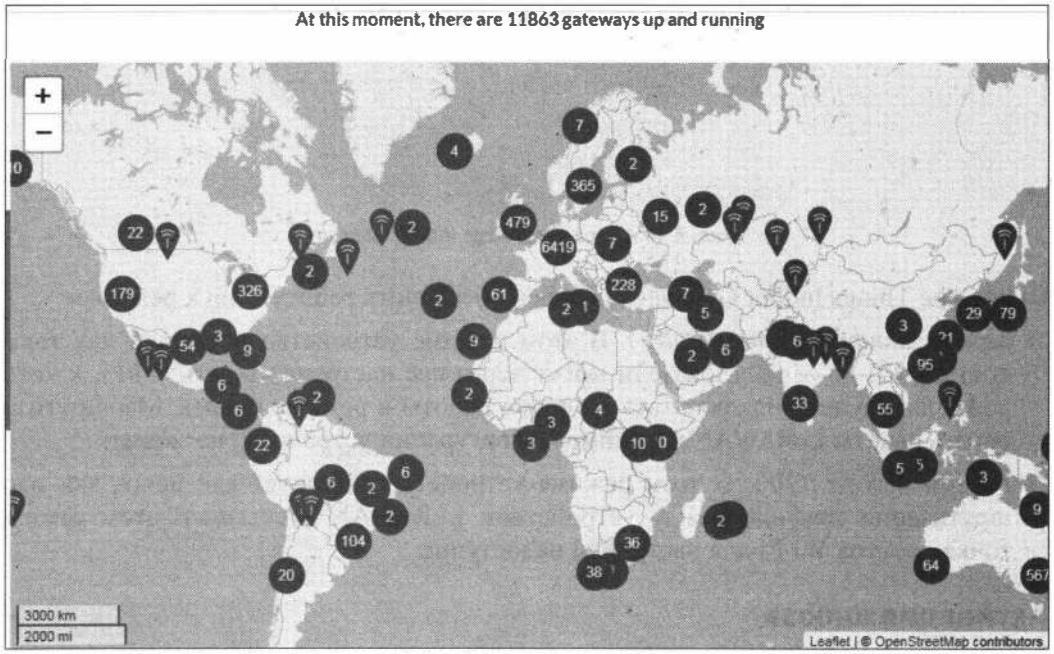


Рис. 5.93. Карта открытых шлюзов сети TTN

5.5.1. Шлюз The Things IndoorGateway

Один из вариантов шлюза — устройство The Things Indoor Gateway (рис. 5.94).

Его характеристики:

- 8-канальный внутренний шлюз LoRaWAN;
- поддержка современного протокола BasicStation;

- встроенная всенаправленная антенна для использования внутри помещений;
- возможность подключиться к любой сети по выбору;
- простые шаги установки, занимающие менее 5 минут;
- настройка и подключение через Wi-Fi;
- питание через кабель USB-C;
- доступны версии EU868, US915, AS923 и CN470.

Рис. 5.94. Шлюз The Things Indoor Gateway

Шлюз The Things Indoor Gateway способен обеспечить работу в двух режимах:

- *режим конфигурации (CONF)*. В этом режиме устройство действует как точка доступа Wi-Fi — пользователи могут через нее настраивать сеть Wi-Fi, к которой шлюз будет подключаться во время работы в режиме шлюза. Маршрутизировать пакеты LoRaWAN в режиме конфигурации устройство не может;
- *режим шлюза (GW)*. В этом режиме устройство действует как шлюз для маршрутизации трафика между устройством LoRaWAN и сетью. В этом режиме точка доступа Wi-Fi для настройки недоступна.

Активация шлюза

Активация шлюза The Things Indoor Gateway выполняется следующим образом:

1. Нажмите кнопку сброса (маленькая кнопка на задней панели шлюза рядом с портом USB-C) и удерживайте ее в течение 5 секунд, пока индикатор не начнет попеременно быстро мигать *зеленым* и *красным* цветом.
2. Удерживайте кнопку SETUP (кнопка в верхней части шлюза рядом со светодиодом) в течение 10 секунд, пока светодиод не начнет быстро мигать *красным* цветом.
   - Теперь шлюз представляет собой точку доступа Wi-Fi, SSID которой MINIHUB-*xxxxxx*, где *xxxxxx* — это последние 6 цифр идентификатора шлюза.

- Пароль для этой сети напечатан на задней панели устройства под надписью **WiFi PW** (Wi-Fi password).

3. После подключения к сети Wi-Fi перейдите с помощью веб-браузера по адресу 192.168.4.1 для доступа к странице конфигурации Wi-Fi (рис. 5.95).
4. Выберите сеть Wi-Fi и введите пароль, если это закрытая сеть.
5. Нажмите кнопку **SAVE & REBOOT**.

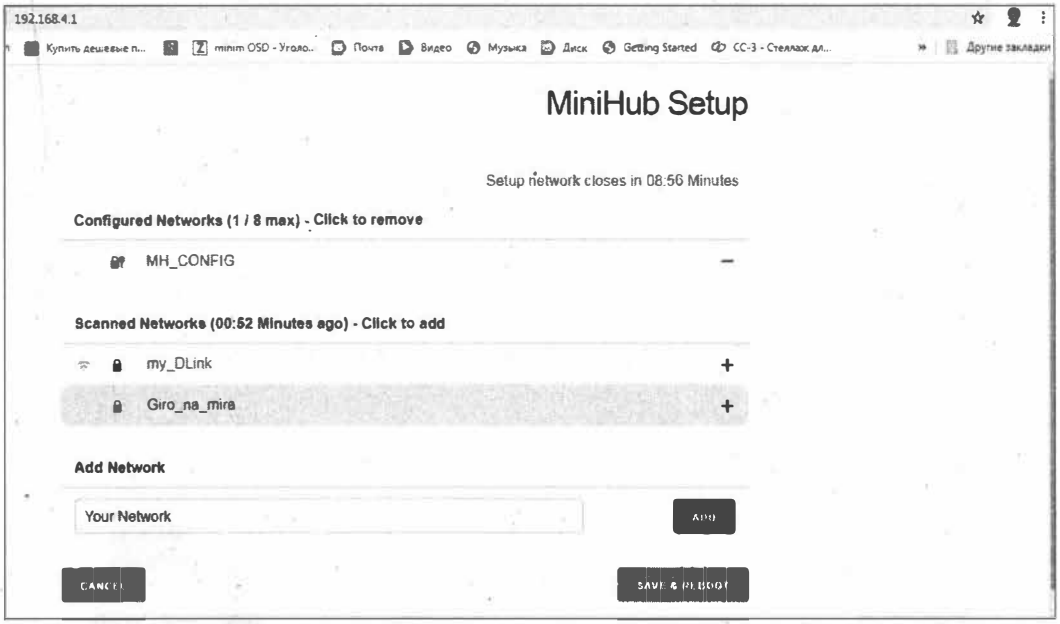


Рис. 5.95. Страница конфигурации шлюза The Things Indoor Gateway

Если ваши настройки выполнены правильно:

- ❒ индикатор шлюза будет мигать *зеленым* цветом в течение нескольких секунд, пока он подключается к сети;
- ❒ затем он начнет попеременно мигать *зеленым* и *красным* цветом в течение нескольких секунд, пока он подключается к конечной точке CUPS и выбирает необходимую информацию для подключения к конечной точке трафика LNS.

После завершения конфигурации индикатор будет гореть зеленым цветом, означающим, что шлюз подключен к сети LoRaWAN и готов к обработке пакетов.

Подключение к сервису The Things Network

Для подключения шлюза к сервису The Things Network в этом сервисе сначала нужно зарегистрироваться, после чего войти в свой профиль и зарегистрировать шлюз. Для регистрации шлюза нажмите на кнопку **Setup a gateway** (рис. 5.96).

Чтобы подключить свой шлюз к консоли The Things Network, потребуется значение **Gateway EUI**, которое можно получить на странице настройки Wi-Fi (рис. 5.97).

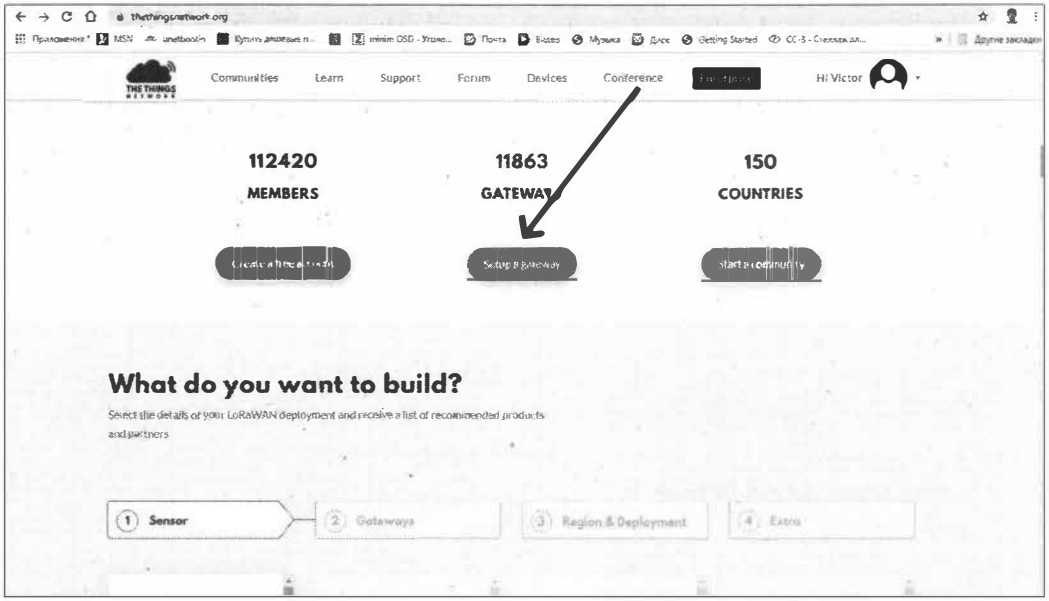


Рис. 5.96. Регистрация шлюза в сервисе The Things Network

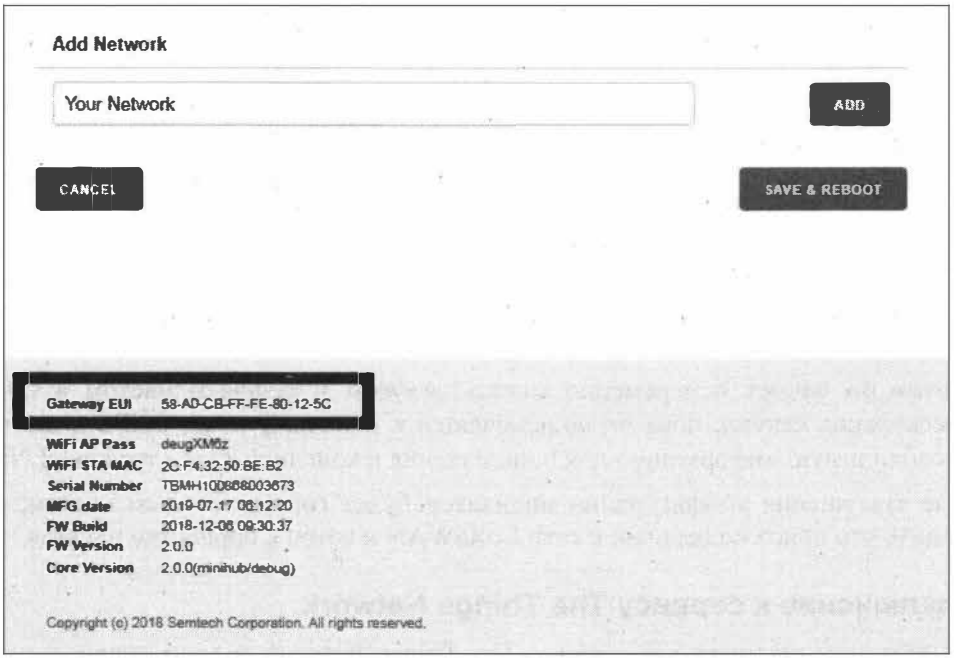


Рис. 5.97. Значение **Gateway EUI** на странице настройки Wi-Fi

Получив это значение, введите его на консоли (рис. 5.98) в поле **Gateway EUI**. Введите там и другие данные — такие, как **Frequency Plan** (Частотный план), **Router** (Маршрутизатор) и **Location** (Местоположение), и зарегистрируйте шлюз, используя опцию **I'm using the Legacy Packet Forwarder**. Если все сделано правильно, вы увидите сообщение **Status connected** (рис. 5.99). Теперь шлюз подключен к сервису, и вы можете подключаться к нему с устройств LoRa.

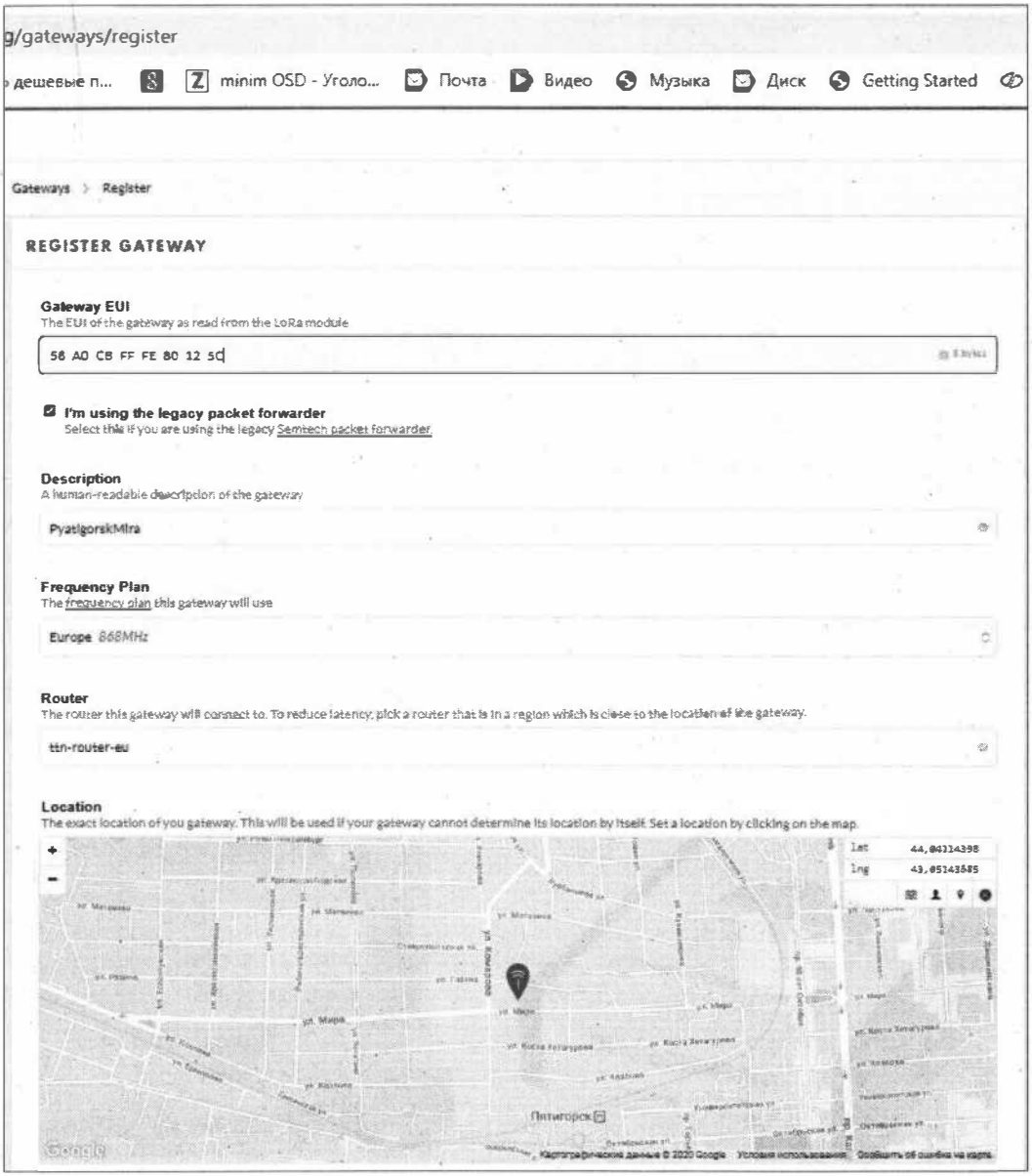


Рис. 5.98. Заполнение данных шлюза

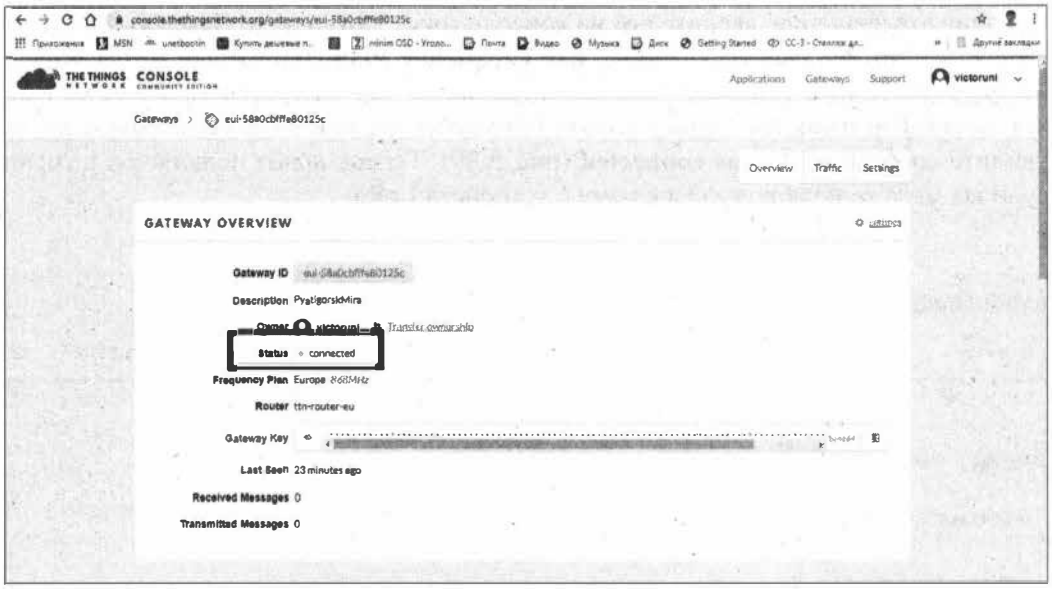


Рис. 5.99. Статус шлюза в консоли The Things Network

5.5.2. Регистрация устройства на основе платы The Things Uno в сервисе TTN

Компания Things Network предоставляет совместимые с экосистемой Arduino аппаратные средства, которые позволяют легко развертывать сети LoRaWAN. В настоящее время компания продвигает совместимую с Arduino плату The Things Uno (упомяную нами в *разд. 2.1*). Плата The Things Uno (рис. 5.100) базируется на микроконтроллере ATmega32u4, т. е., по сути, она основана на платформе Arduino Leonardo, а не Arduino Uno. Плата несет на борту приемопередатчик RN2483 компании Microchip для связи по сети LoRaWAN, у нее также имеется встроенная печатная антенна для обеспечения лучшей связи.

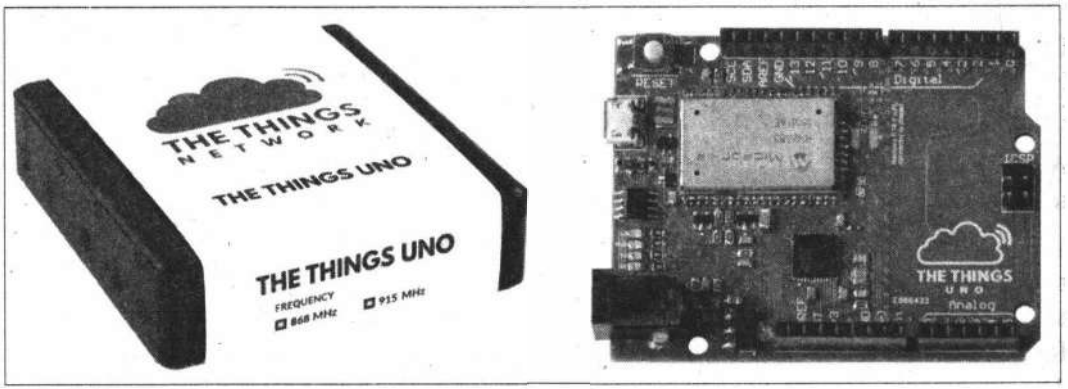


Рис. 5.100. Arduino-совместимая плата The Things Uno

Чтобы получить характерные для платформы Arduino простые в использовании методы подключения к сети и отправки данных, плате The Things Uno потребуется библиотека TheThingsNetwork. Библиотеку можно установить через Менеджер библиотек, как показано на рис. 5.101.

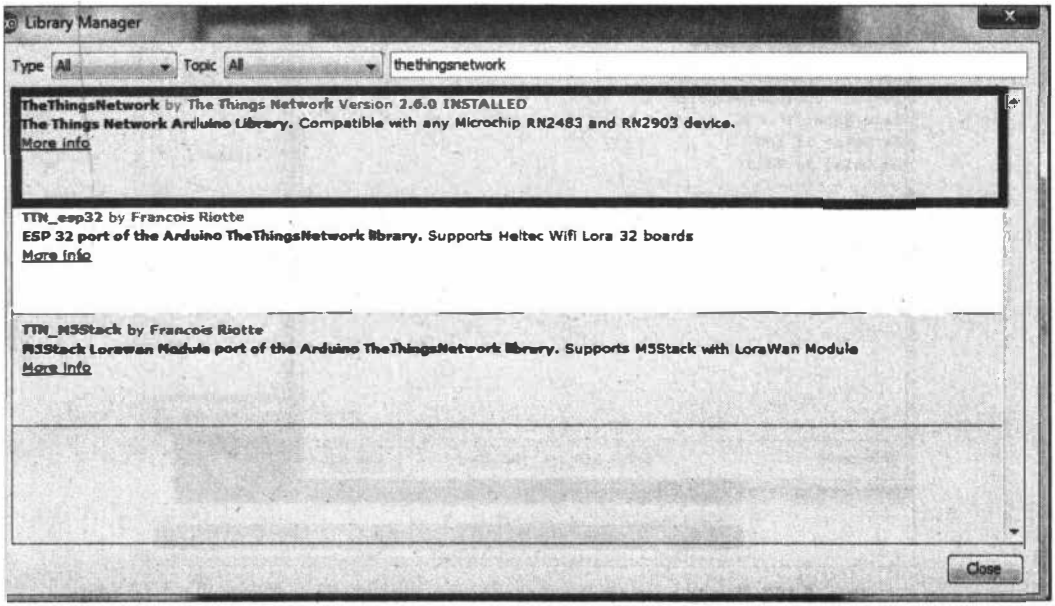


Рис. 5.101. Установка библиотеки TheThingsNetwork через Менеджер библиотек

Загрузите в плату The Things Uno (выбрав в среде Arduino IDE плату Arduino Leonardo) пример DeviceInfo из библиотеки TheThingsNetwork, предварительно внеся в скетч примера следующие изменения:

```
// Replace REPLACE_ME with TTN_FP_EU868 or TTN_FP_US915
#define freqPlan TTN_FP_EU868
```

Загрузив скетч в плату The Things Uno, откройте монитор последовательного порта. В него выводится необходимая нам информация об EUI устройства (рис. 5.102). Эта информация нам чуть позже пригодится.

Затем зайдите в сервисе в консоль The Things Network (рис. 5.103) и выберите **Applications**, а затем **Add application**. Заполните в открывшемся окне (рис. 5.104) необходимые поля поля и нажимите на кнопку **Add Application**.

Теперь к созданному приложению необходимо добавить устройство — нажимите для этого на ссылку **register device** (рис. 5.105), в открывшемся окне (рис. 5.106) введите данные своего устройства (вот где нужен его EUI) и нажмите на кнопку **Register**. Для отправки данных в дальнейшем нам понадобятся два значения: **Application EUI** и **App Key,** которые вы найдете в следующем открывшемся окне (рис. 5.107).

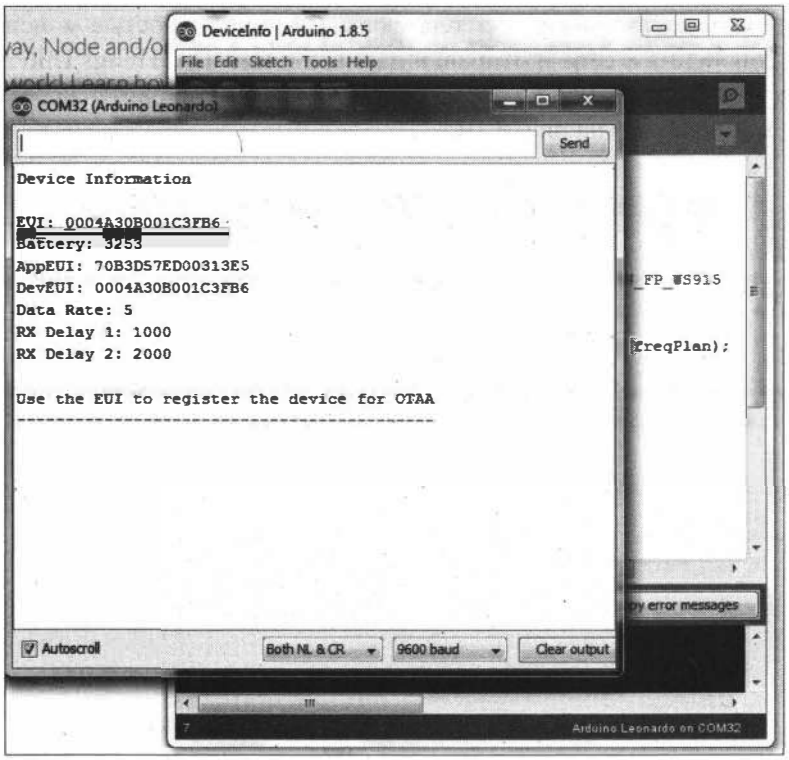


Рис. 5.102. Вывод в монитор последовательного порта информации об устройстве The Things Uno

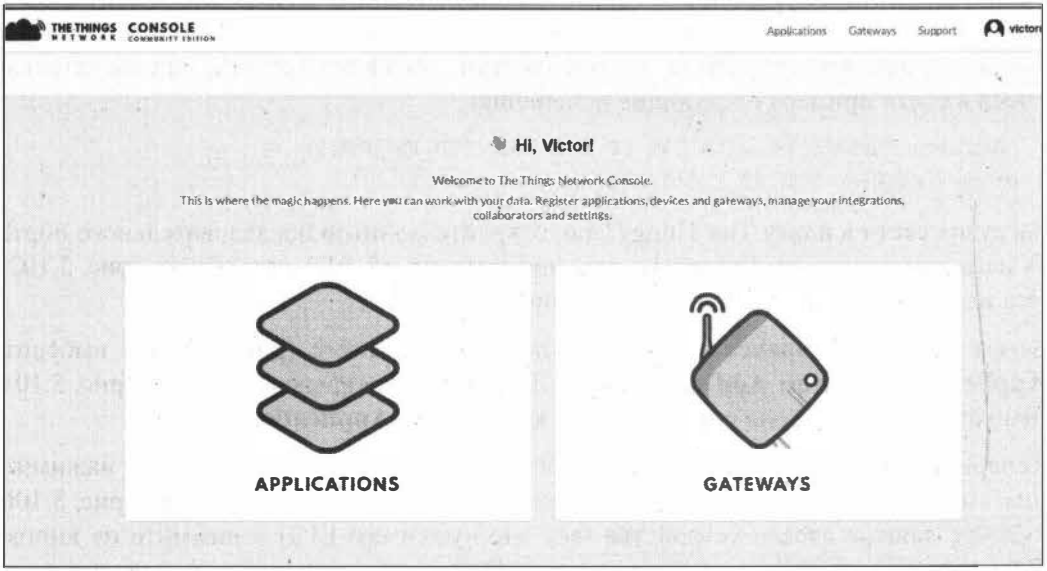


Рис. 5.103. Добавление приложения в консоли The Things Network

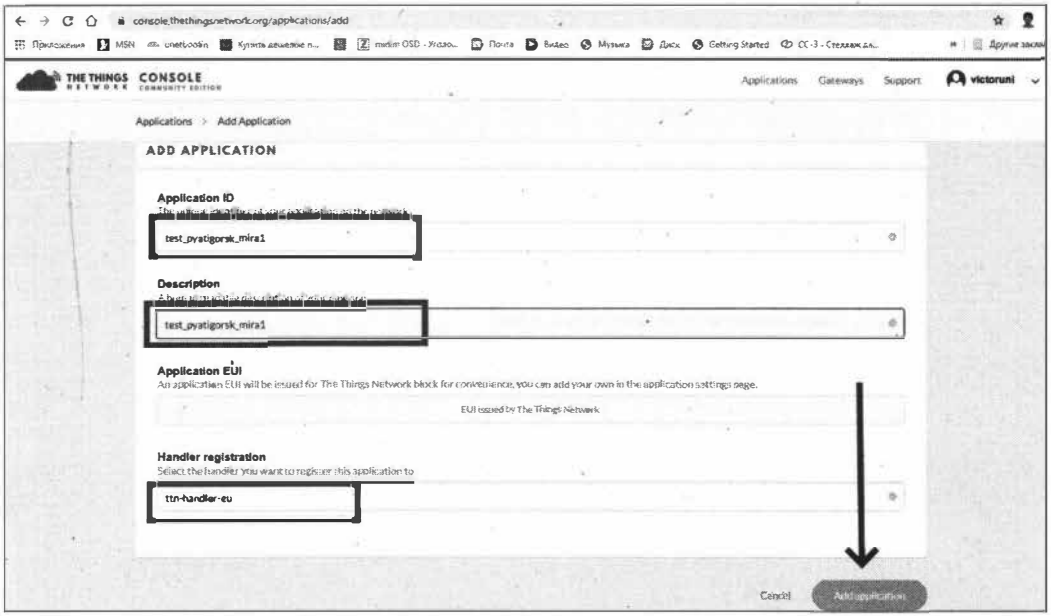


Рис. 5.104. Создание приложения в консоли The Things Network

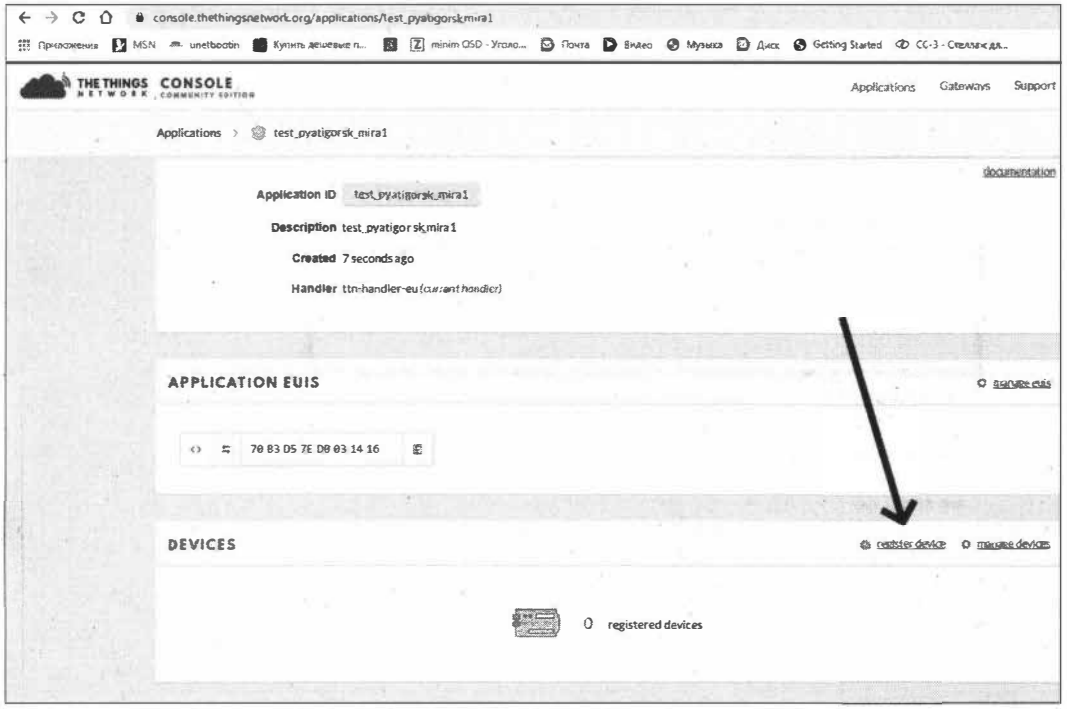


Рис. 5.105. Ссылка **register device**

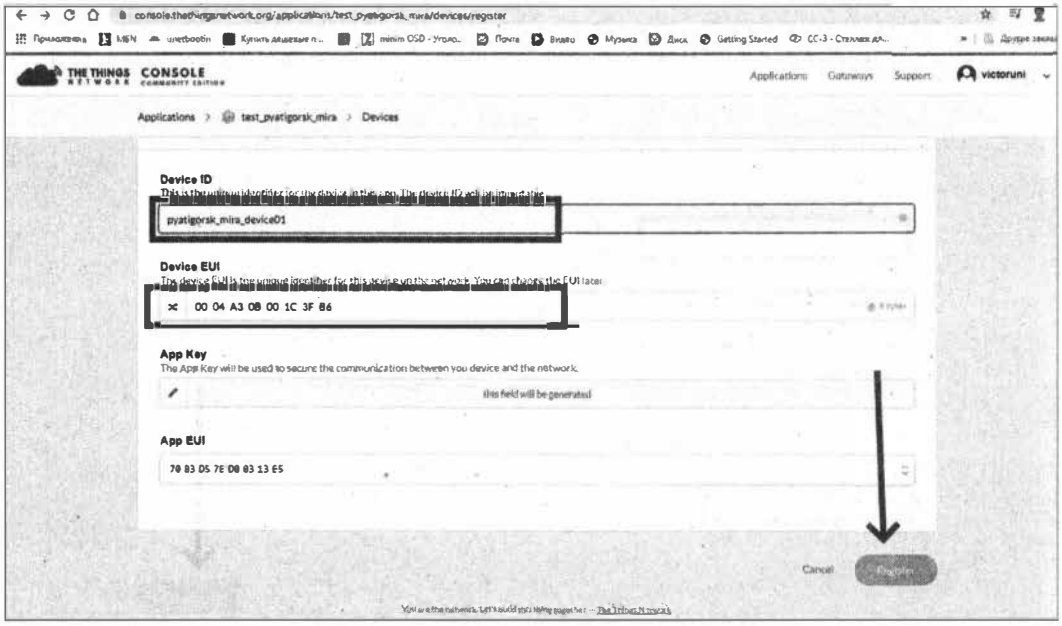


Рис. 5.106. Добавление устройства к приложению

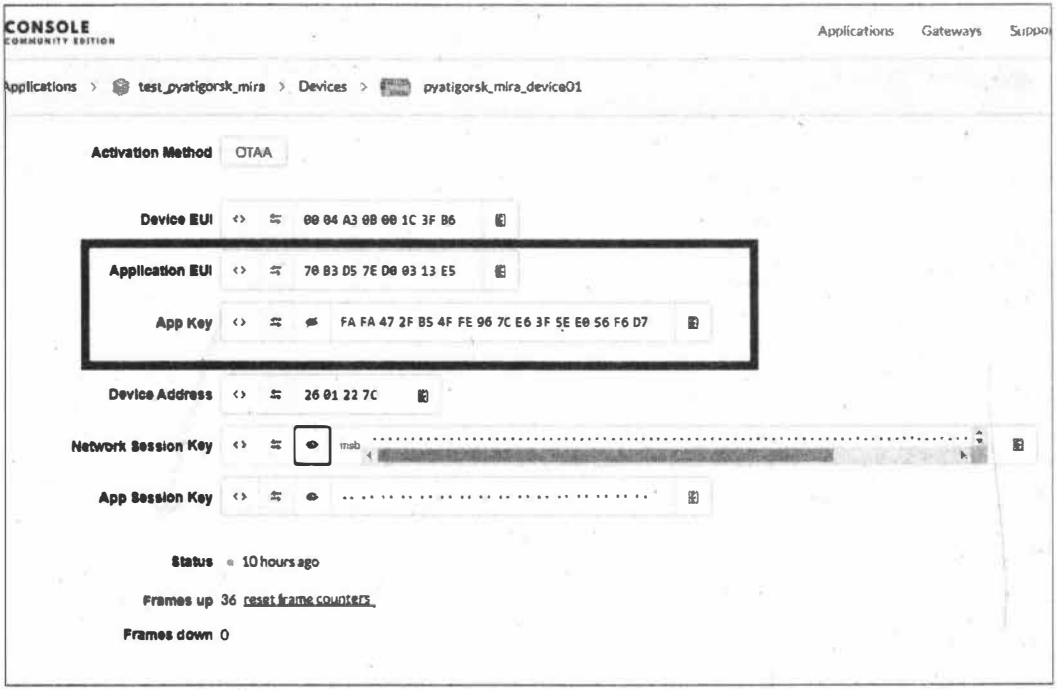


Рис. 5.107. Данные для отправки данных с устройства в приложение

5.5.3. Отправка данных в сервис The Things Network

Рассмотрим отправку данных с устройства The Thing Uno в сервис The Things Network. Для этого загрузите в плату Things Uno пример SendOTAA из библиотеки TheThings Network. Предварительно внесите в скетч примера изменения в следующих строках (впишите свои данные EUI устройства и AppKey приложения в сервисе):

```
const char *appEui = "70B3D57ED00313E5";
const char *appKey = "FAFA472FB54FFE967CE63F5EE056F6D7";
```

Также установите частоту, на которой работает устройство и шлюз:

```
// Replace REPLACE_ME with TTN_FP_EU868 or TTN_FP_US915
#define freqPlan TTN_FP_EU868
```

После загрузки скетча в приложение должны приходить данные. Зайдите в консоль TTN, выберите свое приложение, и на вкладке **Data** вы видите поступление данных (рис. 5.108). Это необработанные данные RAW. Но в *разд. 6.7* мы научимся выделять нужные данные и отправлять их по необходимому нам адресу.

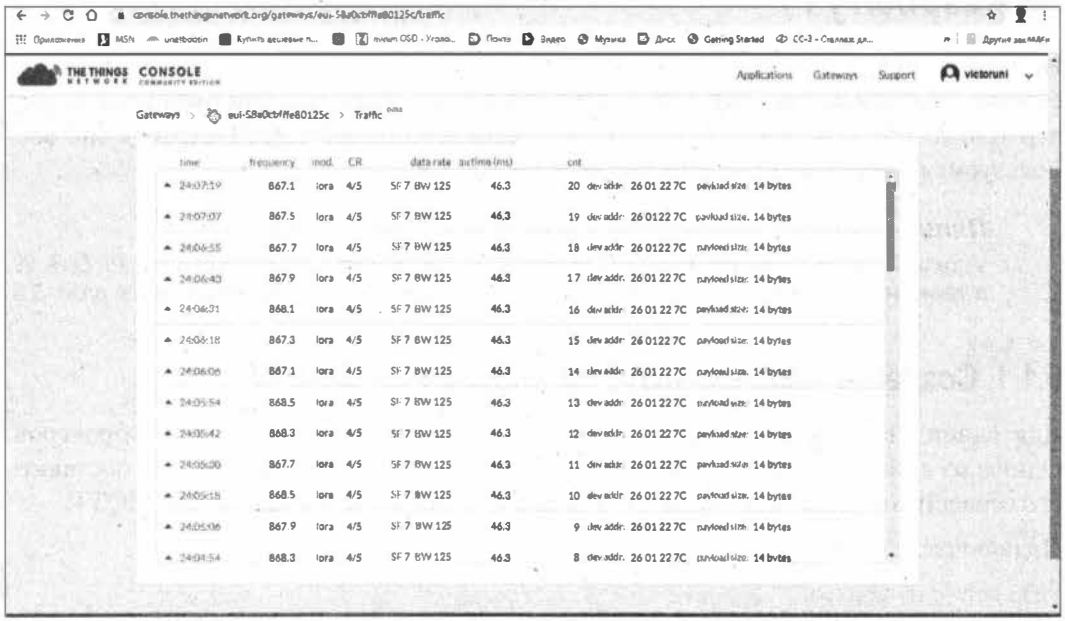
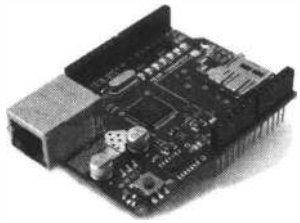


Рис. 5.108. Поступление данных от устройства в сервис The Things Network

ГЛАВА 6



Проекты Интернета вещей

В этой главе представлены примеры проектов IoT, основанных на различных технологиях и протоколах IoT.

6.1. Подключение устройств IoT к серверу MQTT на Raspberry Pi

Не всегда облако для устройств IoT необходимо иметь в сети Интернет. Если вы строите, например, систему локального умного дома, достаточно иметь такой сервер только для локальной сети. Для создания локального MQTT-сервера мы воспользуемся микрокомпьютером Raspberry Pi Zero W.

> ***Примечание***
>
> Установку операционной системы Raspbian на микрокомпьютер Raspberry Pi Zero W, а также настройку его для подключения по Wi-Fi к роутеру мы рассматривали в *разд. 3.5*.

6.1.1. Создание сервера MQTT на Raspberry Pi Zero W

Для одноплатных компьютеров на Linux существует несколько MQTT-брокеров. Одним из самых популярных из них является Mosquitto. Разберемся, как поставить его серверную и клиентскую части, а также проверим работу протокола MQTT.

Подключаемся к Raspberry Pi Zero W по ssh и обновляем систему:

```
sudo apt-get update
sudo apt-get upgrade
```

Обновив систему, приступаем к установке брокера и клиента Mosquitto. Операционная система Raspberry Pi не содержит последнюю версию Mosquitto software, поэтому перед установкой, во избежание ошибок в процессе дальнейшей работы, необходимо обновить библиотеки.

Добавляем ключ и обновляем репозиторий. Для этого выполните следующие команды:

```
sudo wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
sudo apt-get update
```

Устанавливаем MQTT-брокер (сервер):

```
sudo apt-get install mosquitto
```

Устанавливаем MQTT-клиент:

```
sudo apt-get install mosquitto mosquitto-clients
```

После установки Mosquitto сервер будет автоматически запускаться при загрузке системы.

Проверить статус MQTT-брокера можно через терминал:

```
sudo /etc/init.d/mosquitto status
```

После выполнения этой команды в терминале будет показана подробная информация о состоянии вашего MQTT-сервера (рис. 6.1).

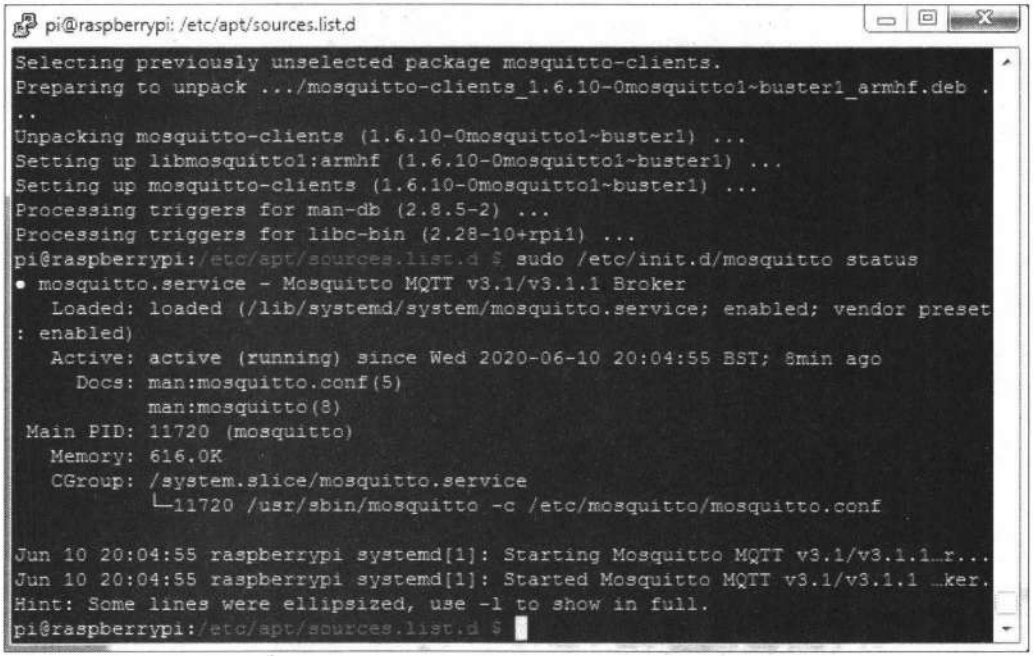


Рис. 6.1. Подробная информация о состоянии MQTT-сервера

Остановка MQTT-брокера:

```
sudo /etc/init.d/mosquitto stop
```

Запуск:

```
sudo /etc/init.d/mosquitto start
```

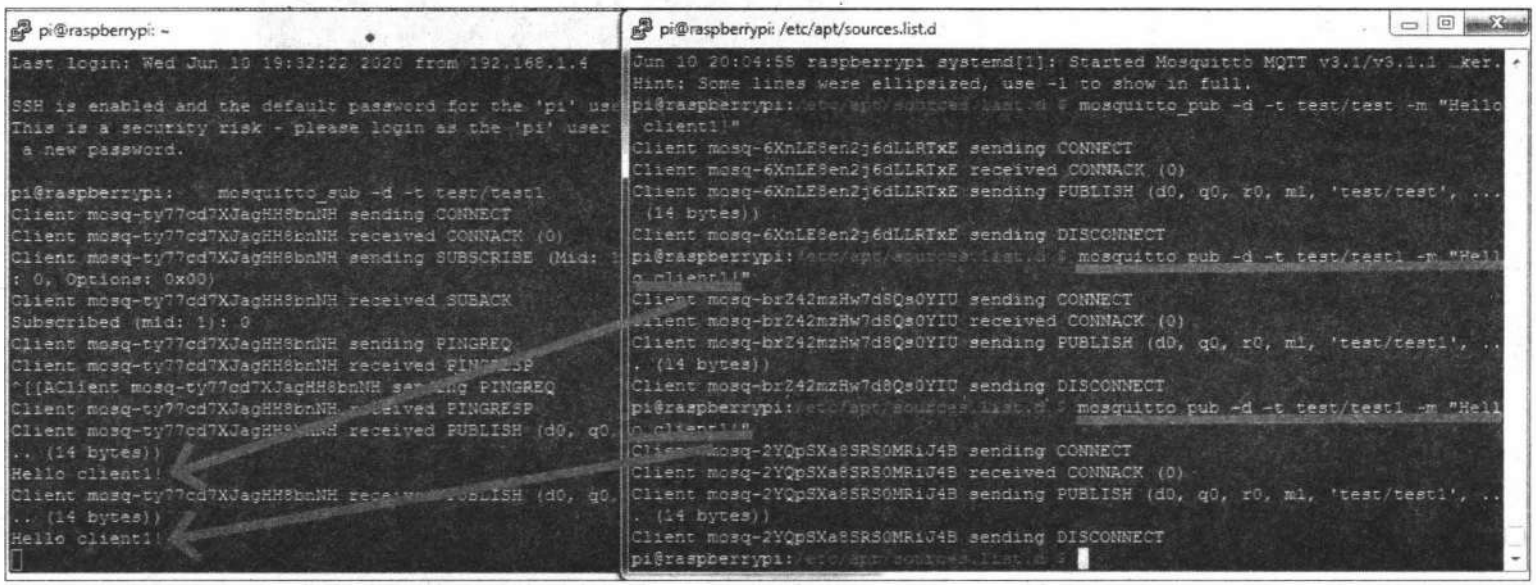


Рис. 6.2. Публикация сообщений в топик и получение сообщений из топика

Теперь проверим работу MQTT-брокера. Для этого нам понадобится MQTT-клиент — например, смартфон с установленным MQTT-клиентом. Но проверить работу MQTT-брокера можно еще проще — поскольку на Raspberry Pi Zero W мы установили и MQTT-брокер, и MQTT-клиент, то мы можем запустить MQTT-клиент из второго терминала.

Так что запускаем на Raspberry Pi Zero W по ssh второй терминал. Подпишемся в нем на топик `test/test1`, выполнив команду:

```
mosquitto_sub -d -t test/test1
```

В первом терминале публикуем сообщение в топик `test/test1`:

```
mosquitto_pub -d -t test/test1 -m "Hello client1!"
```

Это сообщение должно появиться во втором терминале (рис. 6.2).

Для тестов и маленьких проектов можно использовать подключение к MQTT-брокеру без пароля, но в больших системах для повышения уровня безопасности необходимо добавить логин и пароль.

Чтобы клиенты могли подключиться к вашему MQTT-брокеру только по логину и паролю, необходимо создать конфигурационный файл, который будет содержать имя пользователя и зашифрованный пароль:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd <username>
```

Вместо `<username>` необходимо указать здесь имя пользователя, которому будет разрешен доступ, — например, `user1`. После выполнения этой команды пользователь `user1` будет добавлен в систему. Затем нужно ввести два раза пароль для доступа пользователя `user1` к MQTT-брокеру.

Теперь необходимо поменять настройки доступа к системе, для чего надо отредактировать файл **/etc/mosquitto/conf.d/default.conf**:

```
sudo nano /etc/mosquitto/conf.d/default.conf
```

В открывшийся файл добавляем две строки:

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Сохраняем и закрываем файл. осталось перезапустить сервер, и безопасность вашей системы будет теперь на высшем уровне.

6.1.2. Создание устройства IoT для отправки данных на MQTT-сервер на Raspberry Pi

В качестве устройства, публикующего данные в топики MQTT-сервера, мы воспользуемся платой Arduino Nano 33 IoT с подключенным к ней датчиком BME280. Это датчик атмосферного давления, влажности и температуры, работающий по протоколу I^2C. Напряжение питания для датчика 3,3 В. Монтажная схема подключения показана на рис. 6.3.

Подключаемся по Wi-Fi к сети, где находится наш MQTT-сервер. Код подключения к точке доступа Wi-Fi и получения данных с датчика показан в листинге 6.1.

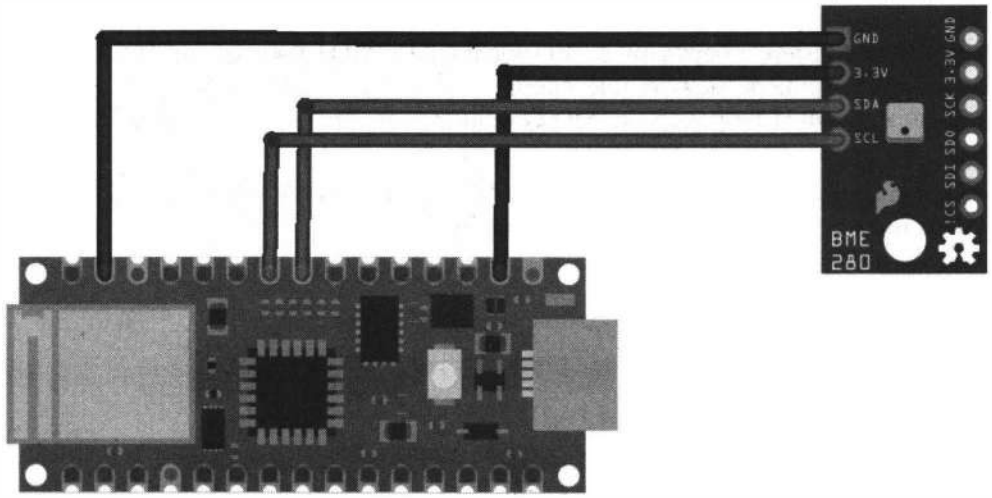


Рис. 6.3. Монтажная схема подключения датчика BME280 к плате Arduino Nano 33 IoT

Листинг 6.1

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

Adafruit_BME280 bme;              // I2C

// данные точки доступа Wi-Fi
char ssid[] = "YourSSID";         // network SSID
char pass[] = "********";         // network password
int status = WL_IDLE_STATUS;      // the Wifi  status

unsigned long millissend=0;
unsigned long intervalsend=5000;

void setup() {
  // Запуск последовательного порта:
  Serial.begin(9600);
  while (!Serial) {
    ;
  }

  // попытка подключения:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
```

```
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);

    // ждем 5 сек:
    delay(5000);
  }

  // вывод данных подключения:
  Serial.print("You're connected to the network");
  printCurrentNet();
  printWifiData();

  bool status;
  // запуск BME280
  status = bme.begin();
  if (!status) {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      while (1);
  }

}

void loop() {
   // получение данных с датчика
   // с выводом в последовательный порт
   if(millis()-millissend>=intervalsend) {
      Serial.print("Temperature = ");
      Serial.print(bme.readTemperature());
      Serial.println(" *C");
      Serial.print("Humidity = ");
      Serial.print(bme.readHumidity());
      Serial.println(" %");
      millissend=millis();
   }
}
```

Электронный архив

Полный скетч, соответствующий листингу 6.1, можно найти в папке *projects\MQTTserver\01* сопровождающего книгу электронного архива (см. *приложение*).

Для отправки данных на сервер Mosquitto по MQTT-протоколу мы воспользуемся Arduino-библиотекой PubsubClient.

Введем авторизацию для подключения к MQTT-серверу по логину и паролю. Создайте конфигурационный файл, который будет содержать имя пользователя и зашифрованный пароль:

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd user2
```

После выполнения этой команды пользователь user2 будет добавлен в систему. Затем необходимо ввести два раза пароль для доступа пользователя user2 к MQTT-брокеру.

Далее необходимо отредактировать файл /etc/mosquitto/conf.d/default.conf:

```
sudo nano /etc/mosquitto/conf.d/default.conf
```

В открывшийся файл добавляем две строки:

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

Сохраняем и закрываем файл. Теперь необходимо перезапустить сервер:

```
sudo /etc/init.d/mosquitto restart
```

Загрузите в плату Arduino Nano 33 IoT код из листинга 6.2, который осуществляет попеременную отправку данных температуры и влажности на MQTT-сервер.

Листинг 6.2

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <PubSubClient.h>

void callback(char* topic, byte* payload, unsigned int length) {
  ;
}

Adafruit_BME280 bme; // I2C
WiFiClient net;
PubSubClient client(net);

const char* mqtt_server = "192.168.0.100";
#define T_TOPIC    "nano33/temp"
#define H_TOPIC     "nano33/humidity"

// данные точки доступа Wi-Fi
char ssid[] = "UserSSID";     // network SSID
char pass[] = "********";     // network password
int status = WL_IDLE_STATUS;     // the Wifi  status

unsigned long millissend=0;
unsigned long intervalsend=5000;

char msg[6];
int queue=0;
```

```
void setup() {
  // Запуск последовательного порта:
  Serial.begin(9600);
  while (!Serial) {
    ; }

  // попытка подключения:
  while (status != WL_CONNECTED) {
    Serial.print("Attempting to connect to WPA SSID: ");
    Serial.println(ssid);
    // Connect to WPA/WPA2 network:
    status = WiFi.begin(ssid, pass);

    // ждем 5 сек:
    delay(5000);
  }

  // вывод данных подключения:
  Serial.print("You're connected to the network");
  printCurrentNet();
  printWifiData();

  // конфигурация подключения к MQTT-серверу
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  bool status;
  // запуск BME280
  status = bme.begin();
  if (!status) {
      Serial.println("Could not find a valid BME280 sensor, check wiring!");
      while (1);
  }

}

void loop() {
   if (!client.connected()) {
      mqttconnect();
   }
   client.loop();
   if(millis()-millissend>=intervalsend) {
     if(queue==0) {
        float t=bme.readTemperature();
        Serial.print("Temperature = ");
        Serial.print(t);
        Serial.println(" *C");
```

```
        snprintf (msg, 5, "%lf", t);
        client.publish(T_TOPIC, msg);
    }
    else {
        float h=bme.readHumidity();
        Serial.print("Humidity = ");
        Serial.print(h);
        Serial.println(" %");
        snprintf (msg, 5, "%lf", h);
        client.publish(H_TOPIC, msg);
    }
    queue=1-queue;
    millissend=millis();
  }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный скетч, соответствующий листингу 6.2, можно найти в папке projects\MQTTserver\02 сопровождающего книгу электронного архива (см. *приложение*).

6.1.3. Создание устройства IoT для получения данных от MQTT-сервера на Raspberry Pi

В качестве устройства, получающего данные с MQTT-сервера, мы воспользуемся платой ESP8266 с подключенной к ней светодиодной матрицей FireBeetle Covers-24x8 LED Matrix (см. *разд. 5.3.8*). Это устройство будет получать данные из топиков `nano33/temp` и `nano33/humidity` и отображать их на экране матрицы.

Загрузите в плату ESP8266 код из листинга 6.3, и вы увидите вывод данных с датчика на экран матрицы (рис. 6.4).

Листинг 6.3

```
#include <ESP8266WiFi.h>
#include "DFRobot_HT1632C.h"
#include <PubSubClient.h>

WiFiClient net;
PubSubClient client(net);

const char* mqtt_server = "192.168.0.100";
#define T_TOPIC    "nano33/temp"
#define H_TOPIC     "nano33/humidity"

const char* ssid     = "UserSSID";
const char* password = "********";
```

```
#define DATA D6
#define CS D2
#define WR D7

DFRobot_HT1632C ht1632c = DFRobot_HT1632C(DATA, WR,CS);

String temp="";
String humidity="";

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.println(topic);
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  char c[length];
  for(int i=0;i<length;i++)
    {c[i]=(char)payload[i];}
  ht1632c.print(c);
  ht1632c.setCursor(18,0);
  String str(topic);
  if(str=="nano33/temp")
     ht1632c.print("C");
  else
     ht1632c.print("%");
}

void setup() {
  Serial.begin(9600);
  delay(10);

  // We start by connecting to a WiFi network
  ht1632c.begin();
  ht1632c.isLedOn(true);
  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  ht1632c.print("WiFi");

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
```

```
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());

  ht1632c.clearScreen();
  ht1632c.setCursor(0,0);
  ht1632c.print("ok");

  /* configure the MQTT server with IPaddress and port */
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

}

void loop() {
   if (!client.connected()) {
      mqttconnect();
   }
   client.loop();

}
```

Электронный архив

Полный скетч, соответствующий листингу 6.3, можно найти в папке *projects\MQTTserver\03* сопровождающего книгу электронного архива (см. *приложение*).

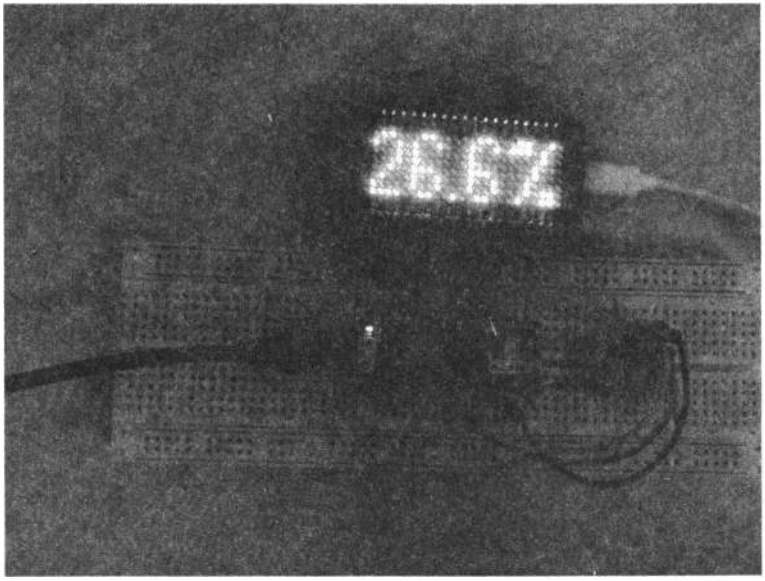


Рис. 6.4. Вывод данных на экран матрицы

6.2. Табло на матрице 32x64 для отображения биржевых котировок в реальном времени

Forex (Форекс) — это международный валютный рынок, товаром на котором служит валюта. Здесь покупают доллары за евро, евро за доллары и т. д. Таким образом, каждая пара валют представляет собой отдельный инструмент. Форекс работает круглосуточно в течение рабочей недели.

Создадим табло для отображения в режиме реального времени некоторых котировок Forex, а также котировок акций и фьючерсных контрактов.

В проекте использованы следующие комплектующие:

- RGB-матрица 64×32 HUB75;
- микрокомпьютер Raspberry Pi Zero W;
- драйвер RGB-матриц для Raspberry Pi;
- блок питания 5 В, 2–5 А.

6.2.1. RGB-матрица HUB75

RGB-матрица 64×32 HUB75 (рис. 6.5) представляет собой панель диагональю 11 дюймов, на которой собраны 2048 RGB-светодиодов. Светодиоды панели светят с яркостью до 1200 кд/м², что позволяет в деталях разглядеть изображение на матрице даже при ярком освещении. С обратной стороны панели распаяна схема управления и предусмотрены крепления под винт (рис. 6.6).

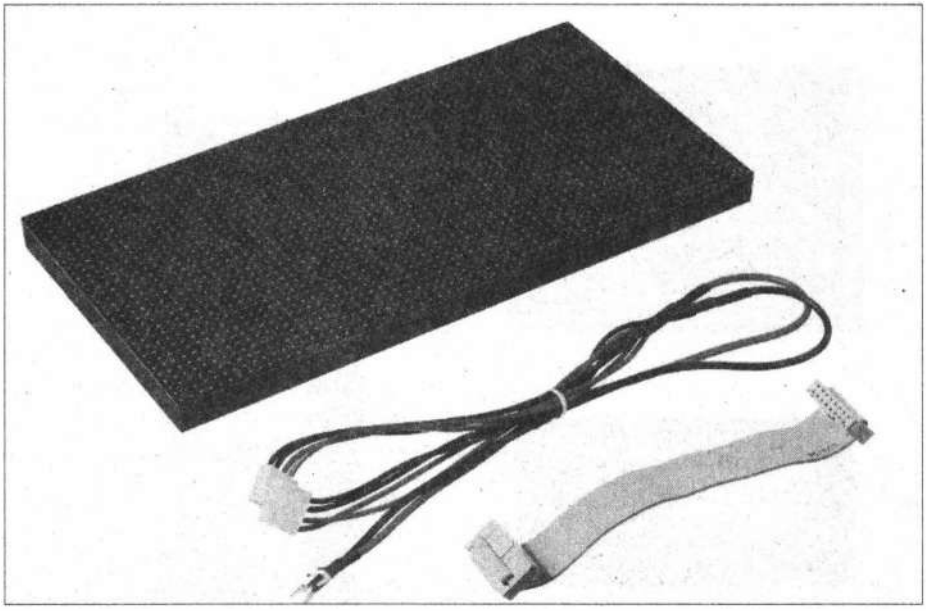


Рис. 6.5. RGB-матрица 64×32

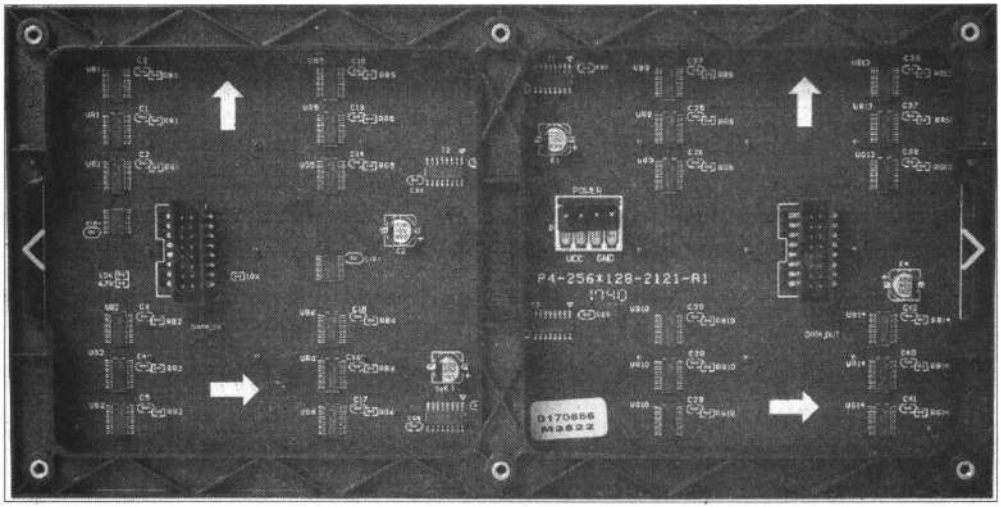


Рис. 6.6. Обратная сторона RGB-матрицы 64×32

На самой матрице нет микроконтроллера и микросхем памяти. Светодиоды подключены через сдвиговые регистры и драйверы. LED-матрица питается от напряжения 5 В и потребляет ток до 4 А. Для подключения матрицы служит 16-проводной шлейф (рис. 6.7).

В качестве микроконтроллера для управления RGB-матрицей мы воспользуемся микрокомпьютером Raspberry Pi — он способен управлять 12 матрицами 32×64. Напомним, что установку операционной системы Raspbian на микрокомпьютер Raspberry Pi Zero W, а также настройку его для подключения по Wi-Fi к роутеру мы рассматривали в *разд. 3.5*.

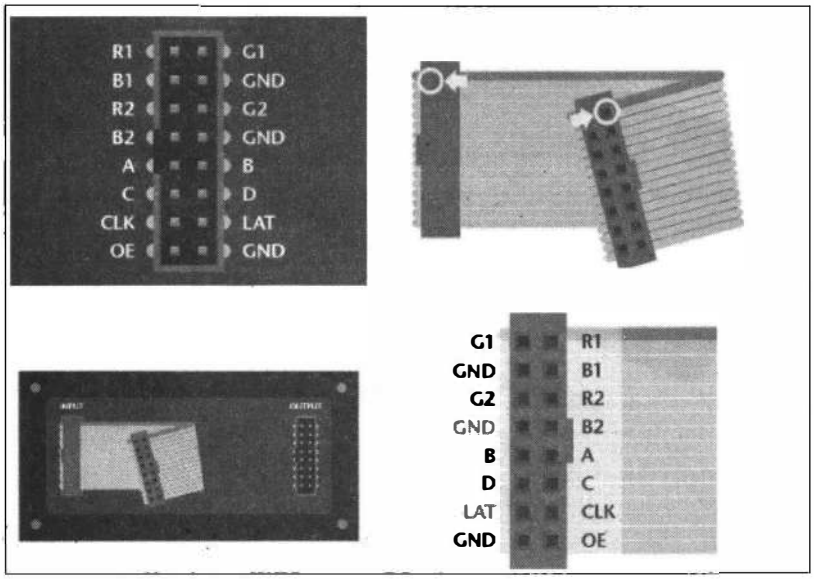


Рис. 6.7. Подключение RGB-матрицы 64×32

6.2.2. Драйвер RGB-матриц для Raspberry Pi

Для подключения матрицы RGB-матрицы 64×32 к Raspberry Pi используется драйвер RGB-матриц. Устанавливается он на Raspberry Pi сверху методом «бутерброда» (рис. 6.8).

Соедините сигнальные линии драйвера и RGB-матрицы через шлейф: один контакт шлейфа подключите в выходной разъем на плате драйвера, а другой — в разъем входящих данных на матрице DATA IN. Подключите питание от драйвера на све-

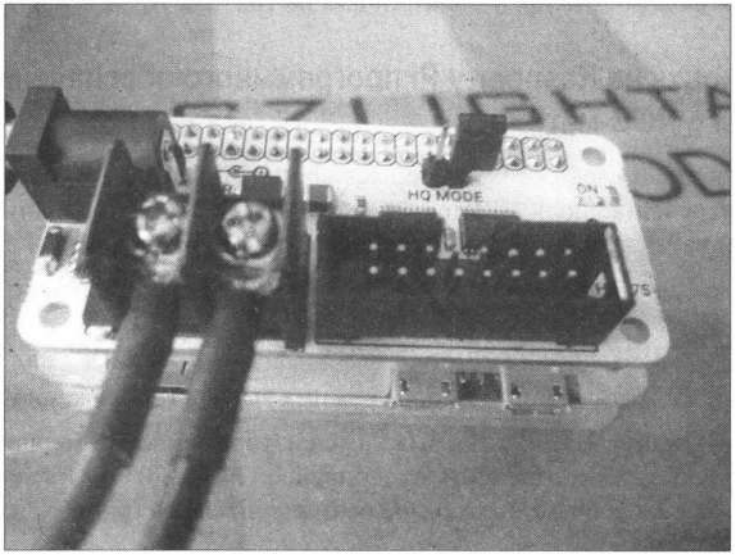


Рис. 6.8. Подключение драйвера RGB-матриц к Raspberry Pi Zero W

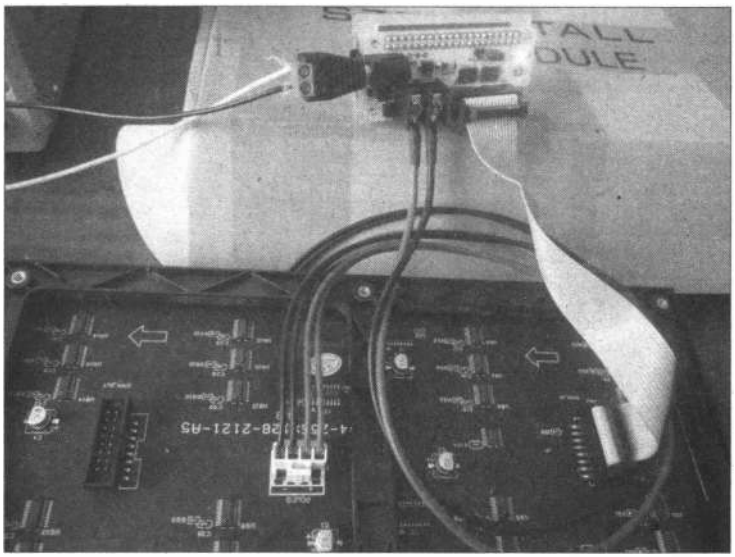


Рис. 6.9. Подключение драйвера RGB-матриц к матрице 64×32

тодиодную матрицу: один конец силового провода подсоедините в клеммник PWR OUT на плате драйвера, а второй — в разъем POWER на матрице. Подключите также питание на плату драйвера от блока питания 5 В через внешний разъем DC Barrel jack (рис. 6.9).

Такая схема потребляет ток до 5 А, но это при включении всех светодиодов на максимум (белый цвет). При подключении нескольких светодиодных панелей необходимо предусмотреть соответственное увеличение запаса по току. Впрочем, в нашем проекте будет выводиться текстовая информация на черном фоне, поэтому можно ограничиться блоком питания, рассчитанным на ток до 2 А.

6.2.3. Установка на Raspberry Pi программного обеспечения для работы с RGB-матрицей

Чтобы установить на микрокомпьютер Raspberry Pi Zero W дополнительное программное обеспечение, подключитесь к нему по ssh. Скачайте и запустите скрипт, позволяющий настроить Raspberry Pi для работы с LED-панелями:

```
wget https://raw.githubusercontent.com/amperka/scriptsRPI/master/
amperka-rpi-rgb-matrix.sh
sudo bash amperka-rpi-rgb-matrix.sh
```

Программа предложит варианты выбора режима вывода изображений (рис. 6.10):

- режим 1 — **Quality**. На матрице выводится изображение без мерцаний и бликов. Но тогда необходимо отключить аудиодрайвер на Raspberry Pi Zero W, а также установить в положение ON джампер на плате драйвера (рис. 6.11);

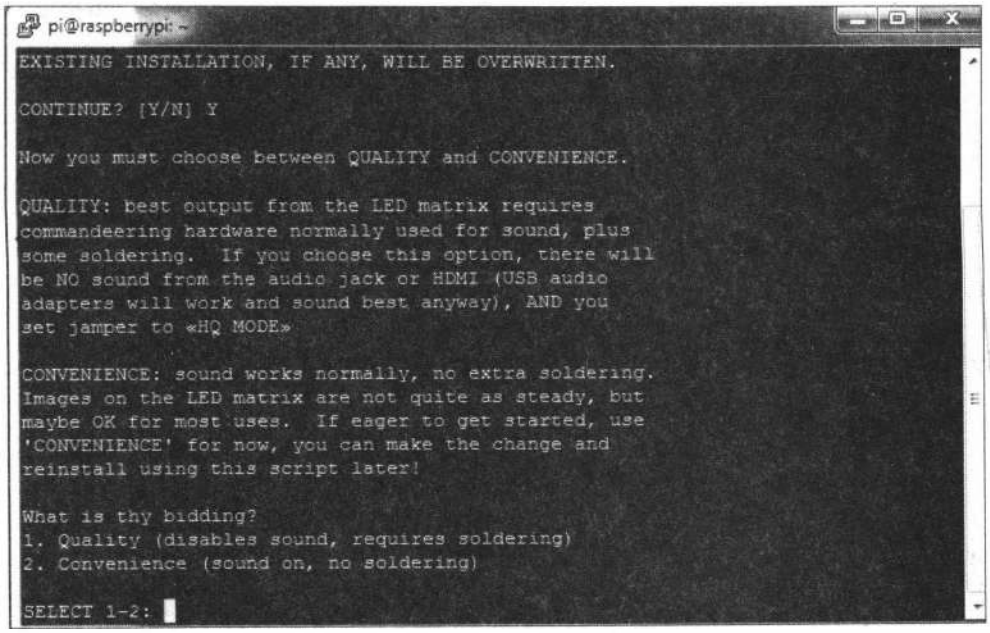


Рис. 6.10. Выбор режима вывода изображений

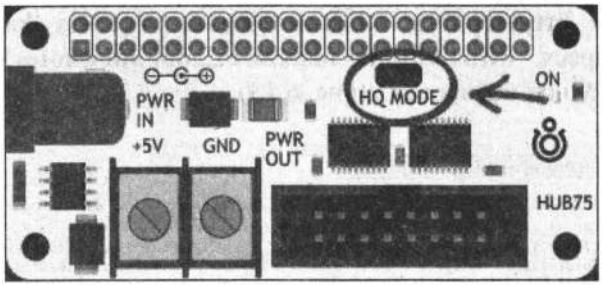


Рис. 6.11. Установка джампера на плате драйвера для режима 1

- режим 2 — **Convenience**. При выводе изображений возможно появление мерцаний. Но звук останется включенным.

Выбираем режим 2 (джампер не включаем) — начнется процесс установки и компиляции, а после перезагрузки в корневой папке должна появиться папка rpi-rgb-led-matrix (рис. 6.12).

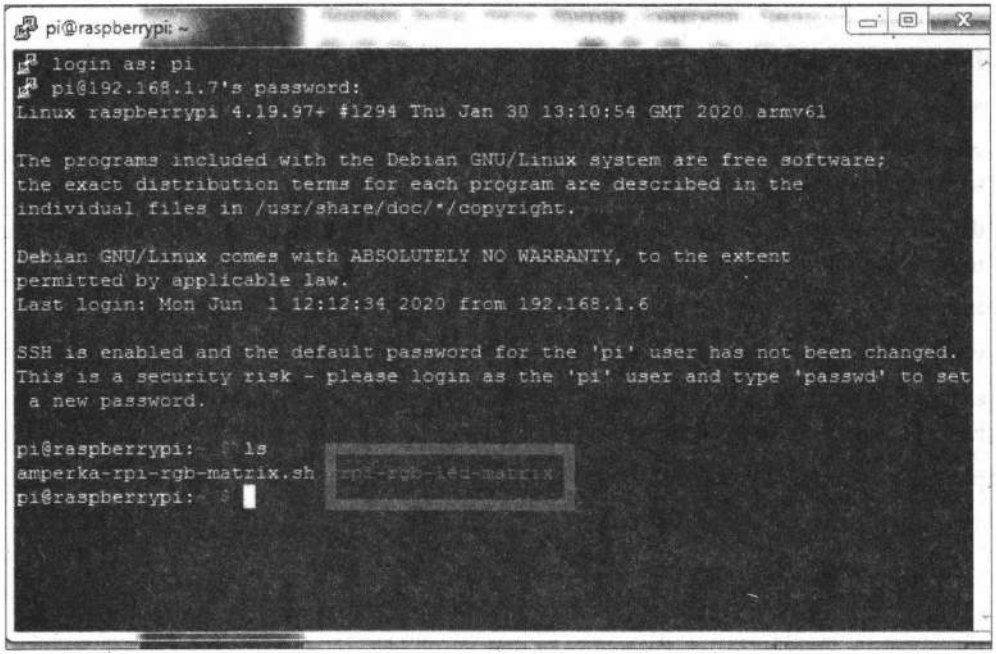


Рис. 6.12. После установки должна появиться папка rpi-rgb-led-matrix

Примеры работы с матрицей на языке Python находятся в папке ./home/pi/rpi-rgb-led-matrix/bindings/python/samples.

6.2.4. Написание скрипта на языке Python

Для получения котировок мы воспользуемся API сайта одного из брокеров Forex: **https://www.instaforex.com**. Информацию по получению котировок можно посмот-

реть на странице: **https://partners.instaforex.com/ru/quotes_description.php**. При обращении по адресу: **https://quotes.instaforex.com/api/quotesTick** мы получаем информацию о текущих котировках (рис. 6.13).

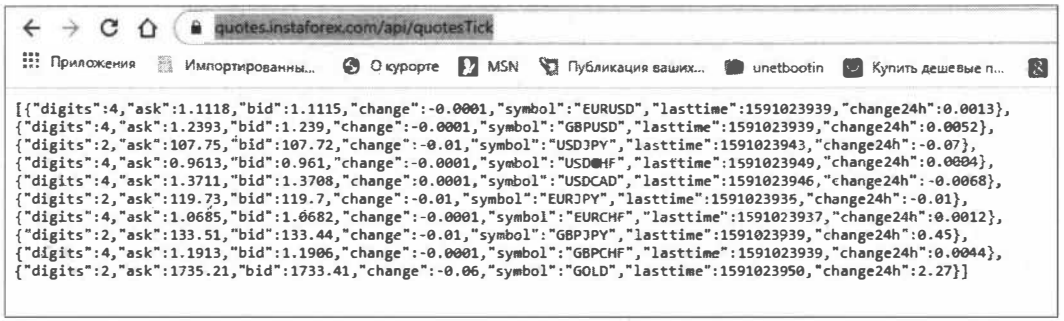


Рис. 6.13. Выдача информации о текущих котировках (API **instaforex.com**)

Здесь:

- **symbol** — символ валютной пары;
- **bid** — значение bid;
- **ask** — значение ask;
- **change** — величина, показывающая изменение относительно предыдущей цены валютной пары;
- **digits** — величина, показывающая, до скольки знаков после запятой округлена цена валютной пары;
- **lasttime** — значение времени последнего изменения цены валютной пары;
- **change24h** — изменение относительно закрытия предыдущего дня.

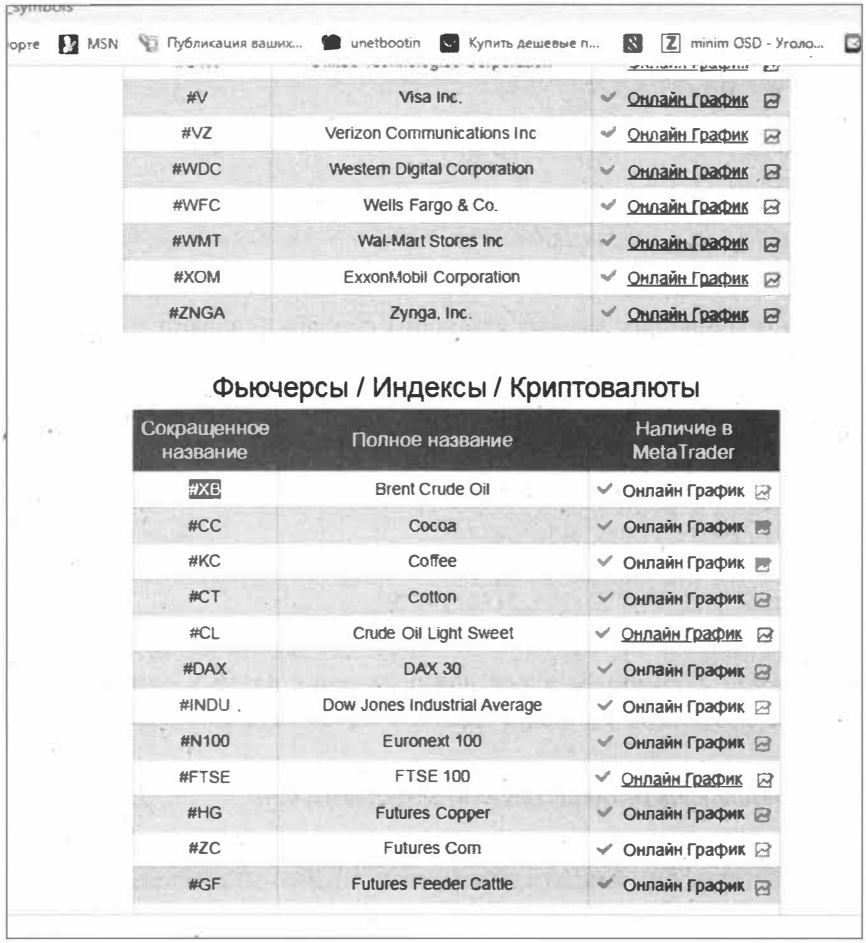
Для получения только необходимых валютных пар следует передать параметр q= со списком валютных пар через запятую. Пример запроса:

https://quotes.instaforex.com/api/quotesTick?q=eurusd,gold,%23bitcoin

Полученный ответ показан на рис. 6.14.

Рис. 6.14. Выдача информации о текущих котировках (API **instaforex.com**) для запрошенных пар

Список пар можно посмотреть на странице **https://www.instaforex.com/ru/trading_symbols**, где есть валютные пары, драгметаллы, акции, криптовалюты, фьючерсы на нефть и прочее (рис. 6.15).



#V	Visa Inc.	Онлайн График
#VZ	Verizon Communications Inc	Онлайн График
#WDC	Western Digital Corporation	Онлайн График
#WFC	Wells Fargo & Co.	Онлайн График
#WMT	Wal-Mart Stores Inc	Онлайн График
#XOM	ExxonMobil Corporation	Онлайн График
#ZNGA	Zynga, Inc.	Онлайн График

Фьючерсы / Индексы / Криптовалюты

Сокращенное название	Полное название	Наличие в MetaTrader
#XB	Brent Crude Oil	Онлайн График
#CC	Cocoa	Онлайн График
#KC	Coffee	Онлайн График
#CT	Cotton	Онлайн График
#CL	Crude Oil Light Sweet	Онлайн График
#DAX	DAX 30	Онлайн График
#INDU	Dow Jones Industrial Average	Онлайн График
#N100	Euronext 100	Онлайн График
#FTSE	FTSE 100	Онлайн График
#HG	Futures Copper	Онлайн График
#ZC	Futures Corn	Онлайн График
#GF	Futures Feeder Cattle	Онлайн График

Рис. 6.15. Список валютных и товарных пар для запроса текущих котировок на API **instaforex.com**

Выбираем необходимые пары и приступаем к написанию программы. Создайте в папке **/home/pi/rpi-rgb-led-matrix/bindings/python/samples** файл **forex.py**.

Подключаем необходимые библиотеки:

```
import json
import requests
import time
from rgbmatrix import RGBMatrix, RGBMatrixOptions
from rgbmatrix import graphics
```

Для RGB-матрицы 64×32 необходимо установить следующие настройки:

```
# Configuration for the matrix
options = RGBMatrixOptions()

options.rows = 32
options.cols = 64
```

```
options.chain_length = 1
options.parallel = 1
options.row_address_type = 0
options.multiplexing = 0
options.pwm_bits = 11
options.brightness = 100
options.pwm_lsb_nanoseconds = 130
options.led_rgb_sequence = "RGB"
options.pixel_mapper_config = ""
```

Для хранения получаемых данных создадим словарь (словари в Python — это неупорядоченные коллекции произвольных объектов с доступом по ключу):

```
arr={'symbolurl':{0:'Forex',1:'EURUSD',2:'%23AAPL',3:'GOLD',4:'%23FB'},
   'symbol':{0:'Forex',1:' EUR/USD',2:'Apple inc',3:'
  GOLD',4:'Facebook'},
   'ask':{0:' ',1:'0',2:'0',3:'0',4:'0'},
   'bid':{0:' ',1:'0',2:'0',3:'0',4:'0'},
   'change':{0:0, 1:0, 2:0 ,3:0 ,4:0},
   'lasttime':{0:' ',1:'0',2:'0',3:'0',4:'0'}
   }
```

Каждые 10 секунд формируем адрес для получения данных одной из пар и получаемые данные сохраняем в словаре. Данные из словаря выводим на экран и в терминал.

Содержимое файла forex.py представлено в листинге 6.4.

Листинг 6.4

```
#!/usr/bin/env python

import json
import requests
import time
from rgbmatrix import RGBMatrix, RGBMatrixOptions
from rgbmatrix import graphics

# Configuration for the matrix
options = RGBMatrixOptions()

options.rows = 32
options.cols = 64
options.chain_length = 1
options.parallel = 1
options.row_address_type = 0
options.multiplexing = 0
options.pwm_bits = 11
options.brightness = 100
```

```
options.pwm_lsb_nanoseconds = 130
options.led_rgb_sequence = "RGB"
options.pixel_mapper_config = ""

matrix = RGBMatrix(options = options)

#
link="https://quotes.instaforex.com/api/quotesTick?q=";
arr={'symbolurl':{0:'Forex',1:'EURUSD',2:'%23AAPL',3:'GOLD',4:'%23FB'},
     'symbol':{0:'Forex',1:' EUR/USD',2:'Apple inc',3:'  GOLD',4:'Facebook'},
     'ask':{0:' ',1:'0',2:'0',3:'0',4:'0'},
     'bid':{0:' ',1:'0',2:'0',3:'0',4:'0'},
     'change':{0:0, 1:0, 2:0 ,3:0 ,4:0},
     'lasttime':{0:' ',1:'0',2:'0',3:'0',4:'0'}
      }

pos=0
offset=0;

matrix.Clear()
canvas=matrix
font1 = graphics.Font()
font1.LoadFont("../../../fonts/7x13.bdf")
font2 = graphics.Font()
font2.LoadFont("../../../fonts/5x7.bdf")
font3 = graphics.Font()
font3.LoadFont("../../../fonts/4x6.bdf")
textColor1 = graphics.Color(0, 255, 0)
textColor2 = graphics.Color(0, 0, 255)
textColor3 = graphics.Color(0, 0, 255)
textColor4 = graphics.Color(0, 255, 0)
textColor5 = graphics.Color(255, 0, 0)

s = graphics.DrawText(canvas, font2, 0, 15, textColor1, "instaforex ")
time.sleep(2)   #

ipos=arr.keys()
ipos=sorted(ipos)
for i in range(len(ipos)):
   print(ipos[i],"=",arr[ipos[i]][0])
   print(ipos[i],"=",arr[ipos[i]][1])
   print(ipos[i],"=",arr[ipos[i]][2])
   print(ipos[i],"=",arr[ipos[i]][3])
   print(ipos[i],"=",arr[ipos[i]][4])

while True:
   #
   matrix.Clear()
```

```
graphics.DrawText(canvas, font1, 0, 10, textColor2,arr['symbol'][pos])
graphics.DrawText(canvas, font2, 5, 20, textColor2, str(arr['ask'][pos]))
if arr['change'][pos]>0:
   graphics.DrawText(canvas, font2, 10, 30, textColor4,
                                              "+"+str(arr['change'][pos]))
elif arr['change'][pos]<0:
   graphics.DrawText(canvas, font2, 10, 30, textColor5,
                                                 str(arr['change'][pos]))
else :
   graphics.DrawText(canvas, font2, 10, 30, textColor3, "")
#
pos=max(1,(pos+1)%5)
print("pos=",pos)

responce=requests.get("https://quotes.instaforex.com/api/quotesTick?q=
                                                 "+arr['symbolurl'][pos])
data1=json.loads(responce.text)
print("data1=",data1)
i=0
for rec in data1:
   print(pos)
   arr['ask'][pos]=rec['ask']
   arr['bid'][pos]=rec['bid']
   arr['change'][pos]=rec['change']
   #print("pos=",pos,"  ask=",arr['ask'][pos],"  bid=",rec['bid'],
                                          "  change=",rec['change'])
   i=i+1

for rec in data1:
   #print(rec)
   ipos=rec.keys()
   ipos=sorted(ipos)
   for i in range(len(ipos)):
      print(ipos[i],"=",rec[ipos[i]])
   print("----------------")
#
time.sleep(10)
```

Электронный архив

Скрипт *forex.py*, соответствующий листингу 6.4, можно найти в папке *projects\forex* сопровождающего книгу электронного архива (см. *приложение*).

Запустите файл на выполнение:

```
sudo python3 /home/pi/rpi-rgb-led-matrix/bindings/python/samples/forex.py
sudo python3 /home/pi/rpi-rgb-led-matrix/bindings/python/samples/forex.py
```

После запуска файла наблюдаем в терминале обработанные результаты запросов (рис. 6.16), а на матрице — текущие котировки для выбранных пар (рис. 6.17–6.20).

```
pi@raspberrypi: ~/rpi-rgb-led-matrix/bindings/python/samples
data1= [{'digits': 4, 'ask': 1.1113, 'bid': 1.111, 'change': -0.0001, 'symbol':
'EURUSD', 'lasttime': 1591027859, 'change24h': 0.0008}]
1
ask = 1.1113
bid = 1.111
change = -0.0001
change24h = 0.0008
digits = 4
lasttime = 1591027859
symbol = EURUSD
----------------
pos= 2
data1= [{'digits': 2, 'ask': 319.28, 'bid': 319.2, 'change': -0.1, 'symbol': '#A
APL', 'lasttime': 1590793199, 'change24h': -0.3}]
2
ask = 319.28
bid = 319.2
change = -0.1
change24h = -0.3
digits = 2
lasttime = 1590793199
symbol = #AAPL
----------------
```

Рис. 6.16. Обработанные результаты запросов текущих котировок для API **instaforex.com**

Рис. 6.17. Вывод на RGB-матрицу 64×32 текущих котировок акций Facebook

Рис. 6.18. Вывод на RGB-матрицу 64×32 текущих котировок EUR/USD

Рис. 6.19. Вывод на RGB-матрицу 64×32 текущих котировок акций Apple

Рис. 6.20. Вывод на RGB-матрицу 64×32 текущих котировок золота

6.3. MQTT-чат для нескольких устройств IoT

Мы уже несколько раз прибегали к использованию самого популярного для IoT-устройств протокола — MQTT, позволяющего отправлять и получать данные через MQTT-сервер. Но мы можем применить этот протокол также для обмена сообщениями, отправляемыми человеком с устройств IoT, т. е. создать своего рода чат. Нам понадобится лишь оснастить каждое IoT-устройство средствами ввода и вывода информации.

6.3.1. Создание IoT-устройства на микроконтроллере ESP32 и дисплее Nextion

Основой IoT-устройства в этом проекте станет микроконтроллер ESP32. Для ввода данных, отправляемых микроконтроллером ESP32 на MQTT-сервер, и отображения данных, поступающих с MQTT-сервера, мы применим сенсорный дисплей Nextion, подключив его к микроконтроллеру ESP32 по последовательному порту Serial1:

- RX1 on GPIO9;
- TX1 on GPIO10.

Монтажная схема подключения показана на рис. 6.21.

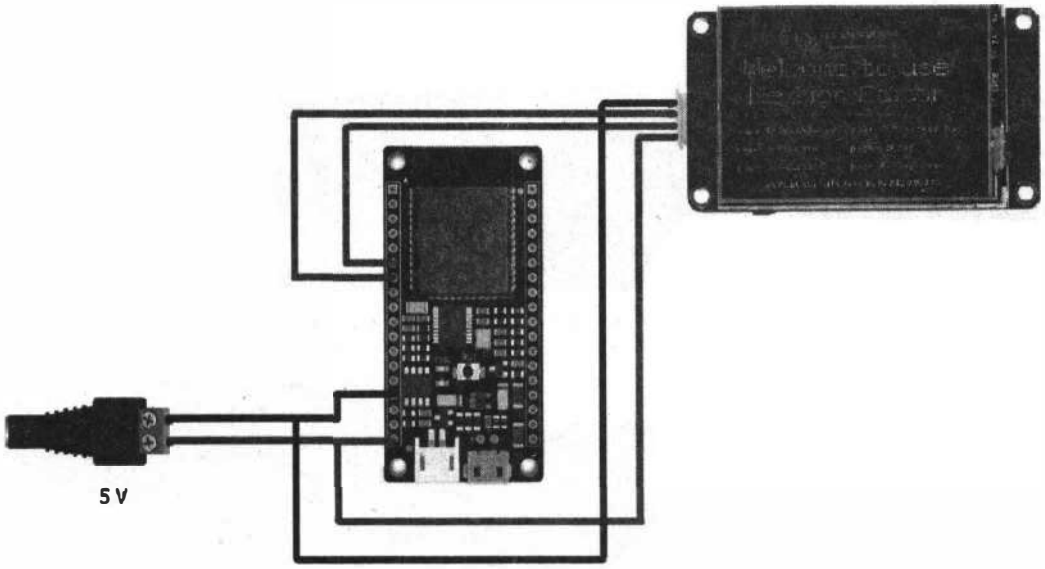


Рис. 6.21. Монтажная схема соединений для подключения дисплея Nextion к плате ESP32

6.3.2. Создание интерфейса пользователя на дисплее Nextion

Дисплей Nextion (рис. 6.22) представляет собой полноценный компьютер с процессором, видеокартой и экраном, причем он выделяет весь свой вычислительный ре-

сурс на обработку графики и позволяет записывать в него свои программы. На модулях дисплеев Nextion имеется разъем UART и выводы GPIO, что дает возможность использовать дисплеи Nextion как совместно с Arduino — подключая дисплей к Arduino по шине UART (обмен с помощью унифицированных команд), так и отдельно — подключая кнопки, светодиоды, реле и прочие элементы схемы напрямую к выводам GPIO дисплея). Через имеющийся у дисплея Nextion разъем для карт памяти micro-SD в него можно загружать программы.

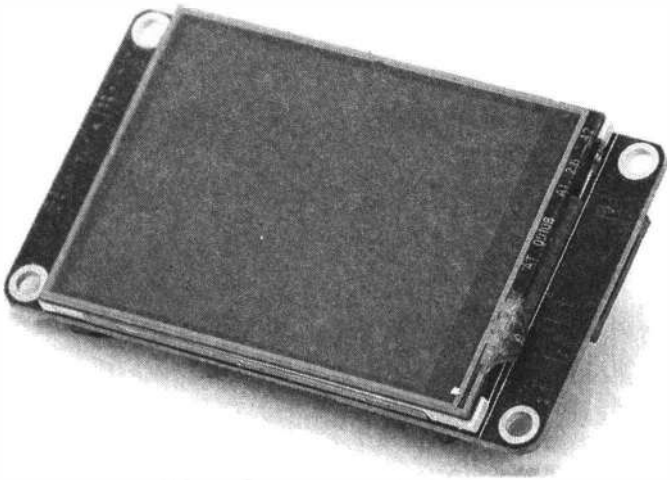


Рис. 6.22. Дисплей Nextion NX3224K024

Для работы с дисплеями Nextion необходимо установить программу Nextion Editor, которая позволяет создавать интерфейс пользователя с использованием различных библиотечных элементов: кнопок, слайдеров, картинок, графики, текста и т. п., а также прописывать алгоритмы поведения дисплея для различных событий элементов, формирующих этот интерфейс.

В программе Nextion Editor нам понадобится создать несколько страниц (рис. 6.23): окно чата для отображения последних сообщений и страницы для создания сообщения (окна клавиатур со строчным алфавитом, заглавными буквами и специальными символами). Если есть желание отправлять сообщения на русском языке, необходимо добавить страницы с русской клавиатурой.

Для всех кнопок необходимо активировать отправку кодов по последовательному порту при событии `Touch Presss Event` (рис. 6.24).

Для прошивки дисплея через UART понадобится адаптер USB-Serial. Монтажная схема подключения дисплея Nextion к адаптеру USB-Serial показана на рис. 6.25.

Для загрузки прошивки выбираем пункт меню **Upload** и в открывшемся окне нажимаем на кнопку **Go**. Процесс прошивки будет отображаться в окне программы (рис. 6.26) и на дисплейном модуле. По окончании прошивки загружаемый проект начнет выполняться и отображаться на дисплейном модуле. Надо отметить, что прошивка через UART занимает весьма долгое время.

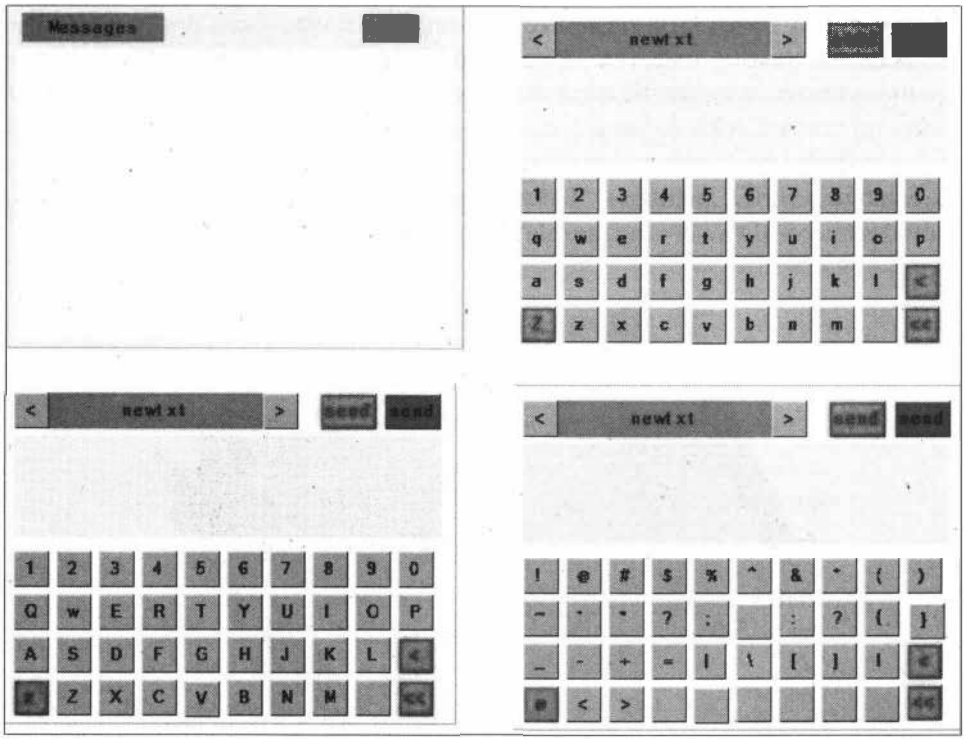


Рис. 6.23. Страницы интерфейса для дисплея Nextion

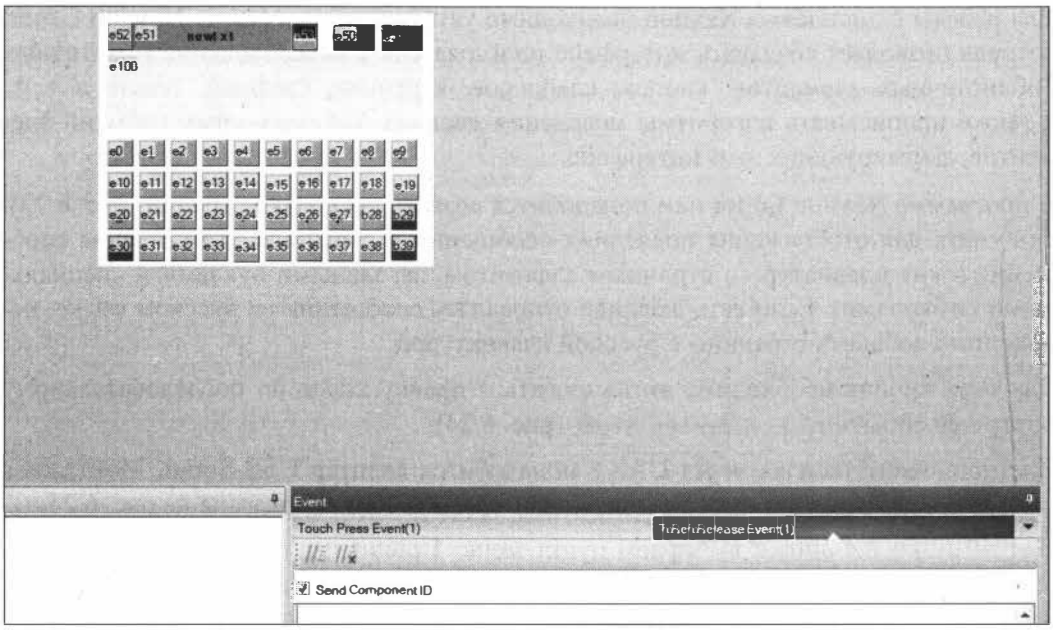


Рис. 6.24. Активация отправки кодов по последовательному порту при событии `Touch Presss Event`

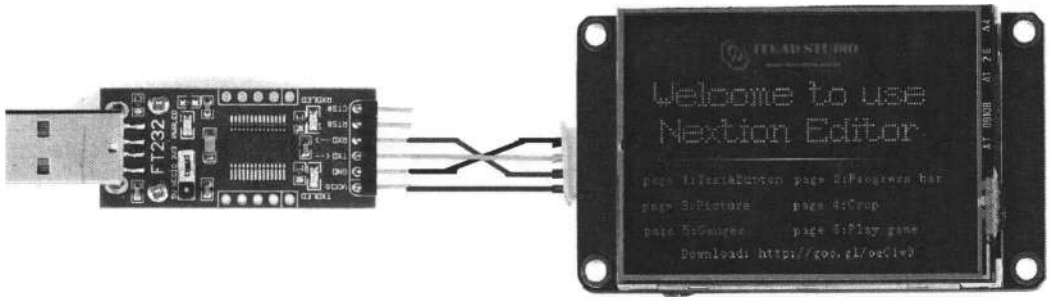


Рис. 6.25. Монтажная схема подключения дисплея Nextion к адаптеру USB-Serial

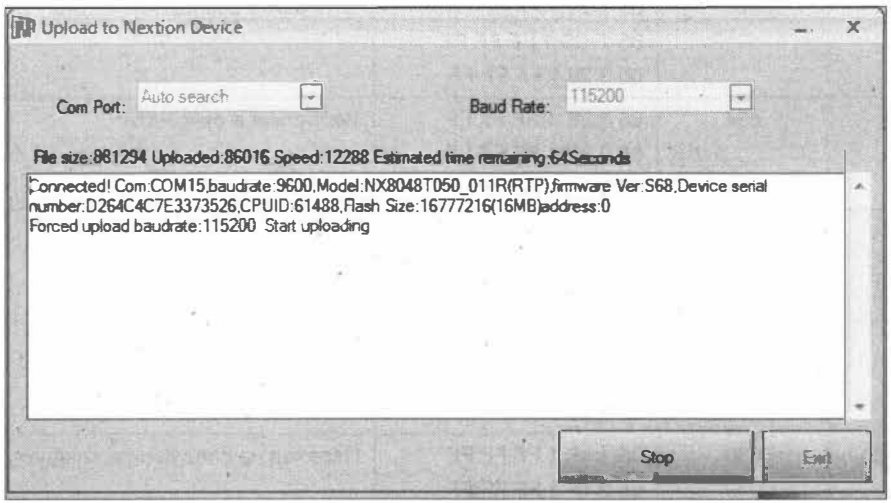


Рис. 6.26. Загрузка прошивки в дисплей Nextion через последовательный порт

ЭЛЕКТРОННЫЙ АРХИВ

Файл проекта *chartMQTT.HMI* для дисплея Nextion можно найти в папке *projects\chartMQTT* сопровождающего книгу электронного архива (см. *приложение*).

6.3.3. Получение данных, поступающих с дисплея Nextion, и отправка данных на дисплей

При нажатии кнопок на дисплее Nextion плата ESP32 получает данные по последовательному порту Serial1. Данные, получаемые из загруженной в дисплей Nextion прошивки, представлены в табл. 6.1.

Скетч в зависимости от полученных данных производит определенные действия, прописанные в процедуре `parse_message()`. Это либо просто печать символа и добавление его в сообщение, либо переход между страницами символов, либо возврат, либо отправка сообщения на сервер и т. д. (см. табл. 6.1).

Содержимое процедуры `parse_message()` представлено в листинге 6.5.

Таблица 6.1. Данные, приходящие из Nextion

Кнопка	Id элемента в Nextion	Данные	Примечание
+ADD	t2	65 0 3 0 FF FF FF	Создать сообщение
<	e52	65 1 2C 1 FF FF FF 65 2 2C 1 FF FF FF 65 3 2C 1 FF FF FF	Выбор получателя сообщения (предыдущий в списке)
>	e53	65 1 2D 1 FF FF FF 65 2 2D 1 FF FF FF 65 3 2D 1 FF FF FF	Выбор получателя сообщения (следующий в списке)
send	e50	65 1 29 1 FF FF FF 65 2 29 1 FF FF FF 65 3 29 1 FF FF FF	Отправить сообщение на MQTT-сервер
back	e54	65 1 2E 1 FF FF FF 65 2 2E 1 FF FF FF 65 3 2E 1 FF FF FF	Вернуться в окно чата
<	b29	65 1 1E 1 FF FF FF 65 2 1E 1 FF FF FF 65 3 1E 1 FF FF FF	Удалить последний символ
<<	b39	65 1 28 1 FF FF FF 65 2 28 1 FF FF FF 65 3 28 1 FF FF FF	Вставка новой строки
z, Z, @	b30	65 1 1F 1 FF FF FF 65 2 1F 1 FF FF FF 65 3 1F 1 FF FF FF	Переход на следующую клавиатуру
остальные			Отправка символа в сообщение

Листинг 6.5

```
void parse_message() {
   if(nextionData[0]!=0x65) {
      return;
   }
   if(nextionData[4]!=0xff && nextionData[5]!=0xff &&
      nextionData[6]!=0xff) {
      return;
   }
   // *****
   // переход на набор сообщения
   if(nextionData[1]==0 && nextionData[2]==3 &&
      nextionData[3]==0) {
    nextionSerial.print("page 1");
```

```
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 // user
 nextionSerial.print("e51.txt=\"");
 nextionSerial.print(users[tuser].userName);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 //·message
 nextionSerial.print("e100.txt=\"");
 nextionSerial.print(tekMessage);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
}
// переход на другую клавиатуру
else if(nextionData[2]==0x1F) {
 nextionSerial.print("page ");
 nextionSerial.print(String((nextionData[1]%3+1)));
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 nextionSerial.print("e100.txt=\"\\r");
 nextionSerial.print(tekMessage);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
}
// send
else if(nextionData[2]==0x29) {
 Serial.println("send");
 publishData();
 nextionSerial.print("page 0");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 //chatMessage=chatMessage+"\\r"+"\\t";
 chatMessage=chatMessage+"I --> "+users[tuser].userLogin+": ";
 chatMessage=chatMessage+tekMessage;
 Serial.println(chatMessage);
```

```
 nextionSerial.print("t0.txt=\"");
 nextionSerial.print(chatMessage);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 chatMessage=chatMessage+"\\r";
 nextionSerial.print("t0.txt+=\"\\r\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);

 tekMessage="";

}
// back
else if(nextionData[2]==0x2E) {
 Serial.println("back");
 nextionSerial.print("page 0");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 Serial.println(chatMessage);
 nextionSerial.print("t0.txt=\"");
 nextionSerial.print(chatMessage);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 delay(10);
 tekMessage="";

}
// предыдущий user
else if(nextionData[2]==0x2C) {
 Serial.println("previos user");
 tuser=(tuser+nusers-1)%nusers;
 nextionSerial.print("e51.txt=\"");
 nextionSerial.print(users[tuser].userName);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
}
```

```
// следующий user
else if(nextionData[2]==0x2D) {
 Serial.println("next user");
 tuser=(tuser+1)%nusers;
 nextionSerial.print("e51.txt=\"");
 nextionSerial.print(users[tuser].userName);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
}
// забой
else if(nextionData[2]==0x1E) {
 if(tekMessage.length()>0){
   int d=1;
   Serial.println(tekMessage.length());
   if(tekMessage.endsWith("\r")==true)
     d=2;
   tekMessage=tekMessage.substring(0,tekMessage.length()-d);
   Serial.println(tekMessage);
   Serial.println(tekMessage.length());
   Serial.println("забой");
   nextionSerial.print("e100.txt=e100.txt-1");
   nextionSerial.write(0xff);
   nextionSerial.write(0xff);
   nextionSerial.write(0xff);
 }
}
// \r\n
else if(nextionData[2]==0x28) {
 Serial.println("\ n");
 tekMessage=tekMessage+"\\r";
 nextionSerial.print("e100.txt+=\"\\r\"");
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
 nextionSerial.write(0xff);
}
// вывод символа
else {
 tekMessage=tekMessage+
       nextionChars[nextionData[1]-1][nextionData[2]-1];
 Serial.println(tekMessage);
 Serial.println(tekMessage.length());
 nextionSerial.print("e100.txt=\"");
 nextionSerial.print(tekMessage);
 nextionSerial.print("\"");
 nextionSerial.write(0xff);
```

```
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
   }
}
```

6.3.4. Отправка и получение данных с MQTT-сервера

По нажатию кнопки **send** происходит отправка данных — публикация в топики: `/user1/user2`, `/user1/user3` или `/user1/user4` — в зависимости от выбранного получателя сообщения. Для отправки данных используется процедура `publishData()`. Содержимое процедуры представлено в листинге 6.6.

Листинг 6.6

```
// отправка данных в темы брокера
void publishData() {
    String topic="/"+iam.userLogin+"/"+users[tuser].userLogin;
    client.publish(topic.c_str(), tekMessage.c_str(), true);
}
// получение данных
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("from topic - ");
 Serial.print(topic);
 Serial.print("  ");
 Serial.print(length);
 Serial.print("=");
 String myString = String((char*)payload);
 Serial.println(myString);
    chatMessage=chatMessage+topic+": ";
    chatMessage=chatMessage+myString.substring(0,9);
    Serial.println(chatMessage);
    nextionSerial.print("t0.txt=\"");
    nextionSerial.print(chatMessage);
    nextionSerial.print("\"");
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    delay(10);
    chatMessage=chatMessage+"\\r";
    nextionSerial.print("t0.txt+=\"\\r\"");
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
}
```

Для получения данных с сервера подписываемся на топики: `/user2/user1`, `/user3/user1`, `/user4/user1`:

```
client.subscribe("/user2/user1");
client.subscribe("/user3/user1");
client.subscribe("/user4/user1");
```

И объявляем функцию обратного вызова `callback`:

```
client.setCallback(callback);
```

Содержимое функции `callback` представлено в листинге 6.7.

Листинг 6.7

```
// получение данных
void callback(char* topic, byte* payload, unsigned int length) {
 Serial.print("from topic - ");
 Serial.print(topic);
 Serial.print("  ");
 Serial.print(length);
 Serial.print("=");
 String myString = String((char*)payload);
 Serial.println(myString);
    chatMessage=chatMessage+topic+": ";
    chatMessage=chatMessage+myString.substring(0,9);
    Serial.println(chatMessage);
    nextionSerial.print("t0.txt=\"");
    nextionSerial.print(chatMessage);
    nextionSerial.print("\"");
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    delay(10);
    chatMessage=chatMessage+"\\r";
    nextionSerial.print("t0.txt+=\"\\r\"");
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
    nextionSerial.write(0xff);
}
```

Загружаем скетч в плату ESP32 и на экране дисплея Nextion набираем сообщения и отправляем их на сервер. Все сообщения чата можно посмотреть на сервере **cloudmqtt.com** (рис. 6.27). При этом из панели сервера можно отправлять сообщения в любые темы — например, для нашего устройства. Входящие и исходящие сообщения пользователя отображаются на экране дисплея (рис. 6.28).

> ***ЭЛЕКТРОННЫЙ АРХИВ***
>
> Полный набор скетчей для этого проекта можно найти в папке *projects\chartMQTT\chart* сопровождающего книгу электронного архива (см. *приложение*).

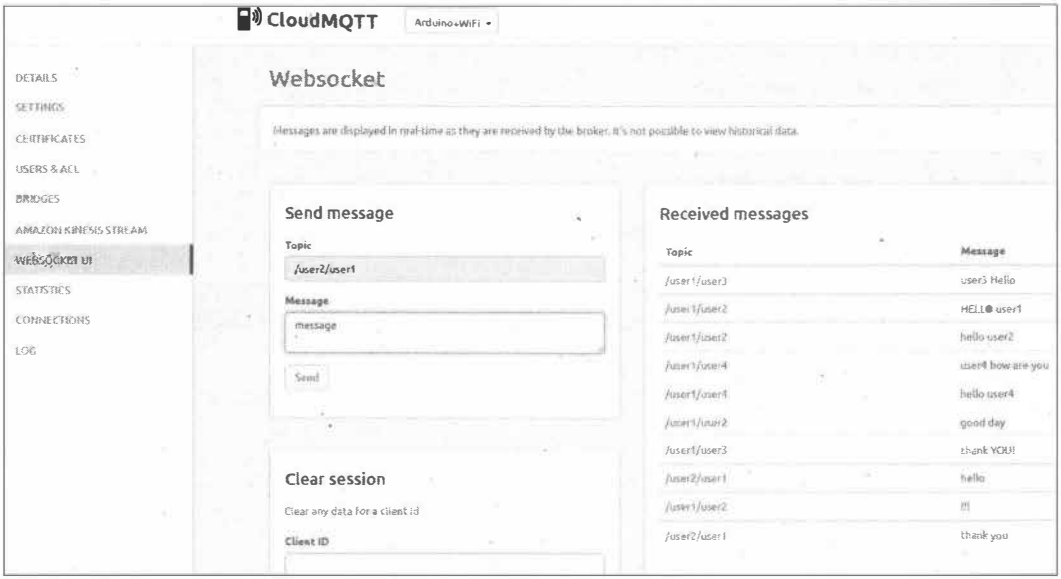


Рис. 6.27. Просмотр сообщений чата на сервере **cloudmqtt**

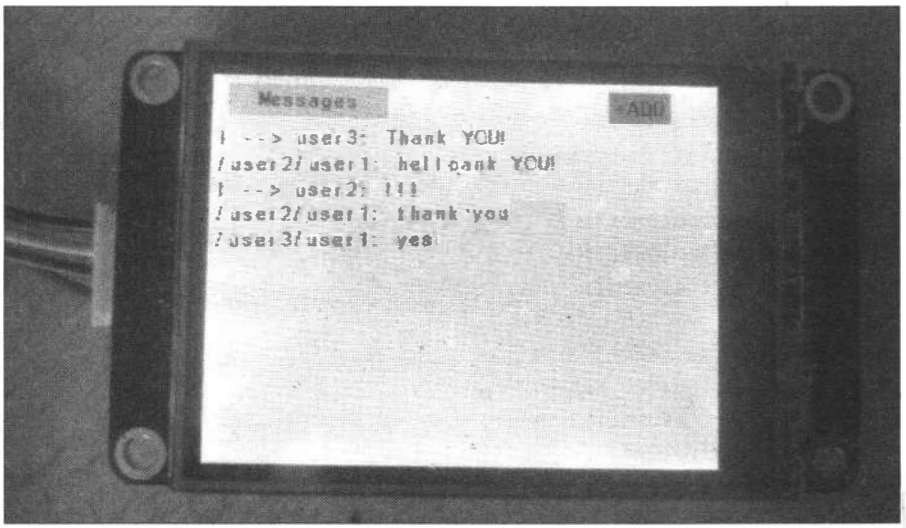


Рис. 6.28. Входящие и исходящие сообщения пользователя на экране дисплея Nextion

6.4. Получение и печать курса валют на термопринтере в проекте IoT

Создадим на основе модуля ESP8266 проект IoT-принтера, который будет печатать курс валют на текущую дату, получая данные через Интернет с сайта **cbr.ru**. В качестве принтера в этом проекте мы воспользуемся бюджетным термопринтером, выпускаемым специально для Arduino (рис. 6.29).

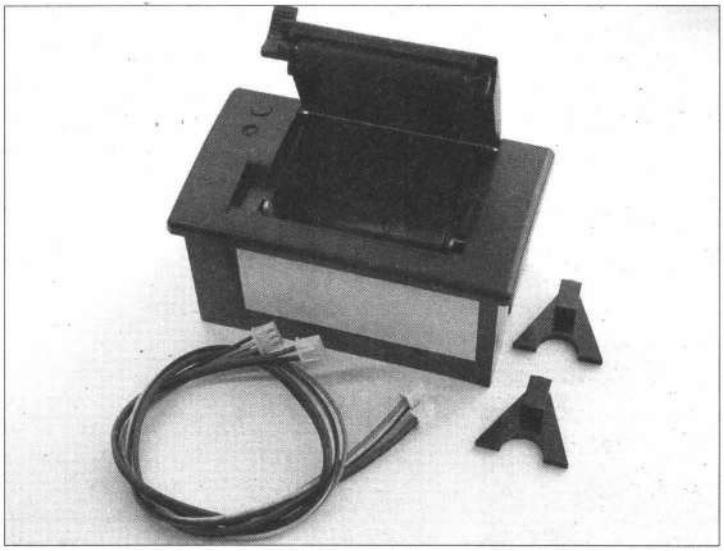


Рис. 6.29. Термопринтер для Arduino

Принтер печатает на термобумаге шириной 2,25 дюйма, которую можно приобрести в магазине канцелярских товаров. Вам также понадобится регулируемый источник питания от 5 до 9 В постоянного тока, который может обеспечить ток более 1,5 А. Общение принтера с платой Arduino мы организуем с помощью UART-соединения.

Прежде всего необходимо провести начальный тест принтера. Подключите принтер к блоку питания, держа нажатой кнопку на его верхней панели, — будут распечатаны таблица шрифтов и некоторая дополнительная информация (рис. 6.30). Нужный нам параметр — скорость обмена по последовательному порту (Baudrate): 19 200 бод.

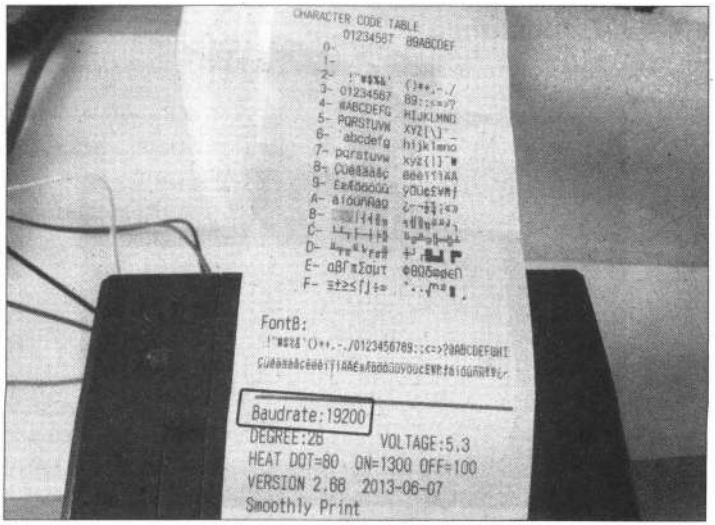


Рис. 6.30. Распечатка тестовой страницы

Подключим теперь термопринтер к модулю ESP8266 (в проекте используется отладочная плата NodeMCU — удобная платформа на основе модуля ESP8266) по схеме, представленной на рис. 6.31, и загрузим в плату NodeMCU код из листинга 6.8 (для программирования нам потребуется Arduino-библиотека для принтера Adafruit Thermal). В результате мы увидим вывод на принтер тестовых данных (рис. 6.32).

Электронный архив

Библиотека Adafruit Thermal размещена в папке *libraries* сопровождающего книгу электронного архива (см. *приложение*).

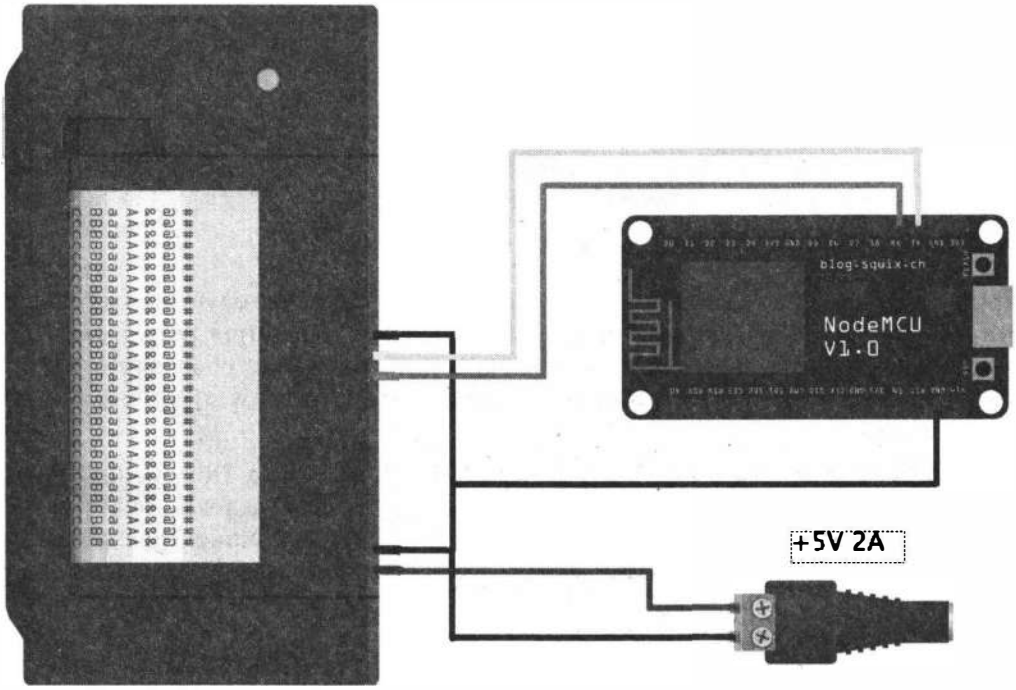


Рис. 6.31. Монтажная схема подключения термопринтера к отладочной плате NodeMCU ESP8266

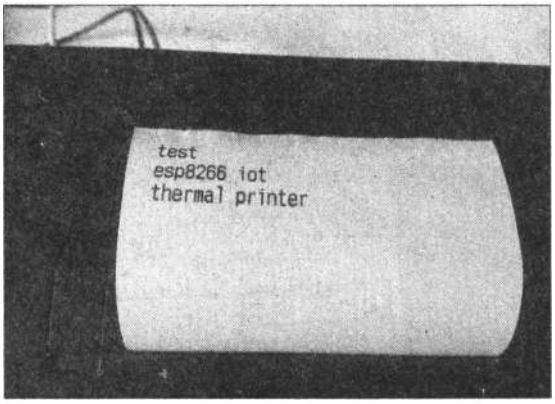


Рис. 6.32. Вывод тестовых данных из скетча на принтер

Листинг 6.8

```
#include "Adafruit_Thermal.h"
Adafruit_Thermal printer(&Serial);     // Pass addr to printer constructor

void setup() {
  // запуск последовательного порта
  Serial.begin(19200);
  // инициализация принтера
  printer.begin();
  delay(3000);
  // настройки по умолчанию
  printer.wake();
  printer.setDefault();
  printer.println();
  printer.println();
  delay(1000);
  printer.println("test ");
  printer.println("esp8266 iot");
  printer.println("thermal printer");
}
void loop() {
}
```

Нашему проекту для правильной работы необходимо знать реальное время, для получения которого мы воспользуемся модулем часов реального времени (RTC) на микросхеме DS3231. Подключение модуля DS3231 к модулю NodeMCU ESP8266 осуществляется по протоколу I^2C: соединяем контакты NodeMCU D3 (GPIO0) и D4 (GPIO2) соответственно с контактами SCL и SDA модуля DS3231 (рис. 6.33).

Теперь нам необходимо получать из Интернета актуальный курс валют. Курсы валют в формате XML доступны на сайте Сбербанка по адресу: **www.cbr.ru/scripts/XML_daily.asp?date_req=<data>**, где **<data>** — дата в формате **dd/mm/yyyy**. Если параметр **date_req** в запросе отсутствует, то вы получите данные на последнюю зарегистрированную дату.

Чтобы делать запросы по адресу получения XML-файла, сначала мы устанавливаем соединение с точкой доступа для подключения к Интернету:

```
WiFi.mode(WIFI_STA);
WiFi.begin(ssid, password);
```

Затем создаем TCP-соединение с сервером **cbr.ru**:

```
WiFiClient client;
const int httpPort = 80;
if (!client.connect(host, httpPort)) {
  Serial.println("connection failed");
  return;
}
```

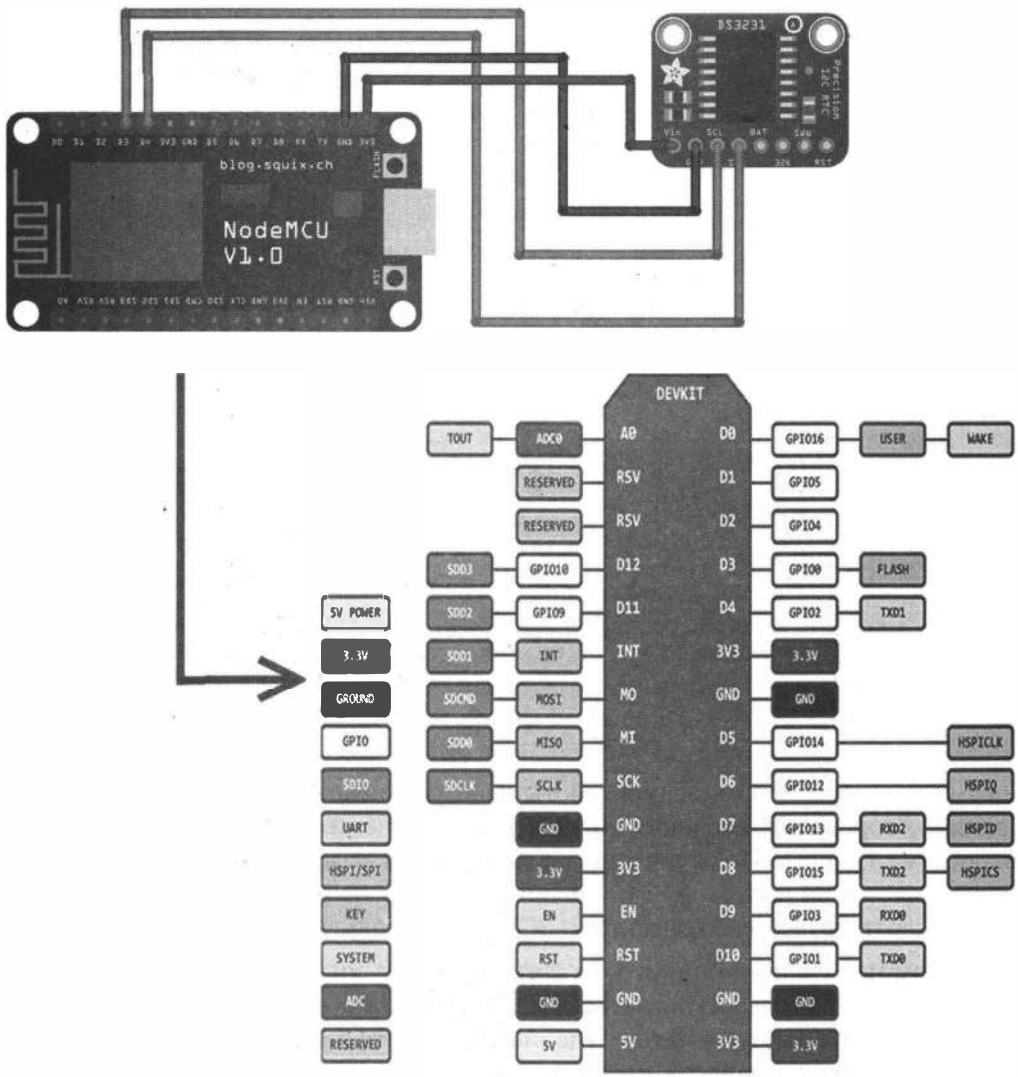


Рис. 6.33. Монтжная схема подключения модуля DS3231 к модулю NodeMCU ESP8266

Формируем строку для запроса XML-файла:

```
String url = "/scripts/XML_daily.asp?date_req="+getDateStr();

Serial.print("Requesting URL: ");
Serial.println(url);
```

Процедура `getDateStr()` выдает строку в формате **dd/mm/yyyy** на текущий день:

```
  String getDateStr() {
  String str = String(day) + "/" + formatDigit(month, 2) + "/20" +
      formatDigit(year, 2);
  return str;
}
```

Отправляем данные на сервер:

```
client.print(String("GET ") + url + " HTTP/1.1\r\n" +
             "Host: " + host + "\r\n" +
             "Connection: close\r\n\r\n");
delay(10);
```

и выводим в последовательный порт ответ сервера:

```
String line;
while(client.available()){
  line = client.readStringUntil('\r');
  Serial.print(line);
  delay(10);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный вариант рассмотренного скетча находится в папке *projects\CBRprinter\01* сопровождающего книгу электронного архива (см. *приложение*).

Загружаем этот скетч в плату NodeMCU и видим результат его работы в мониторе последовательного порта (рис. 6.34).

Работа со строками в Arduino достаточно проблемна, поэтому внесем в наш скетч следующие изменения:

```
int k=0;
while(client.available()){
  line = client.readStringUntil('\r');
  if(k>0) k++;
  if(line.substring(11,14)=="840")
    k=1;
  if(k==5) {
    if(dollar==line.substring(9,16)) ;  // если курс изменился -
                                        // выводим на печать
     else {
      dollar=line.substring(9,16);
      printer.println();
      printer.println(getTimeStr());
      printer.print("$= ");
      printer.print(dollar);
      printer.print(" rub");
      printer.println();
    }
  }
  delay(10);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный вариант рассмотренного скетча находится в папке *projects\CBRprinter\02* сопровождающего книгу электронного архива (см. *приложение*).

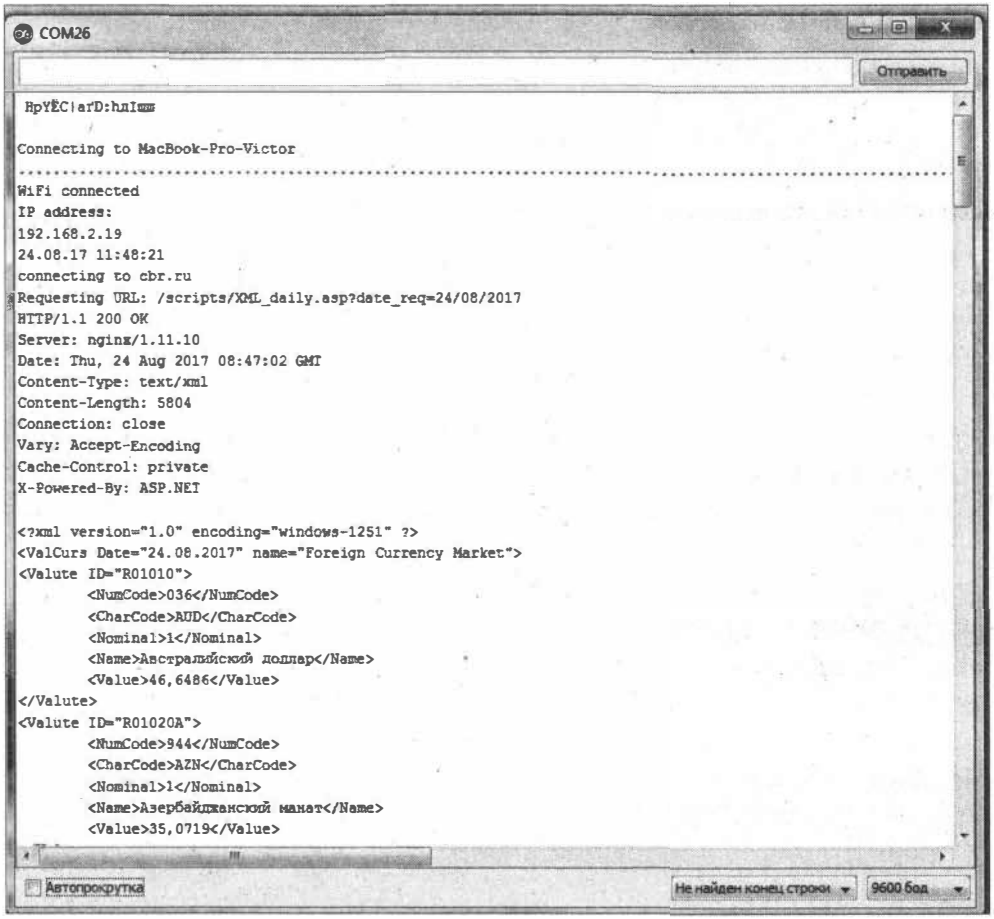


Рис. 6.34. Результат получения XML-файла с сайта **cbr.ru** в мониторе последовательного порта

Рис. 6.35. Вывод курса доллара с сайта **cbr.ru** на принтер

Загружаем этот скетч в плату NodeMCU и видим, как на печать выводится курс доллара (рис. 6.35).

6.5. GPS-трекер и онлайн-сервис поиска стоянок

В этом проекте мы получаем с модуля GPS-приемника данные о местоположения движущего средства и отправляем их в сервис Яндекс.Карты. А сервис использует эти данные для выбора свободной стоянки и построения маршрута.

6.5.1. Подключение GPS-модуля к плате Arduino

GPS (Global Positioning System) — это система, позволяющая с точностью не менее 100 м определить местоположение объекта, т. е. его широту, долготу и высоту над уровнем моря, а также направление и скорость его движения. Кроме того, с помощью GPS можно определить время с точностью до 1 наносекунды. GPS состоит из совокупности определенного количества искусственных спутников Земли (спутниковой системы NAVSTAR) и наземных станций слежения, объединенных в общую сеть. В качестве абонентского оборудования служат индивидуальные GPS-приемники, способные принимать сигналы со спутников и по принятой информации вычислять свое местоположение. Мы используем GPS-приемник V.KEL VK16E (рис. 6.36).

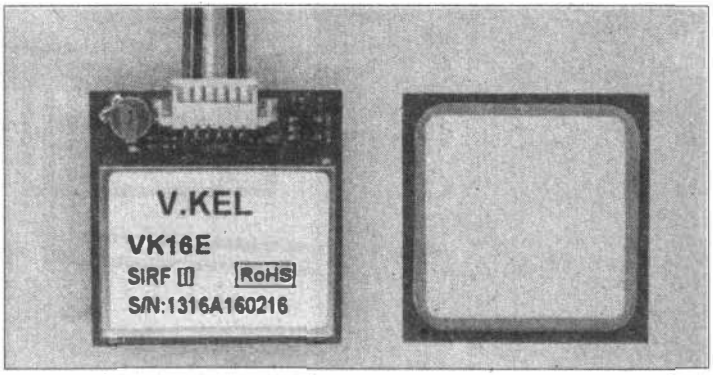


Рис. 6.36. GPS-приемник V.KEL VK16E

Холодный старт GPS-приемника происходит, когда он был выключен длительное время, перемещался в выключенном состоянии на значительное расстояние или его часы не совпадают с данными спутника. При холодном старте со спутников скачивается информация, называемая *альманах*. Время обновления альманаха — от 5 до 15 минут в зависимости от условий приема и количества видимых спутников. Особенность — приемник в это время должен быть неподвижен. *Теплый старт* происходит, когда приемник был выключен более 30 мин. При этом на него скачиваются уточняющие данные. Занимает этот процесс 0,5–1,5 мин. *Горячий старт* происхо-

дит, когда приемник был отключен непродолжительное время. Данные считаются свежими, и приемник просто находит спутники (опираясь на данные альманаха).

Для проверки работоспособности системы подключите модуль GPS-приемника по последовательному порту к компьютеру с ОС Windows, используя адаптер USB-TTL, и запустите программу MiniGPS_v1.7.1.exe. Программа показывает количество найденных приемником спутников и в случае их достаточного числа выводит ваше местоположение: географические широту и долготу. Мигание зеленого светодиода сигнализирует о достаточном для определения положения количестве спутников.

Теперь подключите GPS-приемник к плате Arduino (монтажная схема подключения показана на рис. 6.37) и загрузите в нее скетч из листинга 6.9. В скетче задействованы библиотека SoftwareSerial и библиотека TinyGPS, позволяющая выделять нужные данные из всего потока, передаваемого приемником. Откройте монитор последовательного порта — в него начнут выводиться данные о местоположении (географические широта и долгота), а также текущая дата и время по Гринвичу (рис. 6.38).

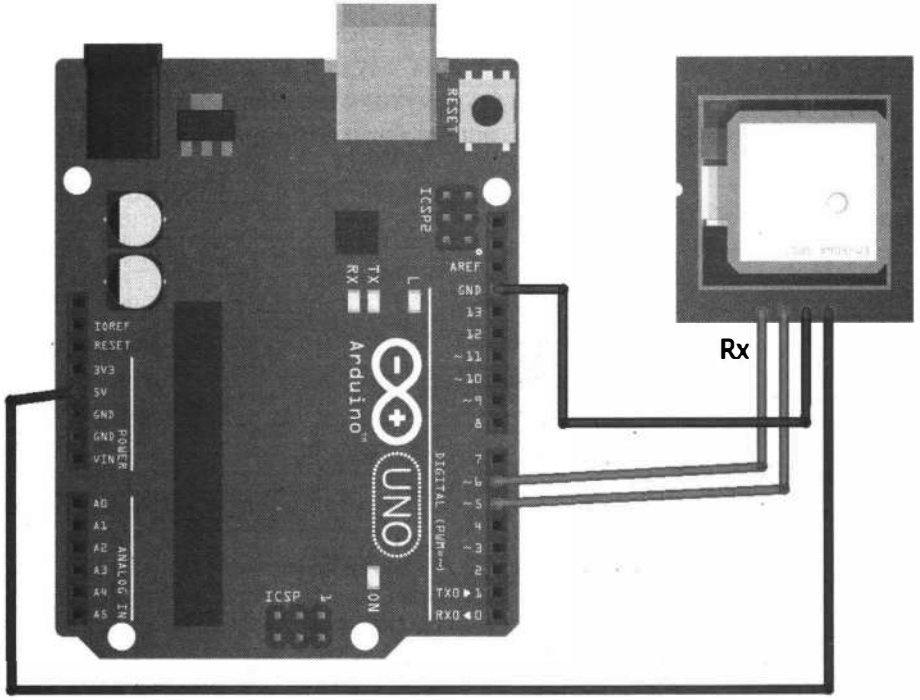


Рис. 6.37. Монтажная схема подключения GPS-приемника к плате Arduino

Листинг 6.9

```
// подключение библиотек
#include <SoftwareSerial.h>
#include <TinyGPS.h>
```

```
// создание экземпляров объектов
TinyGPS gps;
SoftwareSerial gpsSerial(5, 6);

bool newdata = false;
unsigned long start;
long lat, lon;
unsigned long time, date;

void setup() {
   gpsSerial.begin(9600); // скорость обмена с GPS-приемником
   Serial.begin(9600);
   Serial.println("Waiting data of GPS...");
}

void loop() {
    // задержка в секунду между обновлениями координат
   if (millis() - start > 1000) {
      newdata = readgps();
      if (newdata) {
         start = millis();
         gps.get_position(&lat, &lon);
         gps.get_datetime(&date, &time);
         Serial.print("Lat: "); Serial.print(lat);
         Serial.print(" Long: "); Serial.print(lon);
         Serial.print(" Date: "); Serial.print(date);
         Serial.print(" Time: "); Serial.println(time);
      }
   }
}
// проверка наличия данных
bool readgps() {
   while (gpsSerial.available()) {
      int b = gpsSerial.read();
      //в TinyGPS есть ошибка: не обрабатываются данные с \r и \n
      if('\r' != b) {
         if (gps.encode(b))
         return true;
      }
   }
   return false;
}
```

Электронный архив

Полный вариант рассмотренного скетча находится в папке *projects\GPS\01* сопровождающего книгу электронного архива (см. *приложение*).

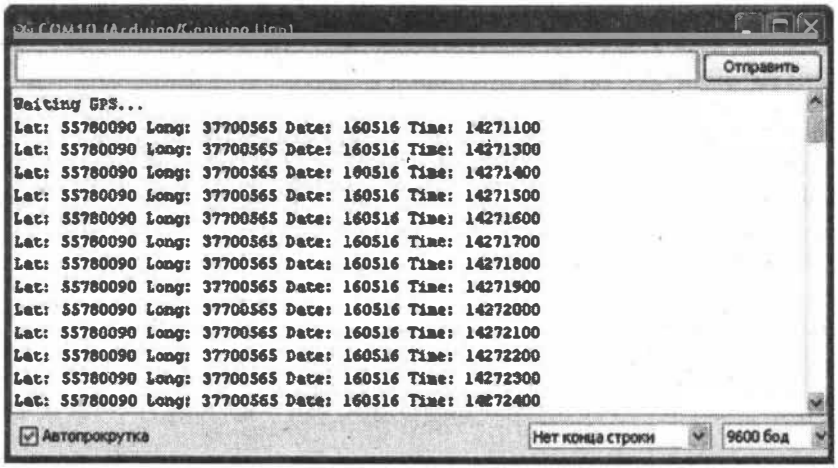


Рис. 6.38. Получение данных с GPS-модуля

6.5.2. Отправка данных по GPRS на сервер

Плата расширения Arduino GPRS/GSM Shield предоставляет возможность использовать сеть мобильной GSM-связи для удаленного приема и передачи данных в Arduino-проектах.

Один из вариантов такой платы — SIM900 Quad-Band GPRS Shield на основе чипа SIM900 (рис. 6.39).

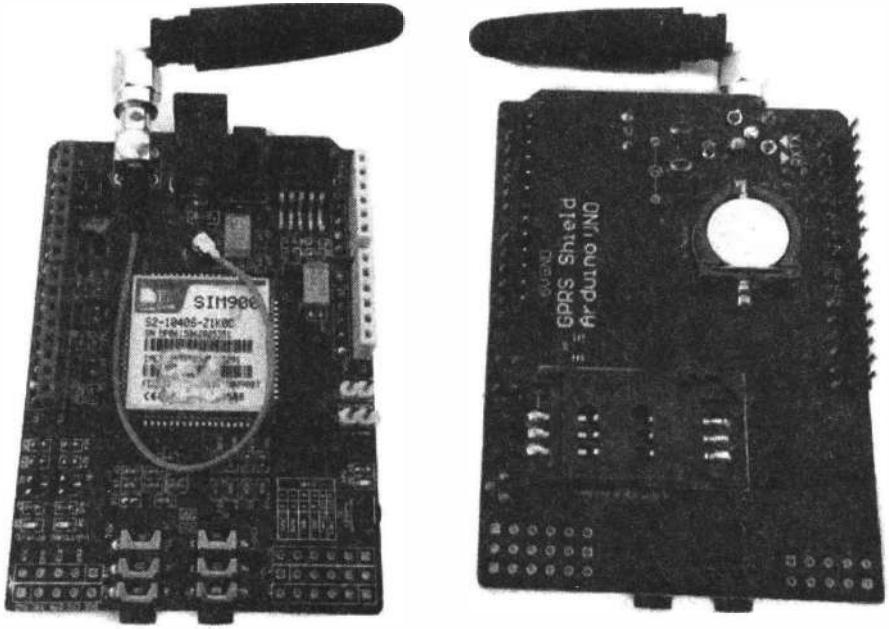


Рис. 6.39. Плата расширения Arduino SIM900 Quad-Band GPRS Shield

Примечание

Работу с шилдом SIM900 Quad-Band GPRS Shield мы в общих чертах рассмотрели в *разд. 3.3.1*. Чтобы не заставлять вас листать книгу туда-сюда, некоторые положения этого раздела здесь будут повторены.

Общение с платой расширения осуществляется по последовательному интерфейсу (через serial-соединение) с использованием набора AT-команд. С помощью перемычек на плате можно установить задействуемые для коммуникации контакты: аппаратный последовательный порт платы Arduino D1/D0 (или D2/D3 на некоторых платах) или программный последовательный порт D8/D7 (для работы с использованием библиотеки SoftwareSerial).

Плату расширения GSM/GPRS Shield SIM900 можно включить двумя способами:

- аппаратным (нажатием кнопки PWRKEY);
- программным.

Для чего нужен программный сброс?

Иногда при обмене по GPRS возникают ситуации, после которых модуль может зависнуть. Причиной этого могут стать некорректные данные, пришедшие по сети и загнавшие чип SIM900 в ступор, или помехи на линии обмена модуля и контроллера, или еще какие бы то ни было неведомые проблемы. Производитель чипа предлагает в таких случаях перезагружать модуль с помощью специальной последовательности импульсов, подаваемых на вход PWRKEY (рис. 6.40).

Однако это не всегда помогает. Самым правильным способом перезагрузки модуля является полное снятие с него питания, выдержка некоторой паузы (хотя бы секунду на всякий случай) и повторная подача питания. Так что для перезагрузки модуля необходимо предусмотреть на плате реле или транзисторный ключ, управляемый контроллером.

И действительно, в реальных условиях программный сброс через D9 не работал, поэтому пришлось подключаться перемычкой и управлять сбросом с цифрового выхода D4 платы Arduino.

Монтажная схема соединений устройств для отправки данных по GPRS на сервер показана на рис. 6.41.

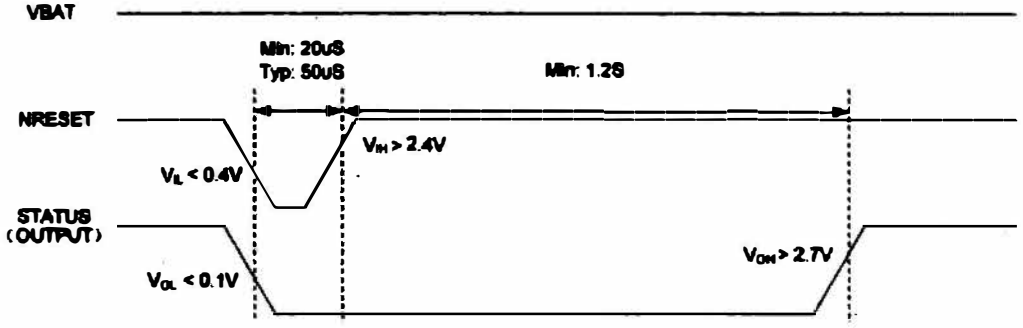


Рис. 6.40. Программный сброс для GSM/GPRS Shield SIM900

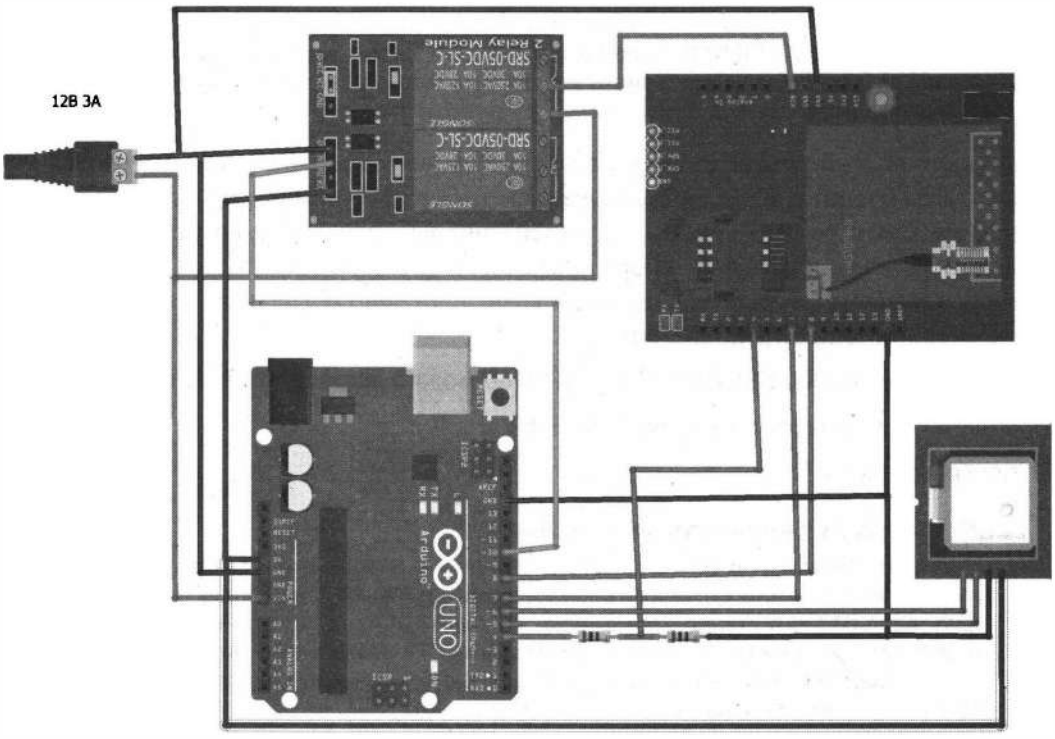


Рис. 6.41. Схема подключения модуля GPS/GPRS Shield к плате Arduino

Рассмотрим сначала управление модулем GPRS/GSM Shield с помощью AT-команд. Для этого установим модуль на плату Arduino и подключим ее к компьютеру. Arduino-скетч отправки и получения данных между компьютером и модулем GPRS/GSM Shield через плату Arduino показан в листинге 6.10.

Листинг 6.10

```
// подключение библиотеки
#include <SoftwareSerial.h>
// создание объекта
SoftwareSerial grs(7, 8); // RX, TX
// скорость обмена
 #define GSMbaud 19200

 String str1;
 char buff[100];

 void setup() {
    Serial.begin(9600);
    gsm.begin(GSMbaud);
    Serial.println("Start");
}
```

```
void loop() {
   if (Serial.available()) {
      str1 = Serial.readStringUntil('\n');
      str1.toCharArray(buffer, hh.length() + 1);
      gsm.write(buffer);
      gsm.board.write('\n');
   }
   if (gsm.available()) {
      Serial.write(gprs.read());
   }
}
```

Данные (с GPS-трекера) мы будем отправлять на сервер по адресу PHP-скрипта:

http://freeparkovka.ru/getkoords.php&id=xxx&lat=yyyyyyy&lon=zzzzzzzzz

обрабатывающего GET-данные и записывающие их в базу данных.

Для отправки данных нужно послать следующие AT-команды:

```
AT+SAPBR=1,1    //Открыть несущую (Carrier)
//тип подключения - GPRS
AT+SAPBR=3,1,"CONTYPE","GPRS"
//APN, для Билайна - internet
AT+SAPBR=3,1,"APN","internet.beeline.ru"
//Инициализировать HTTP
AT+HTTPINIT
//Carrier ID для использования.
AT+HTTPPARA="CID",1
//Отправить данные методом GET
AT+HTTPPARA="URL","http:/freeparkovka.ru/gps_tracker1.php?id=1&lat=
                                                        XXXXXlon=YYYYY"
AT+HTTPACTION=0
//дождаться ответа
AT+HTTPREAD
//остановить HTTP
AT+HTTPTERM
```

Пробуем отправить эту последовательность из монитора последовательного порта (рис. 6.42).

Теперь создаем скетч. Функция `sendATcommand()` (листинг 6.11) отправляет AT-команды и возвращает `True` в случае ожидаемого ответа и `False` при получении любого другого.

Листинг 6.11

```
boolean sendATcommand(char* ATcommand, char* expected_answer,
                                                    unsigned int timeout)
  {
   uint8_t x=0;
   boolean answer=false;
```

Рис. 6.42. Отправка AT-команд на модуль GPRS/GSM Shield из последовательного порта

```
char response[150];
unsigned long previous;

memset(response, '\0', 150);    // Initialize the string
delay(100);
while( GPRSSerial.available() > 0)
   GPRSSerial.read();    // очистить буфер данных
Serial.println(ATcommand);
GPRSSerial.println(ATcommand);    // Send the AT command
x = 0;
previous = millis();
// ждем ответ
do{
  if(GPRSSerial.available() != 0)
    {
    //
    response[x] = GPRSSerial.read();
    x++;
    // сравнение на совпадение
    if (strstr(response, expected_answer) != NULL)
      {
      answer = true;
      }
    }
 }
 // контролировать время ожидания
 while((answer == false) && ((millis() - previous) < timeout));
```

```
    Serial.println(response);
        return answer;
}
```

За отправку группы AT-команд отвечают функции:

- `iniGPRS()` — инициализация GPRS и подключение к сети оператора;
- `sendkoords()` — инициализация HTTP и отправка данных серверу;
- `resetGPRS()` — программный сброс;
- `onoffGPRS()` — перезагрузка модуля выключением/включением питания.

Содержимое основной части скетча показано в листинге 6.12.

Листинг 6.12

```
// подключение библиотеки SoftwareSerial
#include <SoftwareSerial.h>
//Подключение дисплея lcd1602 i2c
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
//Подключение библиотеки для gps
#include <TinyGPS.h>

// Создание объектов библиотек
TinyGPS gps;
SoftwareSerial SerialGps(5, 6);
SoftwareSerial GPRSSerial(7, 8); // RX, TX

//Служебные переменные
unsigned long startGPS;
bool newDataGps=false;

unsigned long millisGPRS;
unsigned long millisGPS;

float lat=44.029678;
float lon=43.072151;
// кол-во ошибок
int error1=0;
int error2=0;
int error3=0;

void setup() {
  GPRSSerial.begin(19200);
  Serial.begin(9600);
  delay(3000);
  Serial.println("iniGPRS - start ");
```

```
  while(!iniGPRS()) {
     error2++;
     if(error2>3) {
        while(!resetGPRS()) {
           error1++;
           if(error1>3) {
              onoffGPRS();
              error1=0;
           }
        }
     error2=0;
     }
  }
  Serial.println("iniGPRS - ok ");
}

void loop() {
// получение каждые 5 секунд
if (millis()-millisGPS >= 3000)
 {
  GPRSSerial.end();
  SerialGps.begin(9600);
  getGPS();

  millisGPS=millis();
 }
// отправка каждые 10 секунд
if (millis()-millisGPRS >= 10000)
 {
  SerialGps.end();
  GPRSSerial.begin(19200);
  if(!sendkoords()) {
     if(error3>3) {
        while(!iniGPRS()) {
           error2++;
           if(error2>3) {
              while(!resetGPRS()) {
                 error1++;
                 if(error1>3) {
                    onoffGPRS();
                    error1=0;
                 }
              }
           error2=0;
           }
        }
     //
     }
  }
```

```
  error3=0;
  millisGPRS=millis();
 }
}
```

Электронный архив

Полный вариант рассмотренного скетча находится в папке *projects\GPS\04* сопровождающего книгу электронного архива (см. *приложение*).

На стороне сервера данные принимает PHP-скрипт getkoords.php (листинг 6.13). Скрипт получает данные и записывает их в таблицу базы данных Users (рис. 6.43).

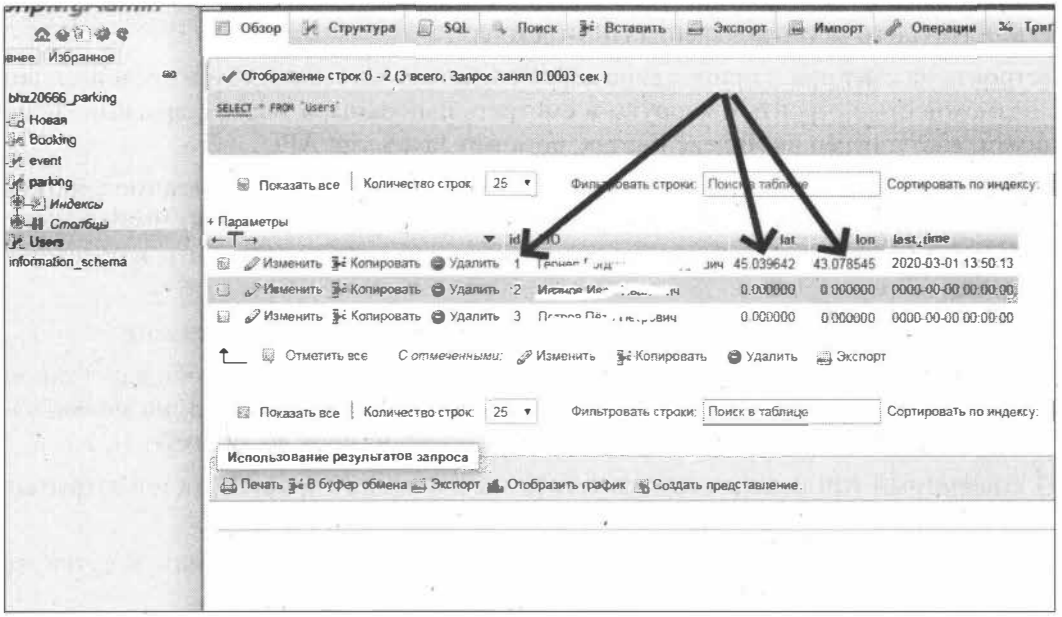


Рис. 6.43. Таблица базы данных Users

Листинг 6.13

```
<?php

$location="localhost";
$user="bhx20666_parking";
$pass="****************";
$db_name="bhx20666_parking";
//
if(! $db=mysqli_connect($location,$user,$pass,$db_name))
 {echo "connect error";}
else
 {;}
```

```
mysqli_query($db,"SET CHARACTER SET 'utf8'");

$query0=" SELECT * FROM Users WHERE id=".$_POST[iduser]." ";
$rez0=mysqli_query($db,$query0);
$row0=mysqli_fetch_assoc($rez0);
$str=$row0[lat].";".$row0[lon].";".$row0[last_time].";";
echo $str;

?>
```

6.5.3. Создание веб-страницы с использованием API Яндекс.Карт

Нам осталось купить хостинг, доменное имя и написать код страницы для отображения текущего местоположения GPS-трекера.

Встроить на сайт или в приложение карту с поиском по топонимам и организациям с возможностью строить маршруты и смотреть панорамы, а также с другими функциями, доступными на Яндекс.Картах, поможет JavaScript API.

С помощью JavaScript API вы можете настроить нужную логику взаимодействия пользователя с картой и определить, как эта карта будет выглядеть. Чтобы задать внешний вид объектов на карте, можно выбрать стандартные элементы или создать собственный макет.

Пользоваться API Яндекс.Карт можно бесплатно, если соблюдать условия:

- все данные должны отображаться на карте, размещенной на общедоступном сайте или в приложении. Сохранять или изменять данные нельзя, но можно кэшировать запросы к геокодеру и маршрутизатору на срок до 30 дней;
- бесплатный API нельзя использовать для мониторинга транспорта и в закрытых системах;
- общее число запросов к геокодеру, маршрутизатору и панорамам в сутки не должно превышать 25 тысяч.

Есть на бесплатное использование и другие ограничения. Подробнее о них можно прочитать в документации.

Для использования API Яндекс.Карт на своем сайте необходимо получить API-ключ (рис. 6.44), а также подключить к странице соответствующую библиотеку (разместить в разделе `header` страницы следующий код):

```
<script src="https://api-maps.yandex.ru/2.1/?apikey=75e85ef2-7142-428b-8bd0-
785b0ba84f55&lang=ru_RU" type="text/javascript"></script>
```

Теперь мы можем задействовать все возможности библиотеки, создавать карту и размещать на ней объекты (рис. 6.45).

Код javascript создания карты:

```
myMap = new ymaps.Map("map", {
                center: [44.036578, 43.074971],
                zoom: 12
            });
```

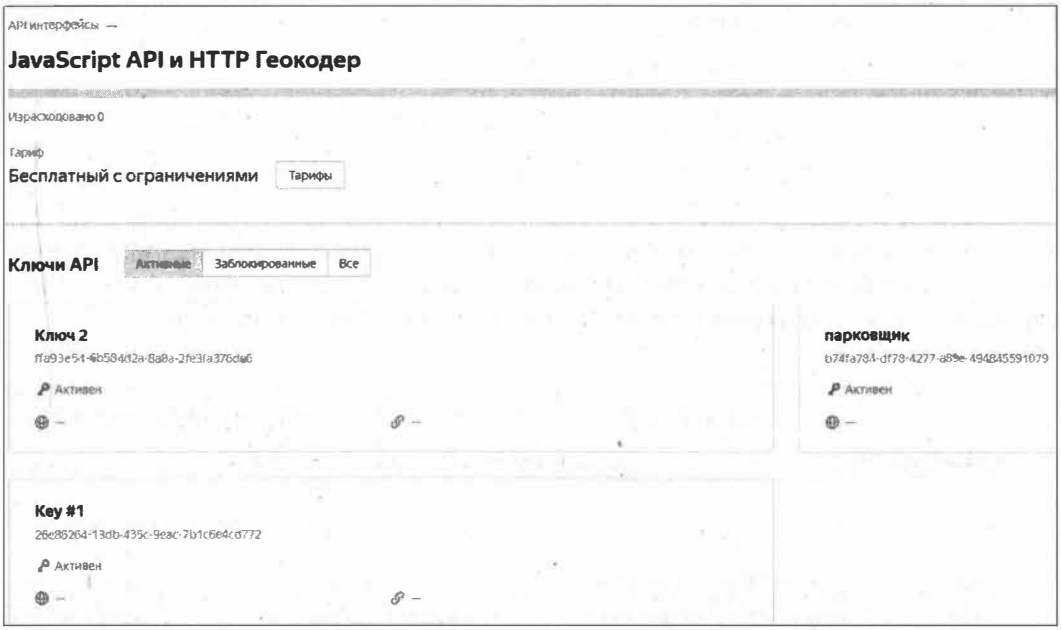


Рис. 6.44. Ключи API

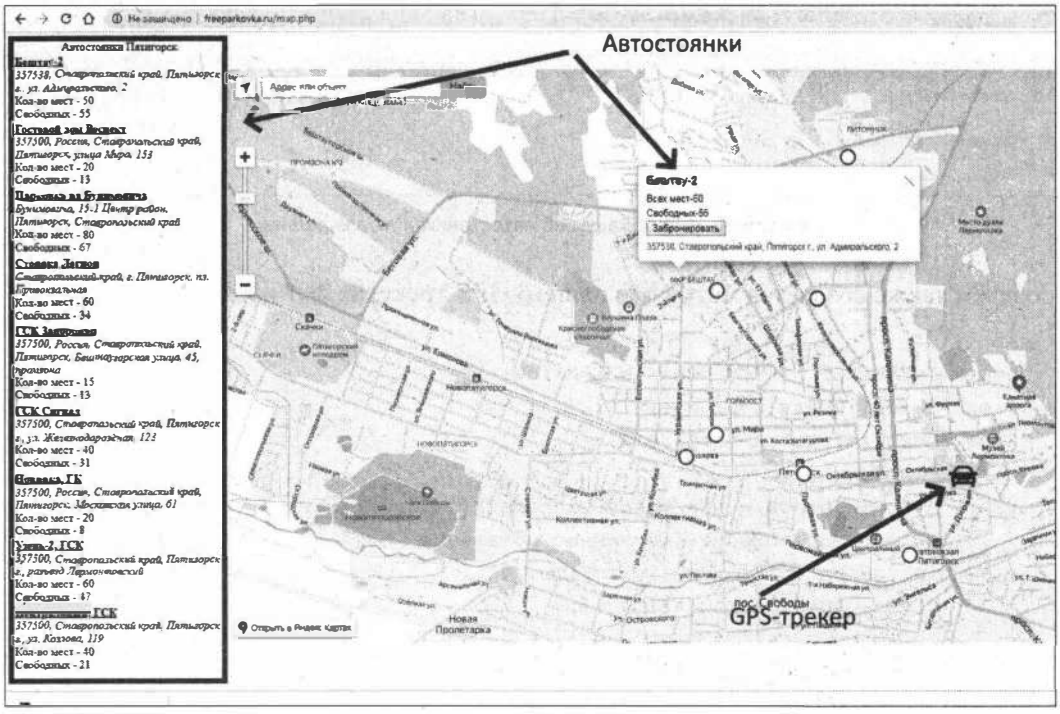


Рис. 6.45. Страница с картой на сайте

Создаем коллекцию объектов:

```
myGeoObjects = new ymaps.GeoObjectCollection({}, {
               preset: "islands#blueCircleIcon",
               strokeWidth: 4,
               geodesic: true
           });
```

в которую заносим метки для всех автостоянок — данные берем из базы MySQL (рис. 6.46). Код создания метки, координат, содержимого балуна формируется PHP-скриптом. Скриптом формируется и список всех автостоянок в таблице.

Профилирование [Построчное редактирование

5 ▾ Фильтровать строки: Поиск в таблице Сортировать по индексу: Ниодного ▾

id	name	adress	info	lat	lon	all_places	free_places
1	Бештау-2	357538, Ставропольский край, Пятигорск г., ул. Адм...		44.054466	43.040054	50	55
2	Гостевой дом Респект	357500, Россия, Ставропольский край, Пятигорск, ул...		44.040119	43.044682	20	13
3	Парковка на Бунимовича	Бунимовича, 15-1 Центр район, Пятигорск, Ставрополь...	-Общедоступная -Бесплатная	44.029716	43.068108	80	67
4	Стоянка Легион	Ставропольский край, г. Пятигорск, пл. Привокзальн...		44.036770	43.055325	60	34
5	ГСК Запорожец	357500, Россия, Ставропольский край, Пятигорск, Бе...		44.052574	43.044807	15	13
6	ГСК Сигнал	357500, Ставропольский край, Пятигорск г., ул. Жел...		44.057716	43.049774	40	31
7	Новинка, ГК	357500, Россия, Ставропольский край, Пятигорск, Мо...		44.051899	43.056980	20	8
8	Узень-2, ГСК	357500, Ставропольский край, Пятигорск г., разъезд...		44.064014	43.060635	60	47
9	Центральный, ГСК	357500, Ставропольский край, Пятигорск г., ул. Коз...		44.038162	43.041115	40	21

Изменить Копировать Удалить Экспорт

Рис. 6.46. Информация об автостоянках в базе данных

Создаем также создается метку для нашего GPS-трекера (PHP-код):

```
<?php
   $query1="SELECT * FROM Users WHERE id=1 ";
   $rez1=mysqli_query($db,$query1);
   $row1=mysqli_fetch_assoc($rez1);

   $str="AvtoPlacemark = new ymaps.Placemark([".$row1[lat].",".$row1[lon]."],
       {balloonContentBody: 'Последние показания <br>".$row1[last_time]."',
        hintContent: '".$row1[last_time]."'},
        {iconLayout: 'default#image',
        iconImageHref: 'iconavto.jpg',
        iconImageSize: [40, 40],
        iconImageOffset: [-20, -20]});";
   echo $str;
  ?>
```

Для обновления информации о положении GPS-трекера, изменении информации и свободных местах на стоянках, отправки запроса бронирования используется технология AJAX. Пример:

```
<script src="js/jquery-3.3.1.min.js"></script>
    <script type="text/javascript">
          ..............
          function getKoordsAvto() {
              $.ajax({
              type: 'POST',
              url: 'getkoords.php',
              cache: false,
              data: {"iduser": 1 },
              success: function (data) {
                 var arr=data.split(";");
                 AvtoPlacemark.geometry.setCoordinates([arr[0],arr[1]]);
              }
            });
          setTimeout(getKoordsAvto,10000);
          }
        ..................................
  </script>
```

По нажатию кнопки Забронировать отправляется запрос на сервер для перенаправления запроса на автостоянки, и на карте строится маршрут (рис. 6.47).

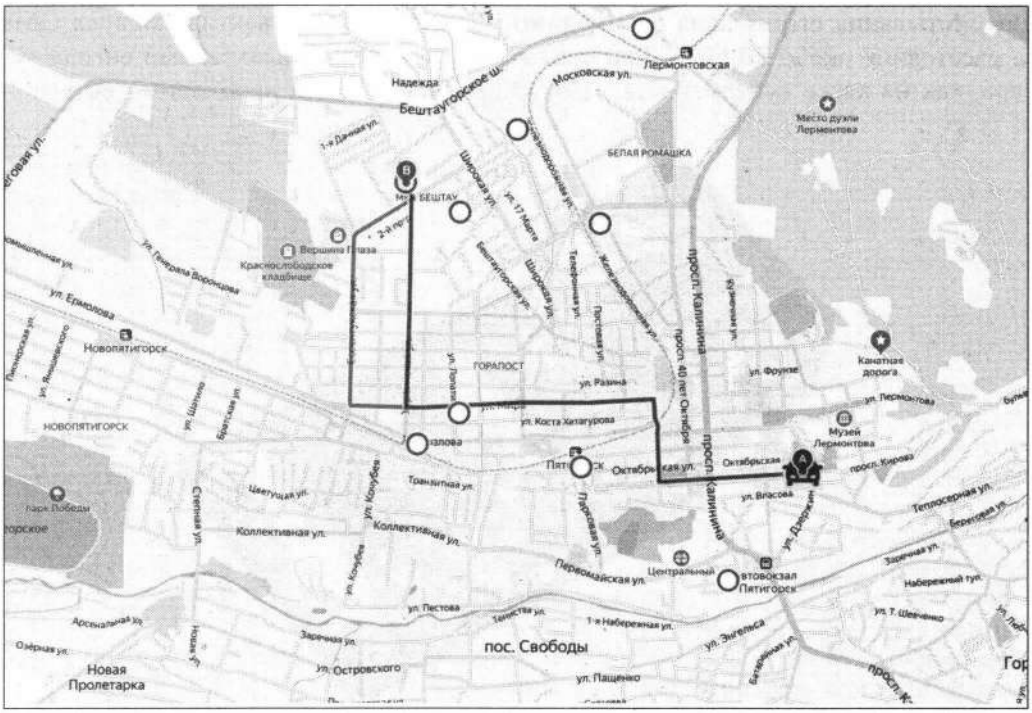


Рис. 6.47. Построение маршрута

ЭЛЕКТРОННЫЙ АРХИВ

Файлы сайта можно найти в папке *projects\GPS\site* сопровождающего книгу электронного архива (см. *приложение*).

6.6. IoT-сканер штрих-кодов с отправкой результатов в облако

В этом разделе мы рассмотрим создание IoT-сканера штрих-кодов, который позволит проводить инвентаризацию товаров с отправкой результатов на сервер в сети Интернет и с сохранением их в базе данных. Процесс осуществляется так: сканер штрих-кода GM65 идентифицирует продукт по его штрих-коду. Затем мы вводим количество продукта на сенсорном дисплее Nextion и отправляем информацию через Wi-Fi на сервер, где вводим ее в базу данных. Такое устройство позволяет автоматизировать процесс инвентаризации товаров.

6.6.1. Сканер штрих-кодов GM65

Сканер штрих-кодов GM65 (рис. 6.48) — это цифровая камера и модуль обработки изображений. Его алгоритм распознает штрих-коды и QR-коды, попавшие в поле зрения камеры, и, если ему не хватает внешнего освещения, включает встроенную светодиодную подсветку. Для точного наведения сканера на штрих-код предусмотрен световой маркер в виде красной полосы.

Для считывания штрих-кода необходимо поднести его к объективу модуля GM65 на расстояние около 20 см — при считывании раздается характерный сигнал зуммера, как на кассе супермаркета. Для ускорения процесса необходимо выровнять

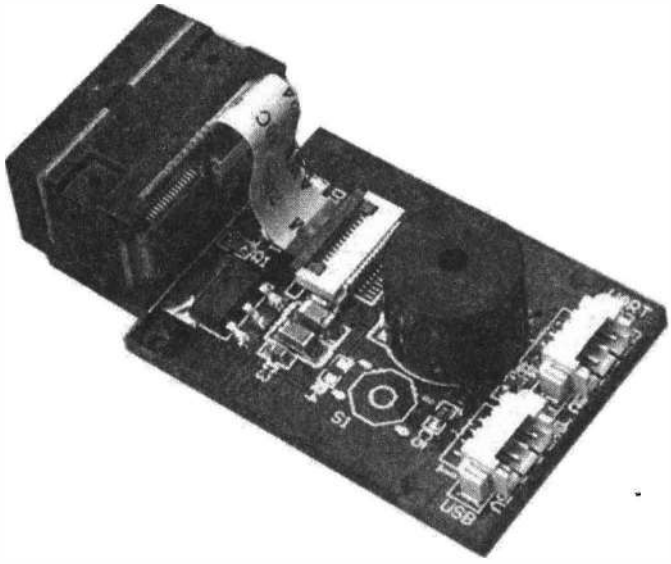


Рис. 6.48. Сканер штрих-кодов GM65

плоскость штрихкода перпендикулярно сканеру. Максимальный угол отклонения составляет 60 градусов.

В режиме по умолчанию сканер штрих-кода подключается к компьютеру по USB, работает как HID-клавиатура и выводит данные в виде строки текста.

Вы можете настроить сканер с помощью команд UART, но гораздо проще использовать сервис QR-кодов: переключать режимы чтения, управлять светодиодом и зуммером, сохранять и сбрасывать настройки, просто ориентируясь на соответствующий QR-код в инструкции устройства. Это позволяет изменять конфигурацию «на лету».

Кнопка на плате используется по умолчанию для активации процесса сканирования. Встроенный зуммер сигнализирует об успешном считывании кода и изменениях в работе устройства.

6.6.2. Подключение и настройка сканера штрих-кодов GM65

В качестве контроллера в этом проекте мы воспользуемся платой Arduino MKR1000 WiFi. Монтажная схема подключения сканера штрих-кодов GM65 к плате Arduino показана на рис. 6.49. Для подключения по UART задействуем последовательный порт Serial1 платы Arduino MKR1000 WiFi.

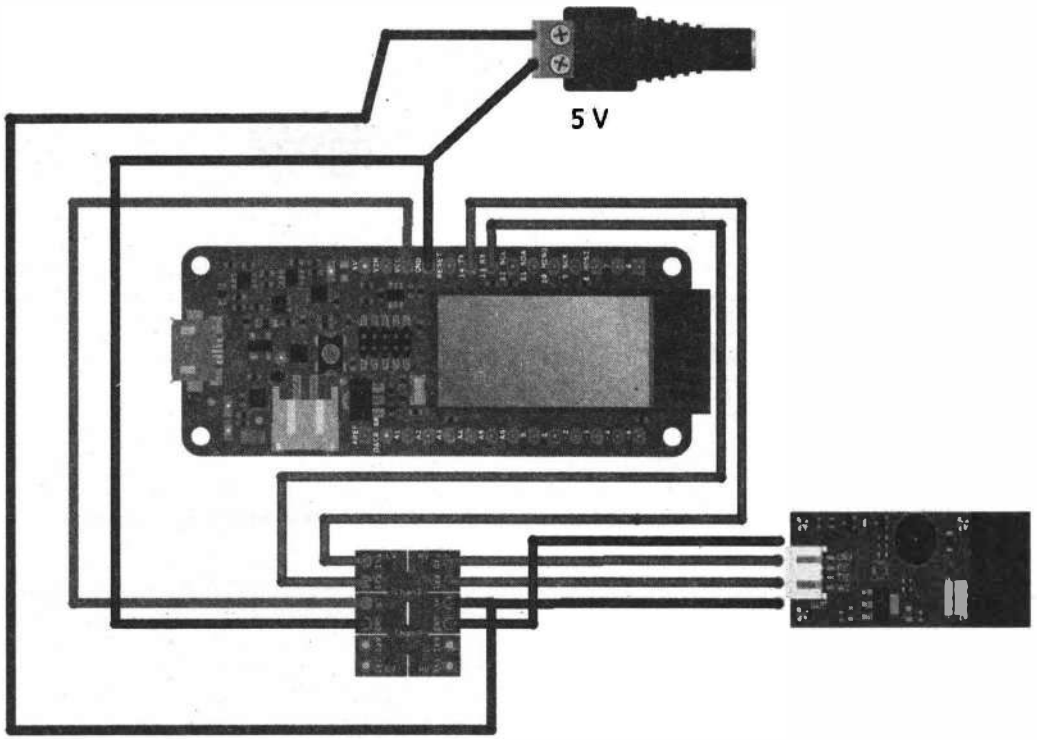


Рис. 6.49. Монтажная схема подключения сканера штрих-кодов GM65 к плате Arduino MKR1000 WiFi

Теперь необходимо соответствующим образом настроить сканер. Подключим сканер к компьютеру с помощью USB-кабеля и последовательно считаем QR-коды, показанные на рис. 6.50. Скетч получения кода со сканера представлен в листинге 6.14.

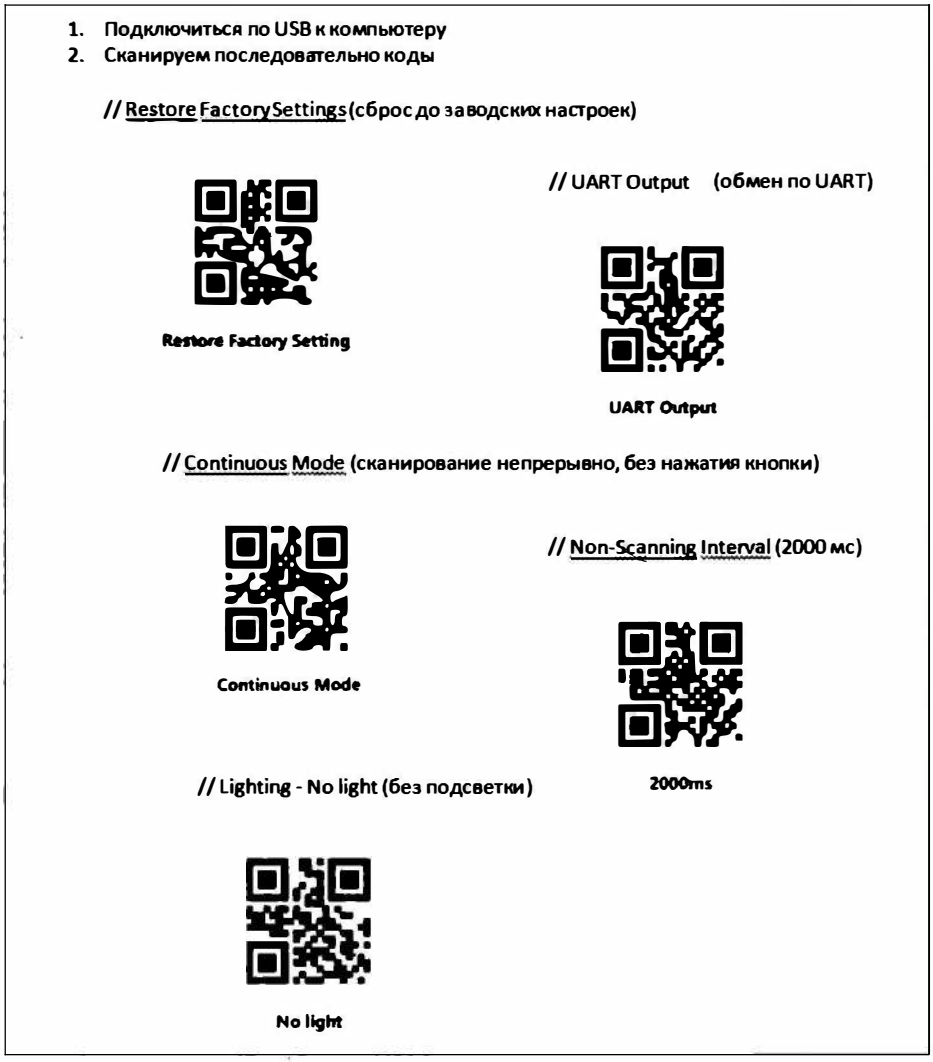


Рис. 6.50. Последовательность QR-кодов для настройки сканера штрих-кодов GM65 для проекта

Листинг 6.14

```
// данные, пришедшие из последовательного порта
String inputString = "";
// все данные получены
boolean stringComplete = false;
int countstr=0;
```

```
// для поиска окончания
unsigned long millisendstr=0;
void setup() {
   // запуск последовательных портов
   Serial.begin(9600);
   Serial1.begin(9600);
   // резервирование 50 bytes для inputString:
   inputString.reserve(50);
}
void loop() {
   // получение данных по Serial1
   serialScanerEvent();
   // при окончании передачи
   if (stringComplete) {
      Serial.println(inputString);
     // очистить строку
     inputString = "";
     stringComplete = false;
     countstr=0;
   }
}
// функция получения данных по Serial1
void serialScanerEvent() {
   //
   if (Serial1.available()>0) {
     // получить байт из буфера:
     char inChar = (char)Serial1.read();
     // добавить в строку:
     inputString += inChar;
     countstr++;
     millisendstr=millis();
   }
   else { // окончание передачи
      if(millis()-millisendstr>1000 && countstr>0) {
         stringComplete=true;
      }
   }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.14, можно найти в папке *projects\barcode\01* сопровождающего книгу электронного архива (см. *приложение*).

Загружаем скетч на плату Arduino и сканируем разные штрих-коды. Данные выводим в монитор последовательного порта (рис. 6.51).

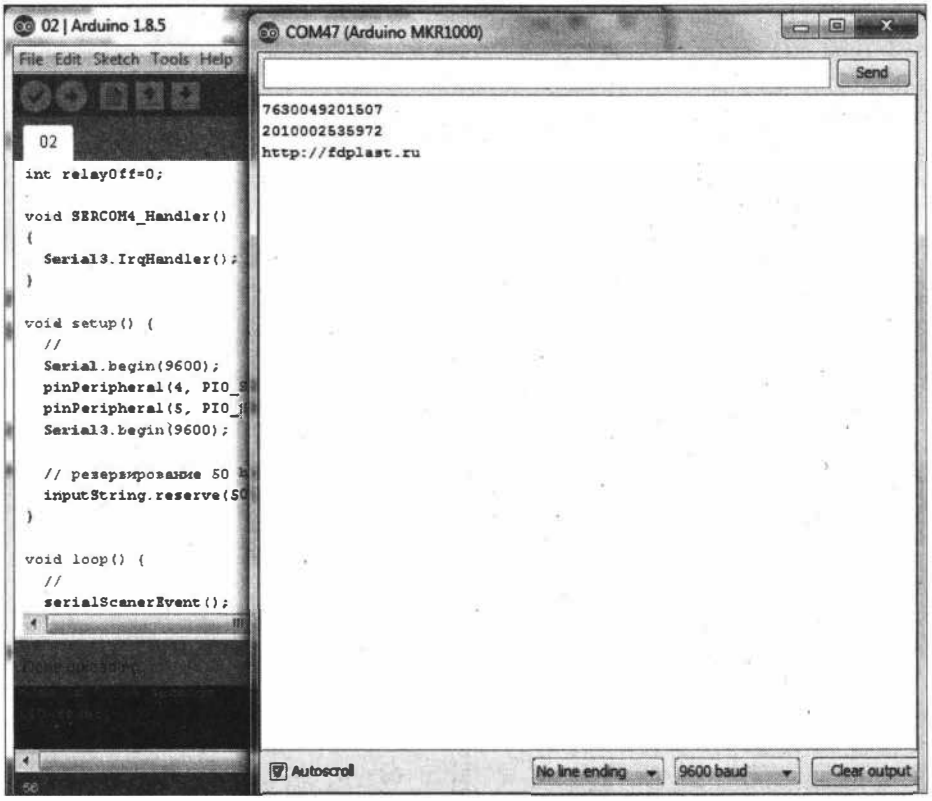


Рис. 6.51. Вывод кодов, полученных со сканера, в последовательный порт

6.6.3. Подключение дисплея Nextion

В качестве дисплея мы воспользуемся сенсорным дисплеем Nextion NX3224K024. В *разд. 6.3.2* о нем подробно рассказано, так что здесь мы просто загружаем изображения и создаем интерфейс приложения (рис. 6.52).

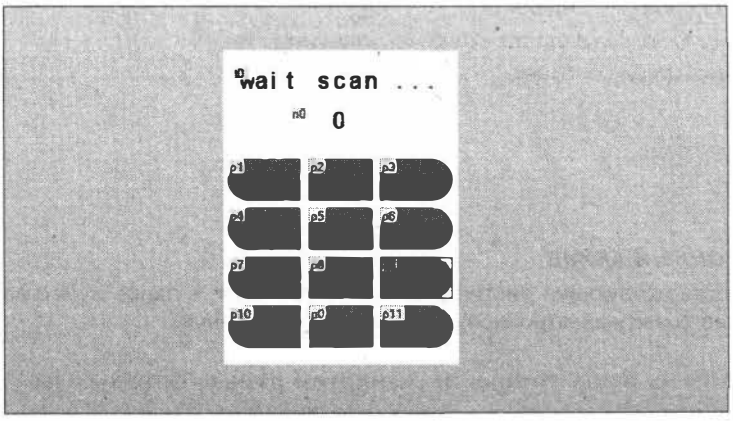


Рис. 6.52. Создание интерфейса в программе Nextion Editor

Назначьте события `Touch Release` для отправки команд через последовательный порт при нажатии кнопок (рис. 6.53). Вы можете просмотреть отправленные коды, запустив отладчик (рис. 6.54).

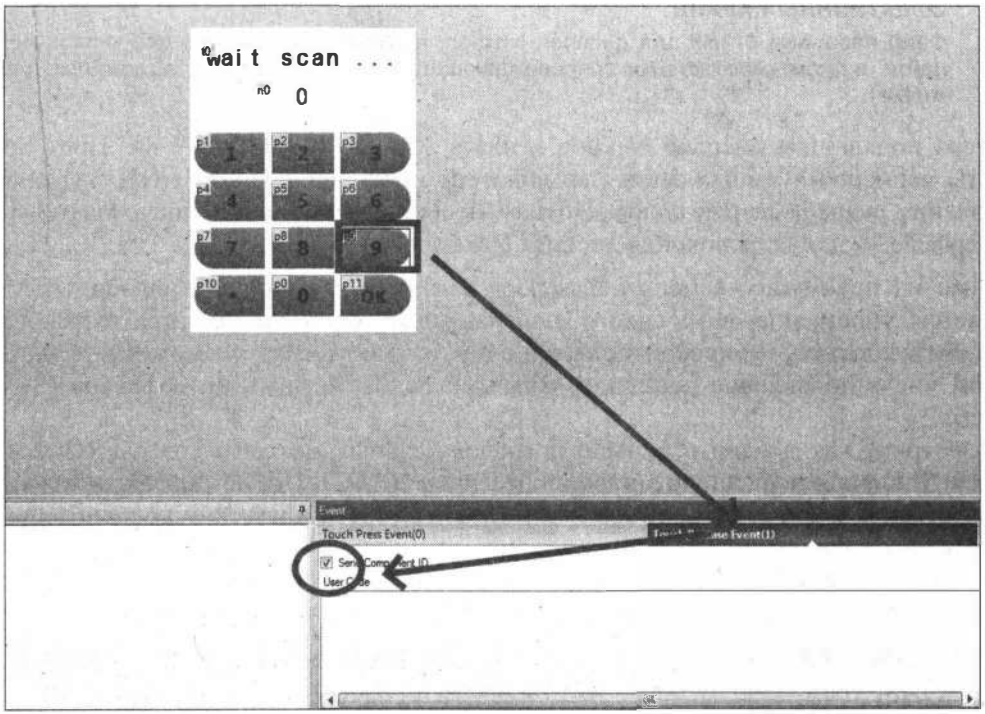


Рис. 6.53. Инициируем отправку команд для событий `Touch Release`

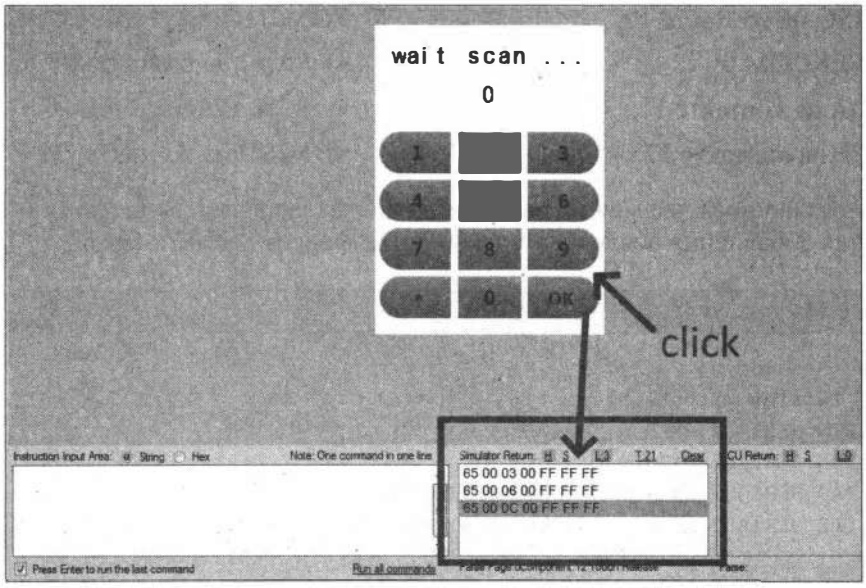


Рис. 6.54. Просмотр кодов для событий `Touch Release`

Для прошивки дисплея через UART понадобится адаптер USB-Serial. Подключение дисплея Nextion к адаптеру USB-Serial и загрузка в него прошивки подробно описаны в *разд. 6.3.2* (см. рис. 6.25 и 6.26).

> ***ЭЛЕКТРОННЫЙ АРХИВ***
>
> Файл прошивки *01.HMI* для дисплея Nextion, а также изображения для кнопок можно найти в папке *projects\Nextion* сопровождающего книгу электронного архива (см. *приложение*).

Теперь подключим дисплей Nextion к плате Arduino MKR1000 WiFi. Порт Serial платы мы используем для связи с компьютером и отладки. Порт Serial1 — для подключения сканера штрих-кодов. Но нам необходим еще один последовательный интерфейс — для подключения дисплея Nextion.

Одним из преимуществ новых платформ Arduino на микроконтроллерах SAMD является упрощение встроенного программного обеспечения, присваивающего каждому контакту микроконтроллера одну из множества возможных функций, в том числе добавление дополнительных последовательных интерфейсов (Serial2 и Serial3).

Рассмотрим, как добавить больше последовательных интерфейсов (SERCOM) на плату Arduino, оснащенную микропроцессором SAMD. Эти интерфейсы являются аппаратными и могут быть типов I^2C, UART и SPI. Подключение дополнительных интерфейсов возможно, поскольку микроконтроллер SAMD имеет шесть встроенных модулей для последовательной коммуникации, которые можно настраивать отдельно друг от друга. На момент приобретения платы настроены лишь четыре:

- SPI / SERCOM 1:
  - MOSI на контакте 8;
  - SCK на контакте 9;
  - MISO на контакте 10;
- I^2C / SERCOM 0:
  - SDA на контакте 11;
  - SCL на контакте 12;
- UART / SERCOM 5:
  - RX на контакте 13;
  - TX на контакте 14;
- WINC1500 SPI / SEERCOM 2:
  - MOSI на контакте 26;
  - SCK на контакте 27;
  - MISO на контакте 29.

Поэтому оставшиеся два можно вывести на контакты платы. В листинге 6.15 показан код для добавления последовательных интерфейсов Serial2 и Serial3.

Листинг 6.15

```
#include <Arduino.h>
#include <wiring_private.h>
#define PIN_SERIAL2_RX       (1ul)
#define PIN_SERIAL2_TX       (0ul)
#define PAD_SERIAL2_TX       (UART_TX_PAD_0)
#define PAD_SERIAL2_RX       (SERCOM_RX_PAD_1)

#define PIN_SERIAL3_RX       (5ul)
#define PIN_SERIAL3_TX       (4ul)
```

```
#define PAD_SERIAL3_TX       (UART_TX_PAD_2)
#define PAD_SERIAL3_RX       (SERCOM_RX_PAD_3)

 // Создание экземпляров Serial
Uart Serial2(&sercom3, PIN_SERIAL2_RX, PIN_SERIAL2_TX, PAD_SERIAL2_RX,
                                                        PAD_SERIAL2_TX);
Uart Serial3(&sercom4, PIN_SERIAL3_RX, PIN_SERIAL3_TX, PAD_SERIAL3_RX,
                                                        PAD_SERIAL3_TX);
 void SERCOM3_Handler()     { Serial2.IrqHandler(); }
 void SERCOM4_Handler()     { Serial3.IrqHandler(); }
```

Для подключения дисплея Nextion сконфигурируем Serial2 на контактах 0 и 1. Монтажная схема соединений показана на рис. 6.55.

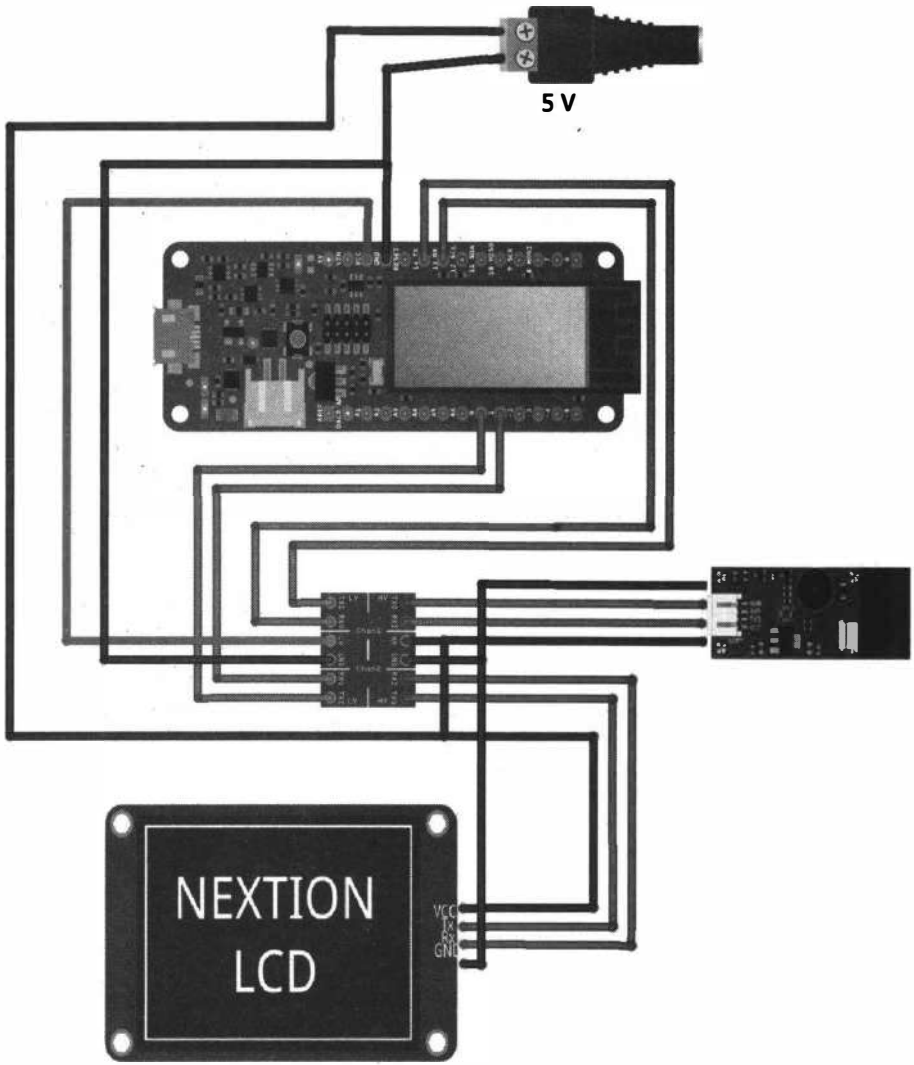


Рис. 6.55. Схема подключения дисплея Nextion и сканера штрих-кодов к плате Arduino MKR1000 WiFi

При нажатии кнопок на дисплее плата Arduino получает данные по последовательному порту Serial2. Необходимо переводить эти данные в количество (для кнопок 0–9), сброс (для кнопки *) или отправку данных на сервер (для кнопки OK). При изменении количества необходимо отправлять на дисплей последовательности для изменения содержимого элемента `n0` (см. рис. 6.52).

Код получения данных (нажатий кнопок) с дисплея Nextion и отправки данных с Arduino (штрих-код и количество) на дисплей Nextion показан в листинге 6.16.

Листинг 6.16

```
void loop() {
  //
  serialScanerEvent1();
  if (stringComplete1) {
    Serial.println(inputString1);
    Serial2.print("t0.txt=\"");
    Serial2.print(inputString1);
    Serial2.print("\"");
    Serial2.write(0xff);
    Serial2.write(0xff);
    Serial2.write(0xff);
    // очистить строку
    inputString1 = "";
    stringComplete1 = false;
    countstr3=0;
   }
  serialNextionEvent2();
  if (stringComplete2) {
    Serial.println(inputString2);
    parse_message(inputString2);
    // очистить строку
    inputString2 = "";
    stringComplete2 = false;
    countstr2=0;
   }

}

void serialScanerEvent1() {
  //
  if (Serial3.available()>0) {
    // get the new byte:
    char inChar = (char)Serial1.read();
    // add it to the inputString:
    inputString3 += inChar;
    countstr1++;
```

```
    millisendstr1=millis();
  }
  else {
    if(millis()-millisendstr1>1000 && countstr1>0) {
       stringComplete1=true;
    }
  }
}

void serialNextionEvent2() {
  //
  if (Serial2.available()>0) {
    // get the new byte:
    char inChar = (char)Serial2.read();
    // add it to the inputString:
    inputString2 += String(inChar,HEX);
    countstr2++;
    inputString2 += " ";
    countstr2++;
    millisendstr2=millis();
  }
  else {
    if(millis()-millisendstr2>200 && countstr2>0) {
       stringComplete2=true;
    }
  }
}
```

Код установки количества продукта при нажатии кнопок показан в листинге 6.17.

Листинг 6.17

```
void parse_message(String msg)
  {
  //Serial.println(msg);
  // * сброс
  if (msg == "65 0 b 0 ff ff ff ")
    {
    counter=0;
    Serial2.print("n0.val=");
    Serial2.print(counter);
    Serial2.print("");
    Serial2.write(0xff);
    Serial2.write(0xff);
    Serial2.write(0xff);
    delay(10);
    }
```

```
// send
else if (msg == "65 0 c 0 ff ff ff ")
  {
  Serial2.print("t0.txt=\"send ...\"");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
  delay(10);
  // **** send data to server
  delay(1000);
  send_data_to_server();
  Serial2.flush();
  //
  Serial2.print("t0.txt=\"wait ...\"");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
  delay(10);
  counter=0;
  Serial2.print("n0.val=");
  Serial2.print(counter);
  Serial2.print("");
  Serial2.write(0xff);
  Serial2.write(0xff);
  Serial2.write(0xff);
  delay(10);
  }
// 0
else if (msg == "65 0 1 0 ff ff ff ")
  {
  add_count(0);
  delay(10);
  }
// 1
else if (msg == "65 0 2 0 ff ff ff ")
  {
  add_count(1);
  delay(10);
  }
// 2
else if (msg == "65 0 3 0 ff ff ff ")
  {
  add_count(2);
  delay(10);
  }
// 3
else if (msg == "65 0 4 0 ff ff ff ")
```

```
    {
    add_count(3);
    delay(10);
    }
  // 4
  else if (msg == "65 0 5 0 ff ff ff ")
    {
    add_count(4);
    delay(10);
    }
  // 5
  else if (msg == "65 0 6 0 ff ff ff ")
    {
    add_count(5);
    delay(10);
    }
  // 6
  else if (msg == "65 0 7 0 ff ff ff ")
    {
    add_count(6);
    delay(10);
    }
  // 7
  else if (msg == "65 0 8 0 ff ff ff ")
    {
    add_count(7);
    delay(10);
    }
  // 8
  else if (msg == "65 0 9 0 ff ff ff ")
    {
    add_count(8);
    delay(10);
    }
  // 9
  else if (msg == "65 0 a 0 ff ff ff ")
    {
    add_count(9);
    delay(10);
    }
  else ;
  }

void add_count(int n) {
  if(counter<100) {
     counter=counter*10+n;
     Serial2.print("n0.val=");
     Serial2.print(counter);
```

```
      Serial2.print("");
      Serial2.write(0xff);
      Serial2.write(0xff);
      Serial2.write(0xff);
      delay(10);
   }
}
```

Код отправки данных на сервер показан в листинге 6.18.

Листинг 6.18

```
void send_temp_to_server() {
   client.stop();
   if (client.connect(server, 80)) {
      //// sending data to the server
      // forming a string
      // uid per line
      String str="/firm/get_barcode.php?barcode=";
      str+=String(cardUID[i],HEX);
      str+="&count="+String(temp);
      Serial.print("str=");Serial.println(str);
      client.println("GET "+str+" HTTP/1.1");
      client.println("Host: ************.ru");
      client.println("User-Agent: ArduinoWiFi/1.1");
      client.println("Connection: close");
      client.println();
      Serial.println(response);
      delay(10); }
   else {
      // no connection
      Serial.println("connection failed");
   }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный скетч проекта можно найти в папке *projects\barcode\04* сопровождающего книгу электронного архива (см. *приложение*).

6.6.4. Получение данных на сервере с занесением в базу данных

Создайте на хостинге базу данных (MySQL) и две таблицы в ней:

- tovar — продукция и ее штрих-коды;
- count — результаты инвентаризации.

Структура таблицы tovar показана на рис. 6.56, а структура таблицы count — на рис. 6.57.

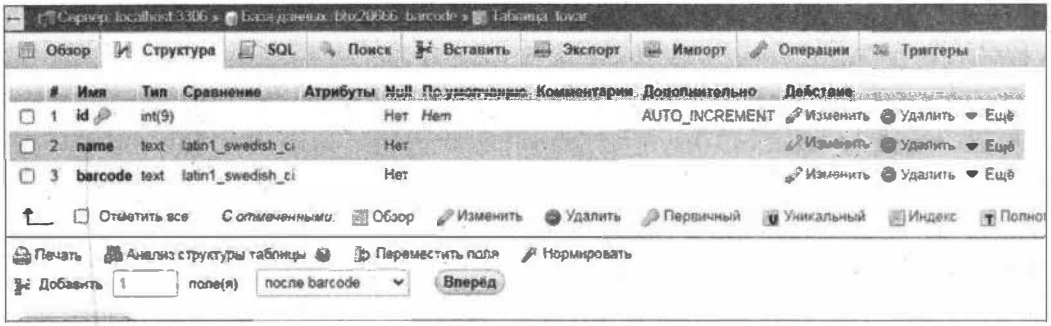


Рис. 6.56. Структура таблицы tovar

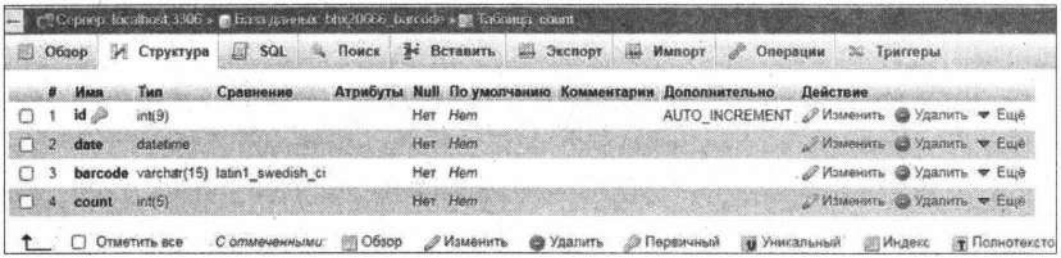


Рис. 6.57. Структура таблицы count

На сервере создайте PHP-скрипт get_barcode.php, который получает приходящие со сканера данные и заносит их в базу данных. Содержимое скрипта приведено в листинге 6.19.

Сканируем и проверяем приход и сохранение данных на сервере (рис. 6.58).

Листинг 6.19

```
<?php

//Параметры MySQL
$location="localhost";
$user="********";
$pass="***************";
$db_name="*****************";

// connect db
if(! $db=mysqli_connect($location,$user,$pass,$db_name))
 {echo "connect error";}
else
 {;}

$query1=" INSERT INTO count SET
          barcode='".$_GET['barcode']."',
          count='".$_GET['count']."',
          date='".date('Y-m-d H:i:s')."' ";
```

```
if(mysqli_query($db,$query1)){
   echo "#yes";
}
else {
   echo "#no";
}
?>
```

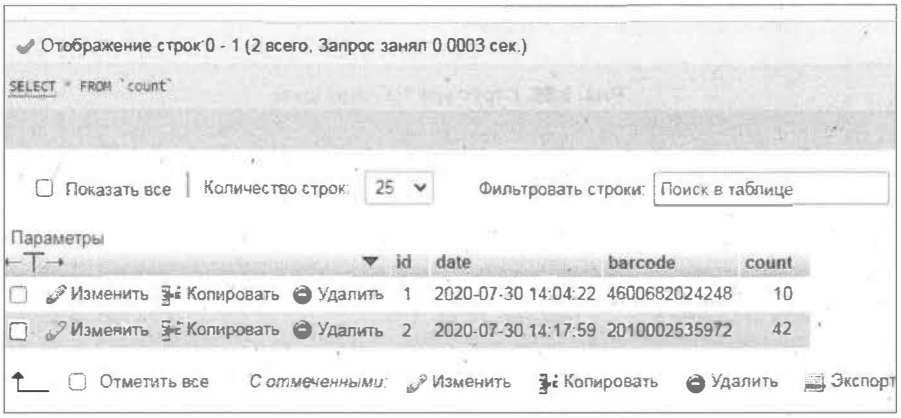


Рис. 6.58. Данные, приходящие со сканера в базе данных

ЭЛЕКТРОННЫЙ АРХИВ

Скрипт *get_barcode.php* проекта находится в папке *projects\barcode* сопровождающего книгу электронного архива (см. *приложение*).

6.7. Бесконтактное измерение температуры персонала с отправкой данных в облако LoRaWAN

Новые реалии — коронавирус COVID-19. В рамках профилактических мер по предотвращению заноса инфекции на предприятие (в организацию) работодателям рекомендуется организовать перед началом каждой рабочей смены «входной фильтр» с проведением бесконтактного измерения температуры тела сотрудников и обязательным отстранением от нахождения на рабочем месте лиц, у которых эта температура повышена.

В настоящее время на многих предприятиях этот процесс происходит следующим образом: медсестра на входе измеряет температуру тела проходящих мимо членов коллектива и вручную записывает в журнал полученные данные. Полное отсутствие автоматизации...

В связи с этим появляется проект-предложение по некоторой рационализации этого процесса: вход сотрудников по RFID-меткам, проверка метки в базе на сервере, бесконтактное измерение температуры и в зависимости от этой температуры подача сигнала на реле для допуска/недопуска сотрудника на предприятие с отправкой данных на сервер для сохранения в базе данных.

Для отправки данных на сервер мы воспользуемся LoRaWAN сетью The Things Network, а в качестве контроллера применим плату The Things Uno с подключенным к ней инфракрасным датчиком MLX90614 для бесконтактного измерения температуры.

6.7.1. Инфракрасный датчик MLX90614

Бесконтактное измерение температуры нам поможет проводить инфракрасный датчик MLX90614 версии DAA (рис. 6.59). Он способен измерять температуру в диапазоне от –70 до 380 градусов по Цельсию с точностью около 0,5°С. Датчик использует протокол I^2C. Назначение его контактов показано на рис. 6.60.

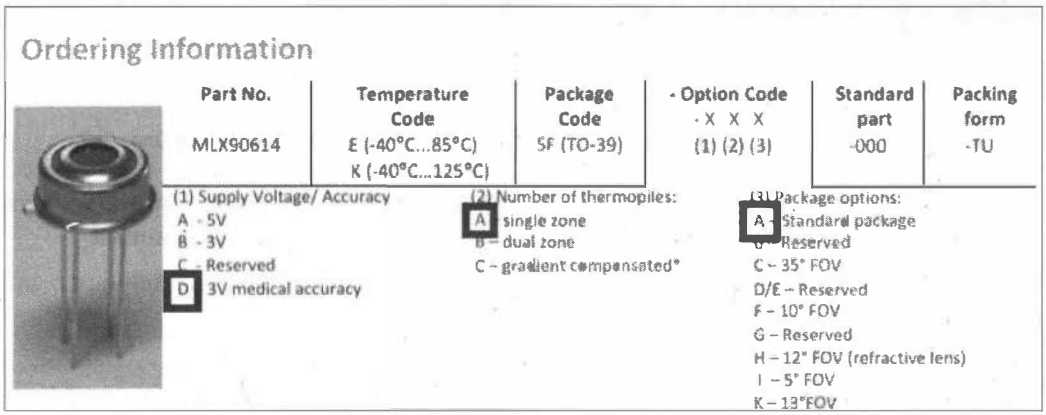


Рис. 6.59. Инфракрасный датчик температуры MLX90614DAA

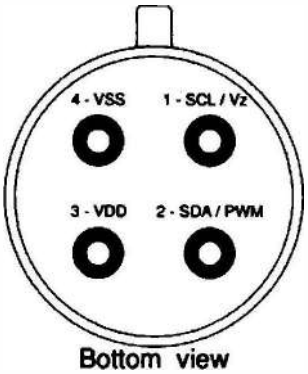


Рис. 6.60. Назначение контактов инфракрасного датчика температуры MLX90614DAA

Монтажная схема подключения инфракрасного датчика температуры к плате The Things Uno показана на рис. 6.61. Как отмечалось в *разд. 5.5.2*, плата The Things Uno базируется на микроконтроллере ATmega32u4, т. е. основана на платформе Arduino Leonardo, а не Arduino Uno, и подключения к ней выполняются так же.

Для программирования нам понадобится библиотека Adafruit_MLX90614.h, которую необходимо установить через Менеджер библиотек: выбираем в меню среды

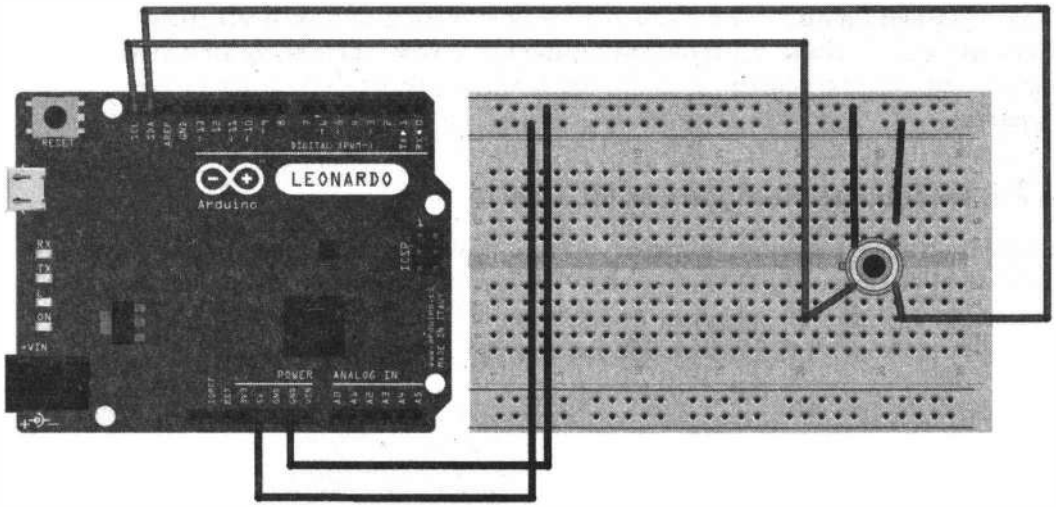


Рис. 6.61. Схема подключения датчика MLX90614DAA к плате The Things Uno

Arduino IDE пункт **Эскиз | Подключить библиотеку | Управлять библиотеками**, ищем библиотеку **Adafruit_MLX90614.h** и нажимаем на кнопку **Установка**.

Загрузите в плату The Things Uno скетч (листинг 6.20) для получения данных с датчика и отправки их в последовательный порт.

Листинг 6.20

```
// Подключение библиотеки
#include <Wire.h>
#include <Adafruit_MLX90614.h>
// Создание экземпляра объекта
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

void setup() {
   Serial.begin(9600);
   Serial.println("start");
   // запуск датчика
   mlx.begin();
}

void loop() {
   Serial.print("Ambient = "); Serial.print(mlx.readAmbientTempC());
   Serial.print("*C");
   Serial.print("\tObject = "); Serial.print(mlx.readObjectTempC());
   Serial.print("*C");
   Serial.print("\tObject = "); Serial.print(mlx.readObjectTempF());
   Serial.println("*F");
   delay(3000);
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.20, можно найти в папке *projects\Vorawan\01* сопровождающего книгу электронного архива (см. *приложение*).

Загрузив скетч в плату, откройте монитор последовательного порта. Температура окружающей среды и температура объекта выводятся на последовательный порт каждые 3 секунды (рис. 6.62).

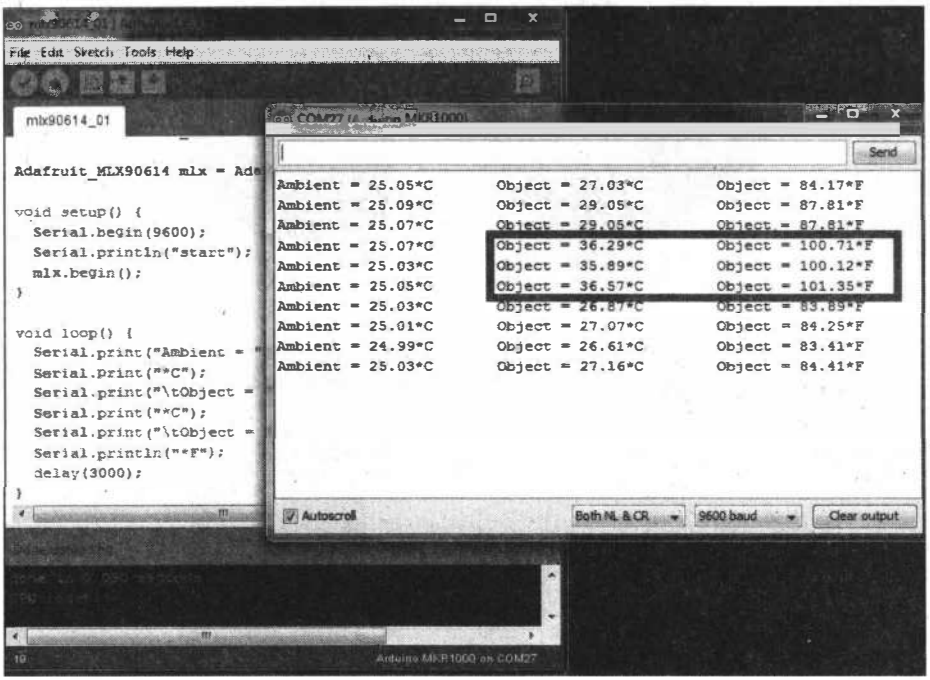


Рис. 6.62. Отправка данных датчика MLX90614 в последовательный порт

6.7.2. Подключение к плате The Things Uno модуля считывателя RFID RC522

Существует большое разнообразие меток RFID. Метки бывают *активными* (со встроенным источником питания) и *пассивными* (без встроенного источника питания — питаемые от тока, индуцированного в антенне сигналом от считывателя). Метки работают на разных частотах: LF (125–134 кГц), HF (13,56 МГц), UHF (860–960 МГц).

Устройства, которые считывают информацию из тегов и записывают в них данные, называются *считывателями*. В проектах Arduino в качестве считывателя очень часто используется модуль RFID-RC522 (рис. 6.63). Модуль выполнен на микросхеме MFRC522 компании NSP и обеспечивает работу с RFID-метками на частоте 13,56 МГц.

Подключение модуля RFID-считывателя RC522 к плате The Things Uno осуществляется по протоколу SPI. Монтажная схема подключения модуля RC522 и датчика

MLX90614DAA к плате The Things Uno показана на рис. 6.64, а вся схема в сборе — на рис. 6.65. Заметьте, что подключение по SPI плат Arduino Leonardo и Arduino Uno выполняется различно.

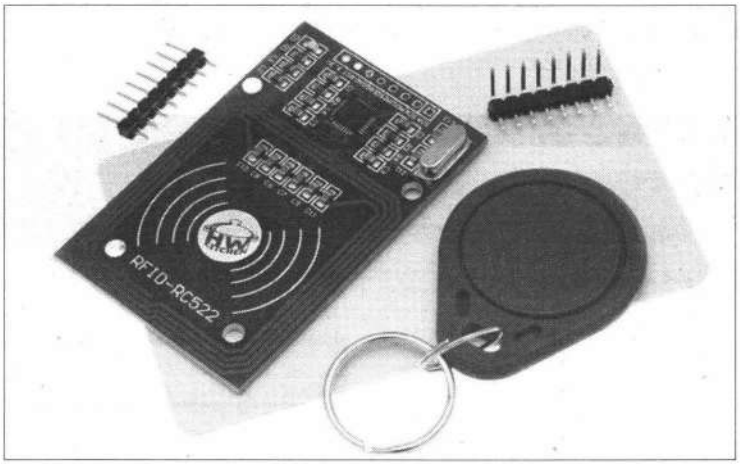


Рис. 6.63. Модуль RFID-RC522 и метка RFID

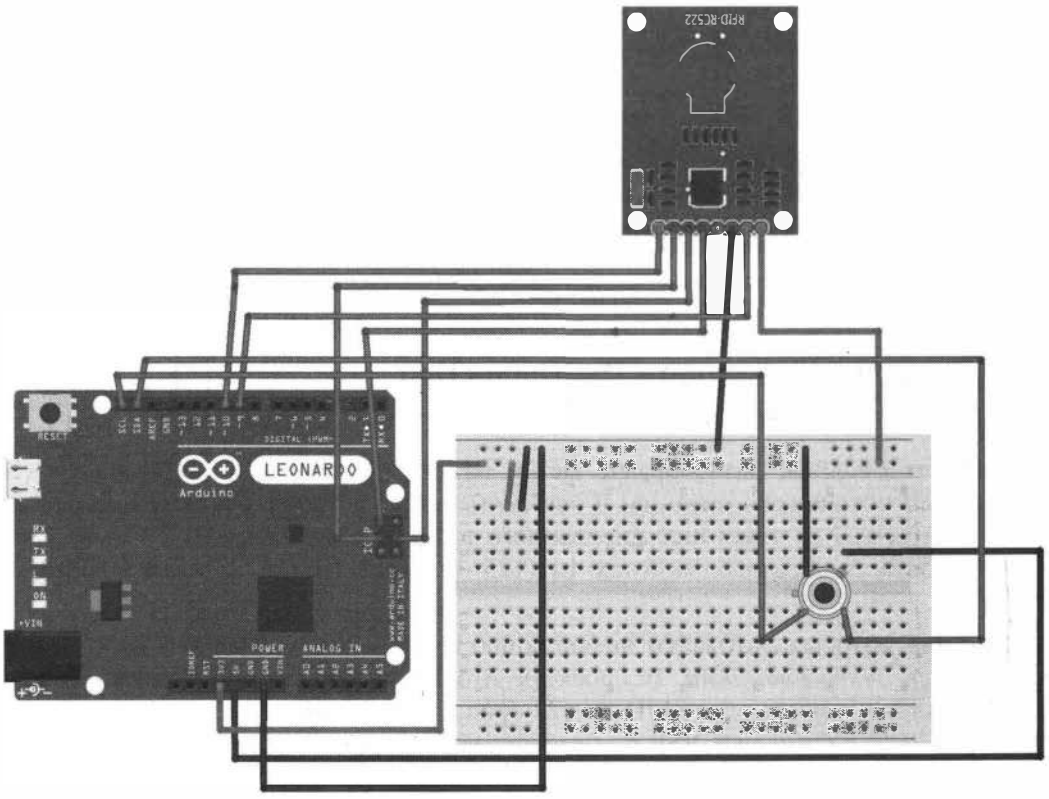


Рис. 6.64. Монтажная схема подключения датчика MLX90614DAA и модуля RFID-RC522 к плате The Things Uno

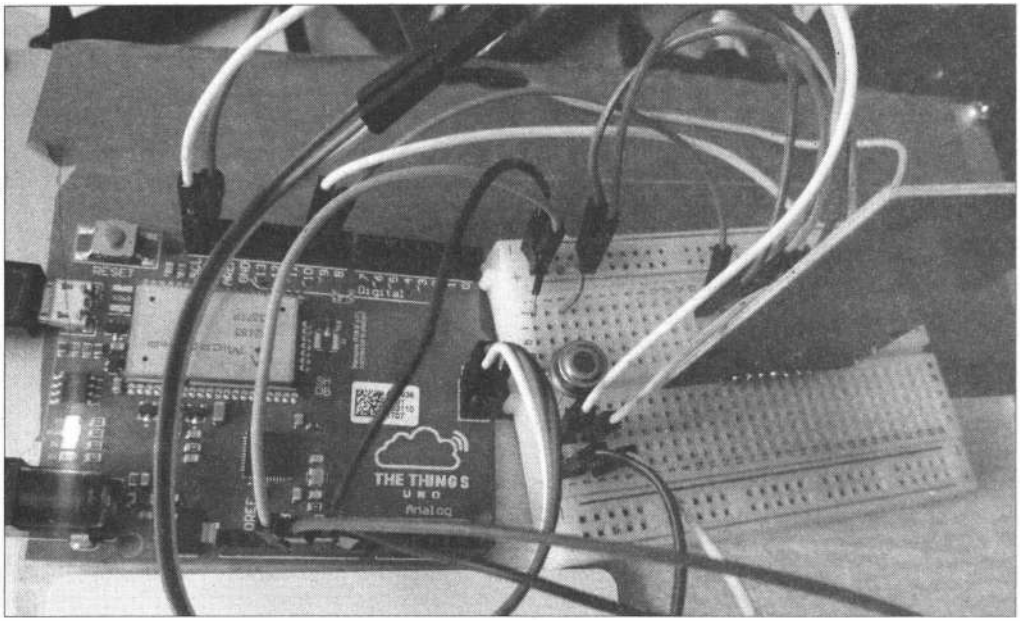


Рис. 6.65. Схема подключения датчика MLX90614DAA и модуля RFID-RC522 к плате The Things Uno в сборе

Напишем скетч, обеспечивающий процедуру измерения температуры ИК-датчиком по поднесению RFID-карты к считывателю (необходимо в течение 5 секунд после поднесения карты к считывателю поднести на пару секунд руку к датчику). Данные выводятся в последовательный порт. Содержимое скетча представлено в листинге 6.21.

Листинг 6.21

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>

MFRC522 mfrc522(10, 9);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

byte cardUID[4] = {0,0,0,0};
float mytemp=0.0;

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

```
  //  MFRC522
  mfrc522.PCD_Init();
  // запуск MLX90614
  mlx.begin();

}

void loop() {
  if ( mfrc522.PICC_IsNewCardPresent()) {
     //
     if ( mfrc522.PICC_ReadCardSerial()) {
        //  UID
        Serial.print(F("Card UID:"));
        for (byte i = 0; i < 4; i++) {
           cardUID[i]=mfrc522.uid.uidByte[i];
           Serial.print(cardUID[i],HEX);
        }
     Serial.println();
     // получить температуру
     get_temperature();
     Serial.print("t=");
     Serial.println(mytemp);
     delay(2000);
     }
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCrypto1();
  }

}
// измерение температуры
boolean get_temperature() {
  int count=0;
  float sumtemp=0.0;
  unsigned long millist=millis();

  do {
    float t=mlx.readObjectTempC();
    if(t>34.0 && t<42.0) {
       count++;
       sumtemp=sumtemp+t;
    }
  }while(millis()-millist<5000 || count<5);
  if(count<5) {
     mytemp=0.0;
     return false;
  }
```

```
  else {
     mytemp=sumtemp/count;
     return true;
  }
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.21, можно найти в папке *projects\Vorawan\02* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату The Things Uno и откройте монитор последовательного порта. При поднесении карты к считывателю и руки к датчику измеряется температура и данные выводятся в последовательный порт (рис. 6.66).

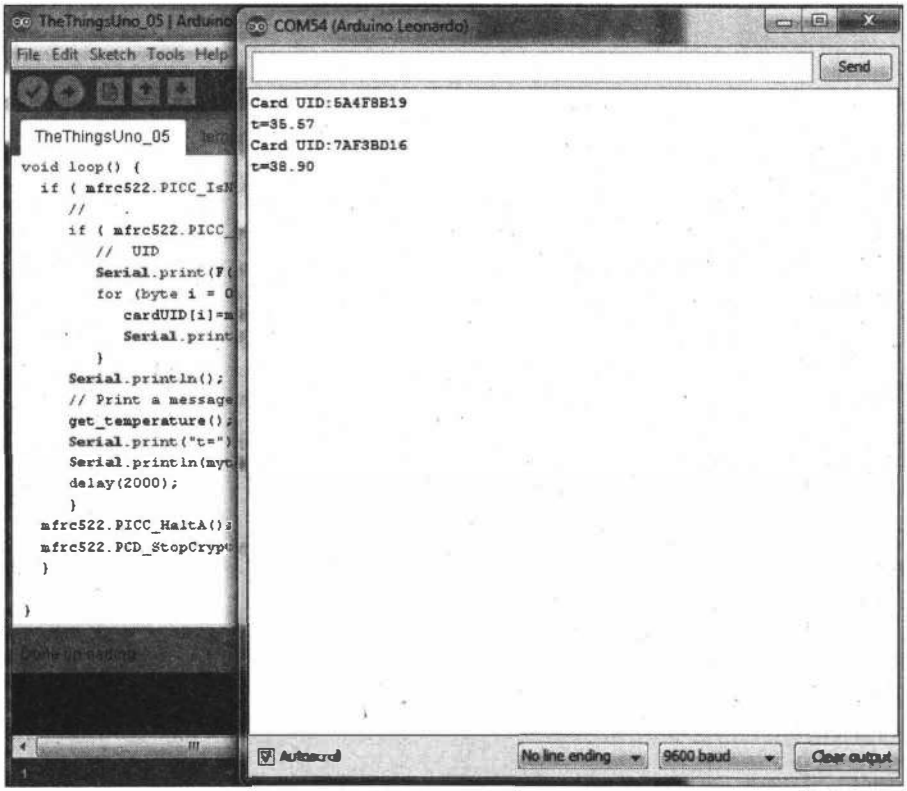


Рис. 6.66. Вывод данных измерения температуры в последовательный порт

6.7.3. Подключение к плате The Things Uno дисплея и реле

Теперь нам нужно подключить к плате The Things Uno дисплей и реле, которое выдаст команду на оборудование, позволяющую или нет сотруднику войти в помещение (при температуре тела выше 37,5 градуса сигнал на реле не открывает препятствие и сотрудник в помещение не допускается).

В качестве дисплея мы воспользуемся символьным LCD дисплеем WH1602 на интерфейсе I^2C, а в качестве реле — релейным модулем для Arduino на 5 В. Схема подключения дисплея и реле показана на рис. 6.67.
Для программирования дисплея нам понадобится библиотека LiquidCrystal_I2C. Содержимое скетча показано в листинге 6.22.

Листинг 6.22

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include <LiquidCrystal_I2C.h>

#define debugSerial Serial

MFRC522 mfrc522(10, 9);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
LiquidCrystal_I2C lcd(0x27,16,2);

byte cardUID[4] = {0,0,0,0};
float mytemp=0.0;
int pin_relay=12;

void setup() {
  //
  debugSerial.begin(9600);
  // display
  lcd.init();
  lcd.backlight();
  lcd.clear();
  //  SPI
  SPI.begin();
  //  MFRC522
  mfrc522.PCD_Init();
  // запуск MLX90614
  mlx.begin();
  // relay
  pinMode(pin_relay, OUTPUT);
  digitalWrite(pin_relay, LOW);
  // сообщение на LCD.
  lcd.setCursor(0,0);
  lcd.print("Wait card....");
}
```

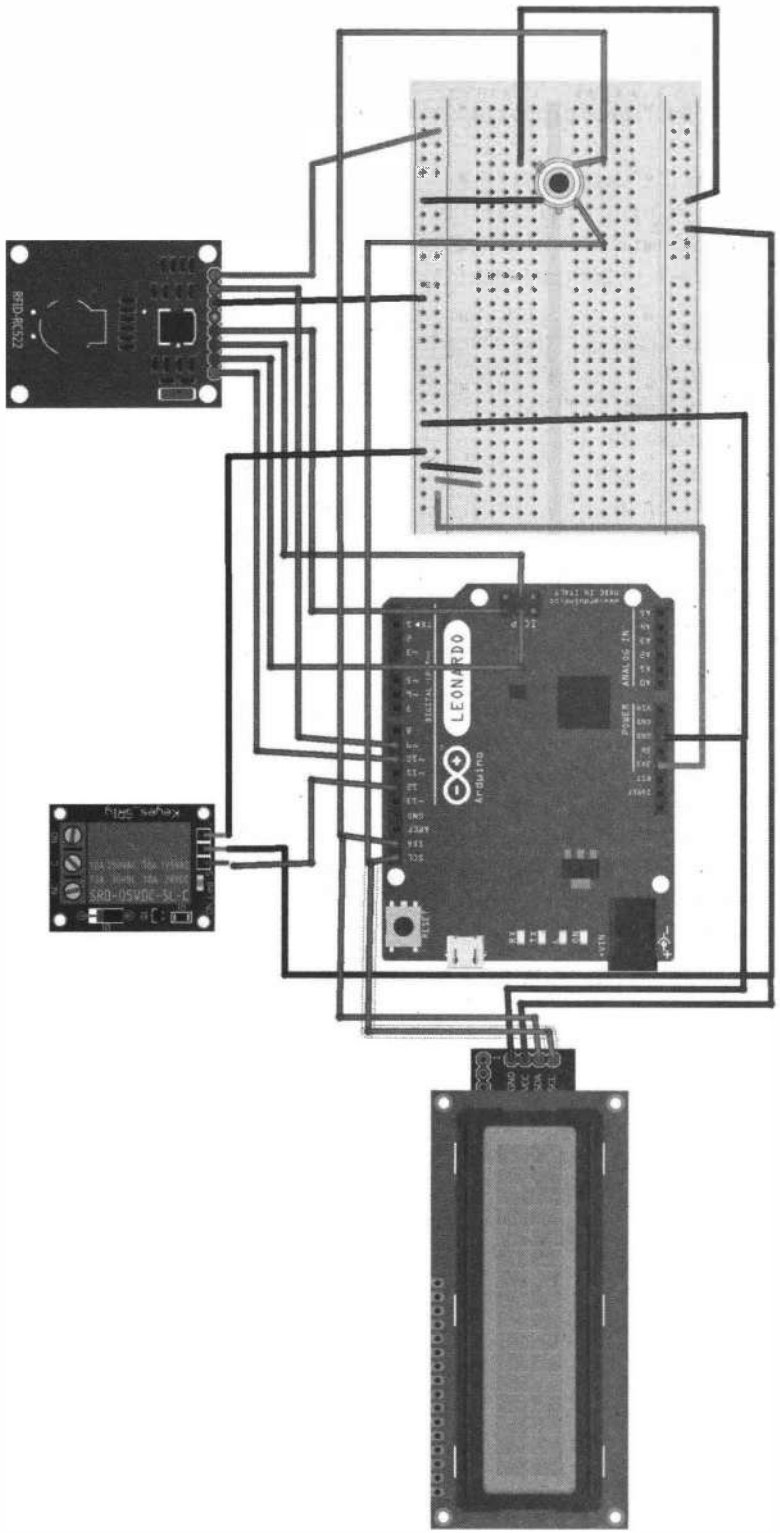


Рис. 6.67. Подключение к плате The Things Uno дисплея WH1602 I2C и релейного модуля

```
void loop() {
  if ( mfrc522.PICC_IsNewCardPresent()) {
     //    .
     if ( mfrc522.PICC_ReadCardSerial()) {
        //  UID
        Serial.print(F("Card UID:"));
        for (byte i = 0; i < 4; i++) {
           cardUID[i]=mfrc522.uid.uidByte[i];
           debugSerial.print(cardUID[i],HEX);
        }
     // сообщение на LCD.
     lcd.setCursor(0,0);
     lcd.print("Gett temp ..");
     if(get_temperature()) {
       //send_ttn(mytemp);
        if(mytemp>37.5) {
           // close
           digitalWrite(pin_relay, LOW);
           lcd.setCursor(0,0);
           lcd.print("CLOSE !!!!!!! ..");
           delay(2000);
        }
        else {
           // open
           digitalWrite(pin_relay, HIGH);
           lcd.setCursor(0,0);
           lcd.print("OPEN !!!!!!! ..");
           delay(5000);
           digitalWrite(pin_relay, LOW);
        }
     }
     debugSerial.print("t=");
     debugSerial.println(mytemp);
     // сообщение на LCD.
     lcd.setCursor(0,0);
     lcd.clear();
     lcd.print("Wait card....");
     delay(2000);
     }
  mfrc522.PICC_HaltA();
  mfrc522.PCD_StopCryptol();
  }

}

boolean get_temperature() {
  int count=0;
```

```
  float sumtemp=0.0;
  unsigned long millist=millis();

  do {
    float t=mlx.readObjectTempC();
    lcd.setCursor(1,0);
    lcd.print("        ");
    lcd.print(t);
    if(t>34.0 && t<42.0) {
       count++;
       sumtemp=sumtemp+t;
    }
  }while(millis()-millist<5000 || count<5);
  if(count<5) {
     mytemp=0.0;
     return false;
  }
  else {
     mytemp=sumtemp/count;
     return true;
  }
}
```

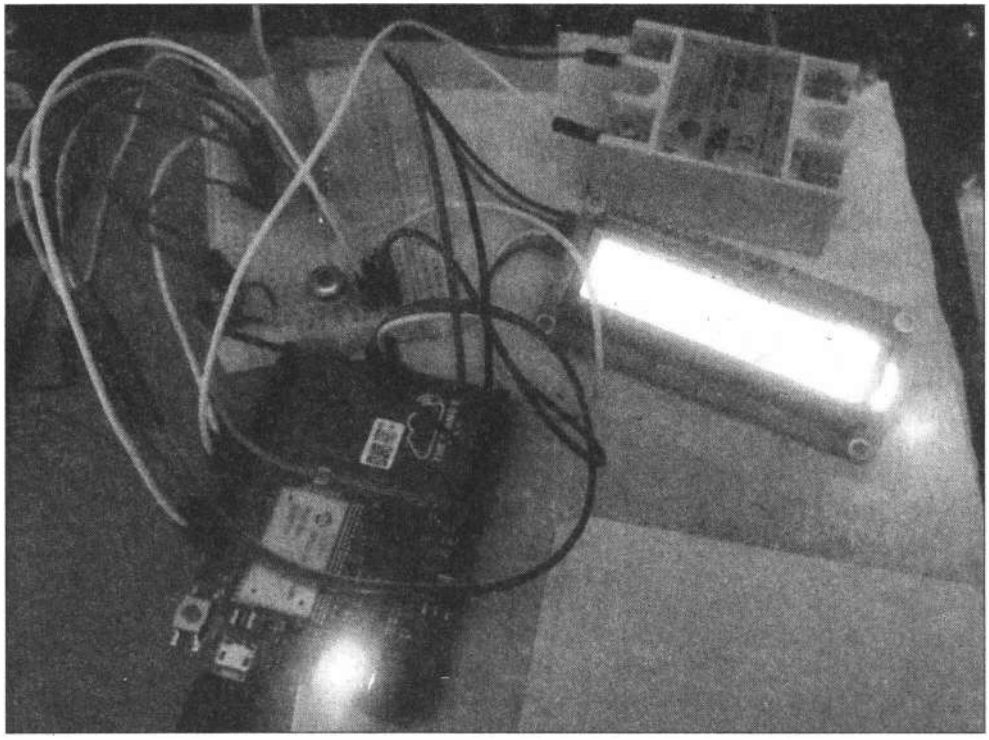


Рис. 6.68. Схема подключения к плате The Things Uno датчика MLX90614DAA, модуля RFID-RC522, дисплея и реле в сборе

ЭЛЕКТРОННЫЙ АРХИВ

Скетч, соответствующий листингу 6.22, можно найти в папке *projects\lorawan\03* сопровождающего книгу электронного архива (см. *приложение*).

ЭЛЕКТРОННЫЙ АРХИВ

Библиотека LiquidCrystal_I2C размещена в папке *libraries* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату The Things Uno, и на дисплее появятся предусмотренные скетчем сообщения (рис. 6.68).

6.7.4. Отправка данных из платы The Things Uno по сети LoRaWAN в службу The Things Network и перенаправление их на сторонний сервер

Для отправки данных по сети LoRaWAN импортируем в скетч библиотеку TheThingsNetwork.h и введем данные из нашего приложения в сервисе The Things Network, которое необходимо предварительно создать:

```
#include <TheThingsNetwork.h>
// Set your AppEUI and AppKey
const char *appEui = "70B3D57ED00313E5";
const char *appKey = "FAFA472FB54FFE967CE63F5EE056F6D7";
#define loraSerial Serial1
#define debugSerial Serial
// Replace REPLACE_ME with TTN_FP_EU868 or TTN_FP_US915
#define freqPlan TTN_FP_EU868
TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);
```

Создание приложения в сервисе The Things Network мы рассматривали в *разд. 5.5.2*. Скетч с отправкой неформатированных данных в сервис The Things Network представлен в листинге 6.23.

Листинг 6.23

```
// Подключение библиотек
#include <SPI.h>
#include <MFRC522.h>
#include <TheThingsNetwork.h>
#include <Wire.h>
#include <Adafruit_MLX90614.h>
#include <LiquidCrystal_I2C.h>

// Данные нашего приложения AppEUI и AppKey
const char *appEui = "70B3D57ED00313E5";
const char *appKey = "FAFA472FB54FFE967CE63F5EE056F6D7";

#define loraSerial Serial1
#define debugSerial Serial
```

```
// Установить частоту TTN_FP_EU868 or TTN_FP_US915
#define freqPlan TTN_FP_EU868

TheThingsNetwork ttn(loraSerial, debugSerial, freqPlan);
MFRC522 mfrc522(10, 9);
Adafruit_MLX90614 mlx = Adafruit_MLX90614();
LiquidCrystal_I2C lcd(0x27,16,2);

byte cardUID[4] = {0,0,0,0};
byte payload[6] = {0,0,0,0,0,0};
float mytemp=0.0;
int pin_relay=12;

void setup() {
  //
  loraSerial.begin(57600);
  debugSerial.begin(9600);
  // display
  lcd.init();
  lcd.backlight();
  lcd.clear();
  //  SPI
  SPI.begin();
  //  MFRC522
  mfrc522.PCD_Init();
  // запуск MLX90614
  mlx.begin();
  // реле
  pinMode(pin_relay, OUTPUT);
  digitalWrite(pin_relay, LOW);
  // ожидание запуска Serial Monitor
  while (!debugSerial && millis() < 10000)
    ;
  debugSerial.println("-- STATUS");
  ttn.showStatus();

  debugSerial.println("-- JOIN");
  ttn.join(appEui, appKey);
  // Print a message to the LCD.
  lcd.setCursor(0,0);
  lcd.print("Wait card....");
}

void loop() {
  if ( mfrc522.PICC_IsNewCardPresent()) {
     //    .
     if ( mfrc522.PICC_ReadCardSerial()) {
```

```
      //  UID
      Serial.print(F("Card UID:"));
      for (byte i = 0; i < 4; i++) {
         cardUID[i]=mfrc522.uid.uidByte[i];
         debugSerial.print(cardUID[i],HEX);
      }
   debugSerial.println();
   debugSerial.println("-- LOOP");
   // В массив отправляемых данных - RFID
   payload[0] = cardUID[0];
   payload[1] = cardUID[1];
   payload[2] = cardUID[2];
   payload[3] = cardUID[3];
   // сообщение на LCD.
   lcd.setCursor(0,0);
   lcd.print("Gett temp ..");
   if(get_temperature()) {
      send_ttn(mytemp);
      if(mytemp>37.5) {
         // close вход
         digitalWrite(pin_relay, LOW);
         lcd.setCursor(0,0);
         lcd.print("CLOSE !!!!!!! ..");
         delay(2000);
      }
      else {
         // open вход
         digitalWrite(pin_relay, HIGH);
         lcd.setCursor(0,0);
         lcd.print("OPEN !!!!!!! ..");
         delay(5000);
         digitalWrite(pin_relay, LOW);
      }
   }
   debugSerial.print("t=");
   debugSerial.println(mytemp);
   // Print a message to the LCD.
   lcd.setCursor(0,0);
   lcd.clear();
   lcd.print("Wait card....");
   delay(2000);
   }
 mfrc522.PICC_HaltA();
 mfrc522.PCD_StopCrypto1();
 }
}
```

Электронный архив

Скетч, соответствующий листингу 6.23, можно найти в папке *projects\lorawan\04* сопровождающего книгу электронного архива (см. *приложение*).

В консоли The Thing Network (рис. 6.69) для преобразования неформатированных данных вводим функцию декодирования входящих данных (листинг 6.24) и можем видеть на вкладке **Data** входящие данные в формате JSON (рис. 6.70).

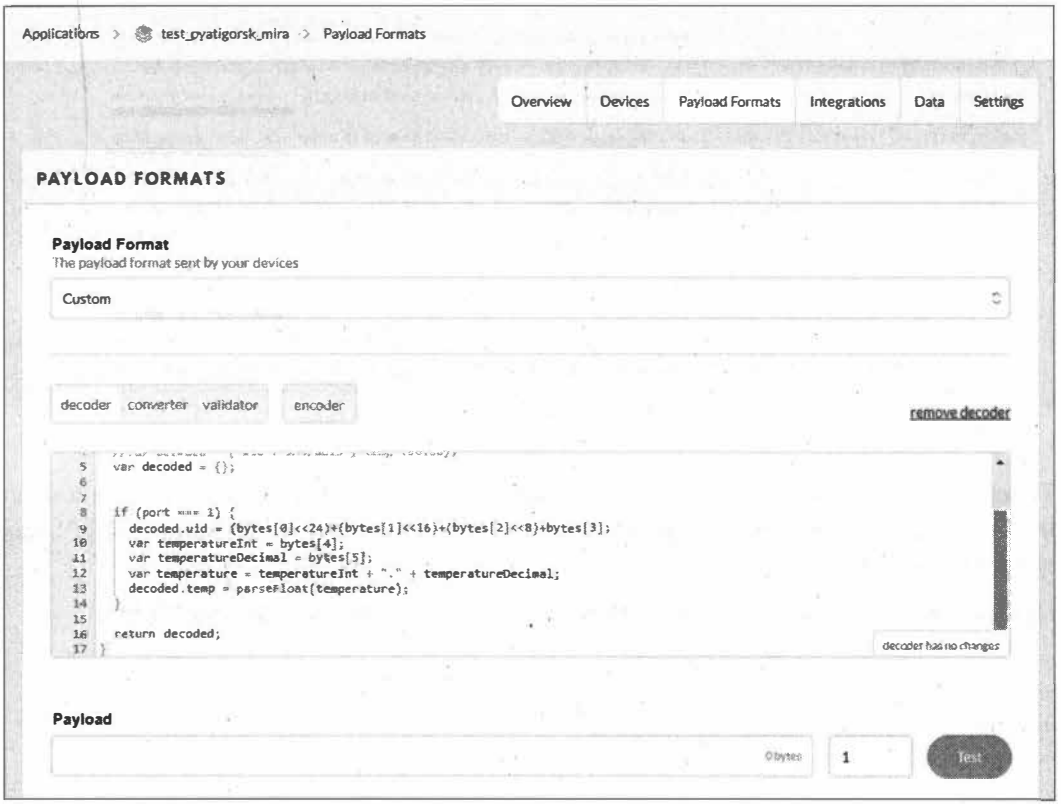


Рис. 6.69. Функция декодирования входящих данных в консоли The Thing Network

Листинг 6.24

```
function Decoder(bytes, port) {
  var decoded = {};
  if (port === 1) {
    decoded.uid = (bytes[0]<<24)+(bytes[1]<<16)+(bytes[2]<<8)+bytes[3];
    var temperatureInt = bytes[4];
    var temperatureDecimal = bytes[5];
    var temperature = temperatureInt + "." + temperatureDecimal;
    decoded.temp = parseFloat(temperature);
  }
return decoded;
}
```

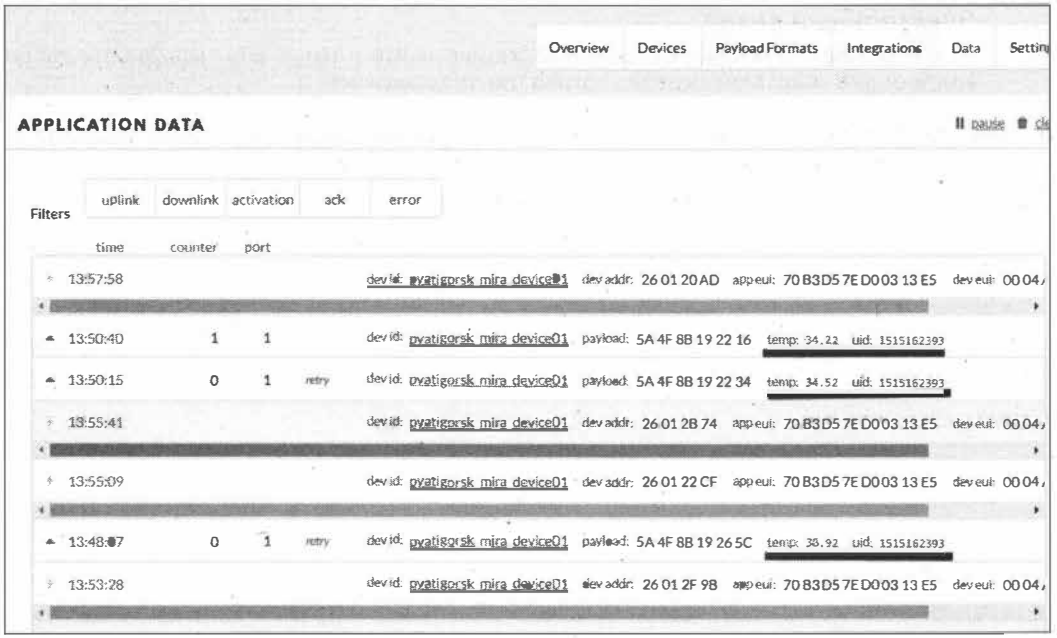


Рис. 6.70. Входящие данные на вкладке Data

Теперь перенаправим данные, отправленные в службу The Things Network, на нужный нам ресурс через HTTP с помощью опции Integrations (интеграция HTTP), как показано на рис. 6.71.

Пример перенаправленных на другой сервер данных при интеграции HTTP:

```
{"app_id":"test_pyatigorsk_mira",
"dev_id":"pyatigorsk_mira_device01",
"hardware_serial":"0004A30B001C3FB6",
"port":1,
"counter":0,
"payload_raw":"Wk+LGSIx",
"payload_fields":{"temp":34.49,"uid":1515162393},
"metadata":{
"time":"2020-07-06T11:48:22.5344963362",
"frequency":867.7,
"modulation":"LORA",
"data_rate":"SF7BW125",
"coding_rate":"4/5",
"gateways":[{
"gtw_id":"eui-58a0cbfffe80125c",
"timestamp":681777324,
"time":"2020-07-06T11:48:22.4334089752",
"channel":0,
"rssi":-42,
```

```
"snr":9.25,
"rf_chain":0}]},
"downlink_url":"https://integrations.thethingsnetwork.org/ttn-eu/api/v2/
down/test_pyatigorsk_mira/test01?key=ttn-account-v2.iGsvHMLzmrCXJ1p6D9u2Tm1x-
yoHo3y0feranDWkqGg"}
```

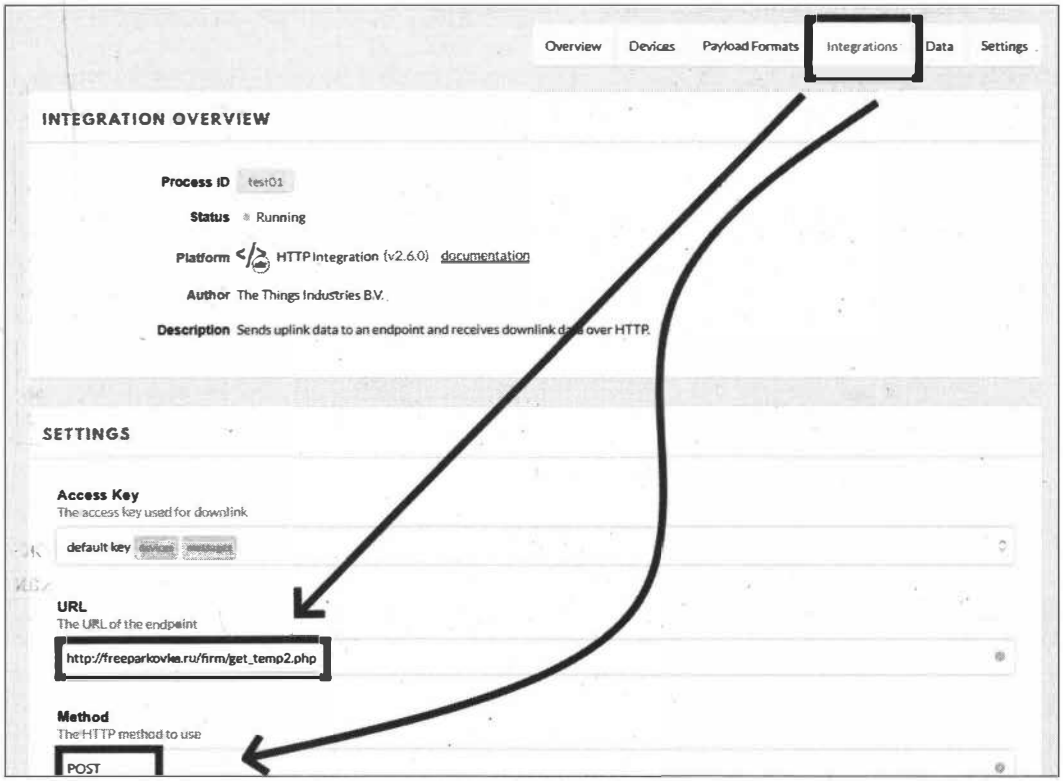



Рис. 6.71. Интеграция HTTP — перенаправление данных на другой адрес

6.7.5. Создание базы данных сотрудников на сайте компании для сбора ежедневных показаний температуры на входе

Создайте на хостинге базу данных (MySQL) и две таблицы в ней:

- users — данные о сотрудниках и их UID;
- temp — данные о температуре тела, измеренной на входе.

Структура таблицы users показана на рис. 6.72, структура таблицы temp — на рис. 6.73.

На сервере создайте PHP-скрипт get_temp2.php, который получает отправляемые данные и заносит их в базу данных. Содержимое скрипта приведено в листинге 6.25.

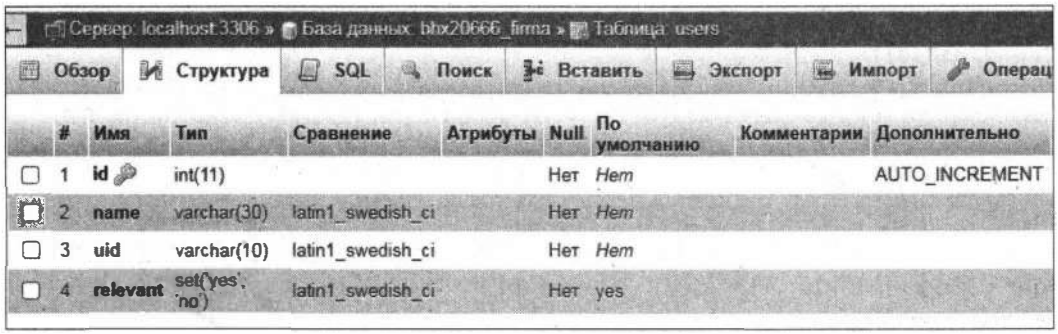


Рис. 6.72. Структура таблицы users

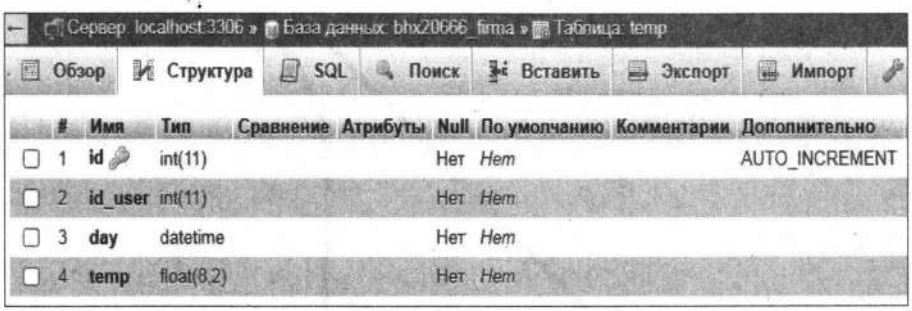


Рис. 6.73. Структура таблицы temp

Листинг 6.25

```
<?php
//
$location="localhost";
$user="************";
$pass="************";
$db_name="************";
// connect db
if(! $db=mysqli_connect($location,$user,$pass,$db_name))
 {echo "connect error";}
else
 {;}

        $contentType = isset($_SERVER["CONTENT_TYPE"]) ?
        trim($_SERVER["CONTENT_TYPE"]) : '';
        if(strcasecmp($contentType, 'application/json') != 0){
            echo 'Content type must be: application/json';
        }

        // получить the RAW post data.
        $content = trim(file_get_contents("php://input"));
```

```
        // записать данные в файл
        file_put_contents('json', $content . PHP_EOL, FILE_APPEND);

        // декодировать входящие необработанные данные post из JSON
        $decoded = json_decode($content, true);

$filed = "json";

$query0=" SELECT * FROM users WHERE
uid='".dechex($decoded['payload_fields']['uid'])."' ";
$rez0=mysqli_query($db,$query0);

if(mysqli_num_rows($rez0)>0) {
    $row0=mysqli_fetch_assoc($rez0);
    $query1=" INSERT INTO temp SET
          id_user='".$row0['id']."',
          temp='".$decoded[payload_fields]['temp']."',
          day='".date('Y-m-d H:i:s')."' ";
    mysqli_query($db,$query1);
    echo "#yes";
}
else {
    echo "#no";
}
?>
```

ЭЛЕКТРОННЫЙ АРХИВ

Скрипт *get_temp2.php* можно найти в папке *projects\lorawan* сопровождающего книгу электронного архива (см. *приложение*).

6.7.6. Страница для удаленного просмотра данных о температуре тела сотрудников

Чтобы просмотреть данные, отправленные на сервер, создадим страницу сайта. Код такой страницы (содержимое файла view_temp.php) приведен в листинге 6.26. Вид страницы с данными показан на рис. 6.74.

Листинг 6.26

```
<?php
//
$location="localhost";
$user="********";
$pass="********";
$db_name="********";
```

```
// connect db
if(! $db=mysqli_connect($location,$user,$pass,$db_name))
{echo "connect error";}
else
{;}
$query0=" SELECT * FROM temp WHERE uid='".$_GET['uid']."' ";
$rez0=mysqli_query($db,$query0);
$content1.=date('Y-m-d')."<br><br>";
$content1.="<table>";
$query1="SELECT * FROM temp WHERE day >= CURDATE() ";
$rez1=mysqli_query($db,$query1); $i=1;
while($row1=mysqli_fetch_assoc($rez1)) {
   $content1.="<tr>";
   $content1.="<td>".$i."</td>";
   $rez2=mysqli_query($db,"SELECT name FROM users
       WHERE id =".$row1['id_user']." ");
   $row2=mysqli_fetch_assoc($rez2);
   $content1.="<td>".$row2['name']."</td>";
   $content1.="<td>".mb_substr($row1['day'],10,9)."</td>";
   if($row1['temp']>37.5)
      $content1.="<td style='background-color:red'>
      ".$row1['temp']."</td>";
   else
      $content1.="<td>".$row1['temp']."</td>";
   $content1.="</tr>";
   $i++;
}
$content1.="</table>";
echo $content1;
?>
```

Электронный архив

Файл *view_temp2.php* можно найти в папке *projectsVorawan* сопровождающего книгу электронного архива (см. *приложение*).

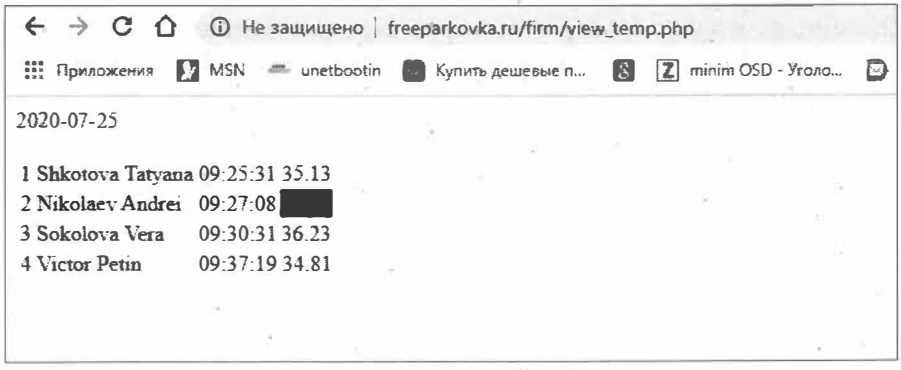


Рис. 6.74. Страница с данными о температуре тела сотрудников

6.8. «Умная гантеля» для автоматического подсчета числа упражнений с использованием TinyML и Edge Impulse

Рассмотрим, как построить систему обнаружения жестов для носимых устройств на плате Arduino Nano 33 BLE Sense с использованием TinyML и Edge Impulse.

ML (Machine Learning, машинное обучение) — это тренировка математической модели на некоторых данных, для того чтобы прогнозировать какое-то событие или явление на новых данных. То есть это попытка заставить алгоритмы программ совершать действия на основе предыдущего опыта, а не только на основе имеющихся данных.

TinyML — это машинное обучение для микроконтроллеров, имеющих, по сравнению с компьютерами, скромные вычислительные возможности. TinyML является развивающейся областью, в которой предстоит еще многое сделать. Но это даст возможность использовать ML в миллиардах микроконтроллеров в сочетании со всевозможными датчиками, что может существенно изменить мир встраиваемых решений.

Для обучения нужны реальные данные (обучающая выборка) и значение целевой переменной, которое соответствует заданным данным. Модель наблюдает и находит зависимости между данными и целевой переменной. Эти зависимости используются моделью для нового набора данных, чтобы прогнозировать целевую переменную, которая неизвестна.

Машинное обучение разделяется на три основных вида:

- с учителем (Supervised machine learning);
- без учителя (Unsupervised machine learning);
- глубокое обучение (Deep learning).

Глубокое обучение подразумевает под собой анализ Big Data — большого массива информации и использует для работы нейронные сети, а это требует больших вычислительных мощностей. Мы же в этом проекте воспользуемся онлайн-плат. формой Edge Impulse TinyML для сбора данных, обучения модели и ее развертывания на устройстве.

Итак, создадим проект «Умная гантеля», добавив к обычной гантеле микроконтроллер, который будет автоматически подсчитывать количество выполняемых посетителем спортзала различных упражнений и отправлять эти данные на смартфон. Наша гантеля будет вести подсчет для трех упражнений (рис. 6.75).

В качестве микроконтроллера мы применим плату Arduino Nano 33 BLE Sense, на которой распаян IMU-модуль на 9 степеней свободы, содержащий трехосевые сенсоры: акселерометр, гироскоп и магнитометр. Эту плату необходимо закрепить на гантеле. Для питания микроконтроллера можно использовать две батареи 18650.

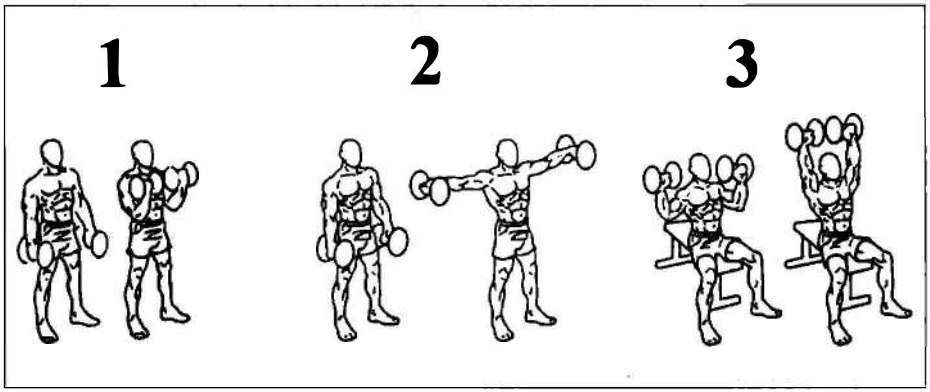


Рис. 6.75. Упражнения для «умной гантели»

6.8.1. Установка программного обеспечения

Как уже отмечалось ранее, мы в этом проекте для сбора данных, обучения модели и ее развертывания на устройстве воспользуемся онлайн-платформой Edge Impulse TinyML (**https://www.edgeimpulse.com**).

Прежде всего необходимо создать в ней учетную запись, перейдя по адресу **https://studio.edgeimpulse.com/signup**. Создав учетную запись, войдите в свой профиль и создайте новый проект (рис. 6.76).

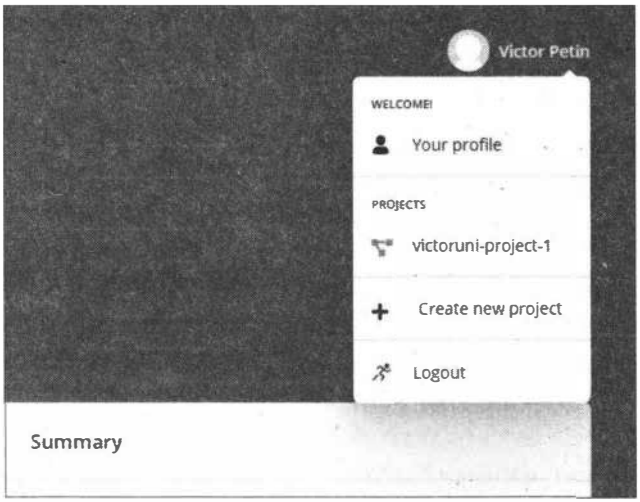


Рис. 6.76. Создание проекта в Edge Impulse

Платформа поддерживает несколько устройств, одно из которых — плата Arduino Nano 33 BLE Sense. Количество устройств со временем увеличивается; кроме того, можно самостоятельно портировать другие платформы.

Чтобы настроить Arduino Nano 33 BLE Sense для работы в Edge Impulse, необходимо на компьютере установить следующее программное обеспечение:

- Node.js v12 или выше;
- Arduino CLI — интерфейс командной строки для Arduino;
- Edge Impuls CLI.

Установщик Arduino CLI можно скачать на странице: **https://arduino.github.io/arduino-cli/latest/installation** (рис. 6.77).

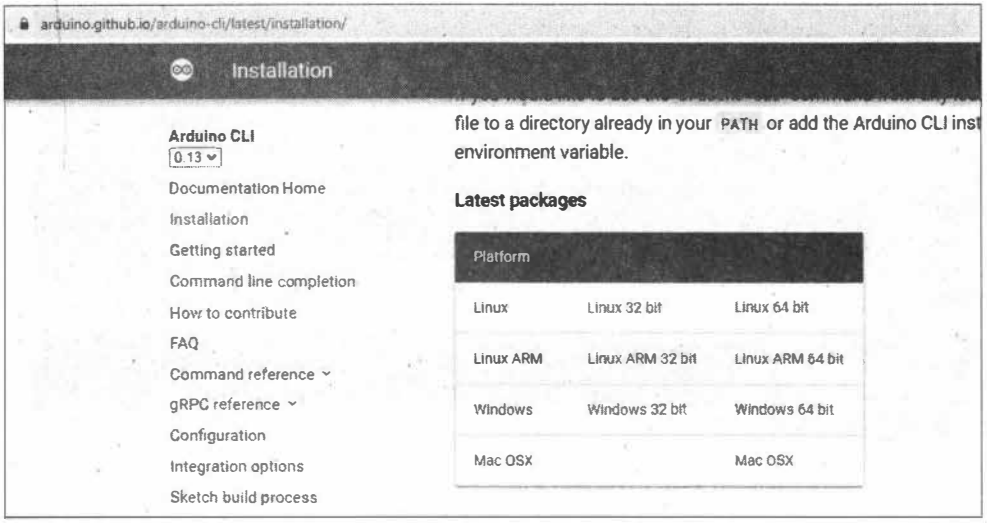


Рис. 6.77. Установка Arduino CLI

Установив Arduino CLI, далее работаем в командной строке (через cmd.exe) и устанавливаем Edge Impuls CLI (рис. 6.78), предварительно перейдя в папку, где находится командный файл arduino-cli:

```
npm install -g edge-impulse-cli @serialport/terminal
```

На плате пока нет нужной прошивки, поэтому загрузите на компьютер последнюю версию прошивки Edge Impulse (**https://cdn.edgeimpulse.com/firmware/arduino-nano-33-ble-sense.zip**) и разархивируйте полученный файл. Подключите плату Arduino Nano 33 BLE Sense к компьютеру с помощью кабеля micro-USB. Дважды нажмите на плате кнопку RESET, чтобы запустить загрузчик, — индикатор на ней должен начать пульсировать. Из командной строки запустите скрипт для своей операционной системы (flash_windows.bat для Windows), как показано на рис. 6.79.

Подождите, пока мигание индикатора не завершится, и нажмите кнопку RESET один раз, чтобы запустить новую прошивку.

Наберите в командной строке

```
edge-impulse-daemon
```

Это запустит мастер, который попросит вас войти в систему и выбрать проект Edge Impulse (рис. 6.80). Теперь устройство подключено к Edge Impulse. Чтобы убедиться в этом, перейдите в проект Edge Impulse и выберите **Devices** (рис. 6.81).

Рис. 6.78. Установка Edge Impuls CLI

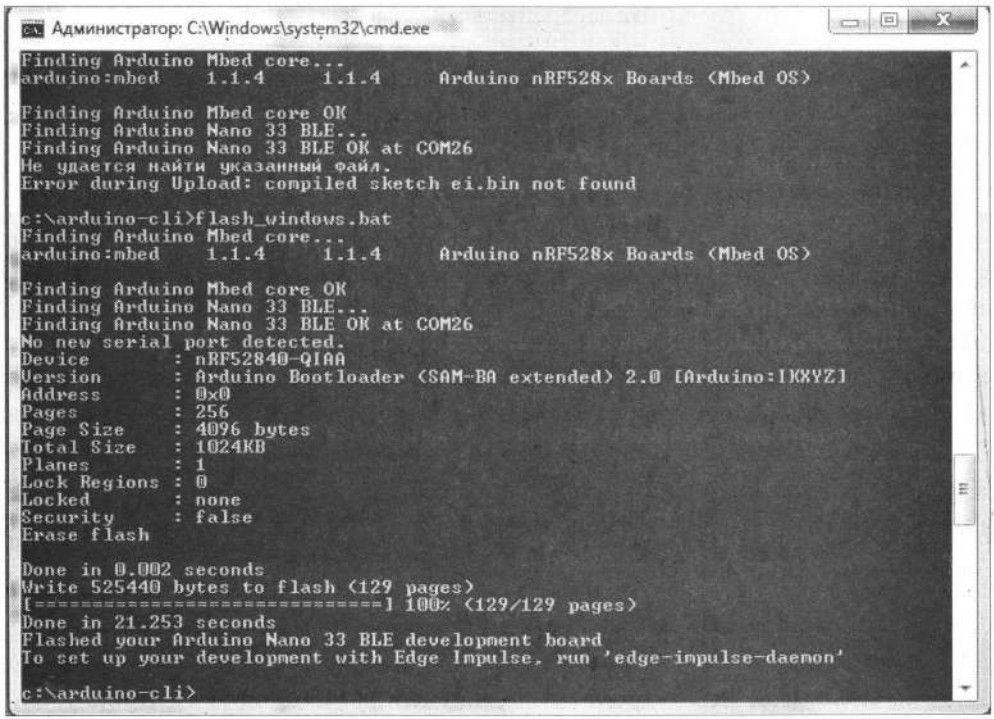


Рис. 6.79. Загрузка прошивки в плату Arduino Nano 33 BLE Sense

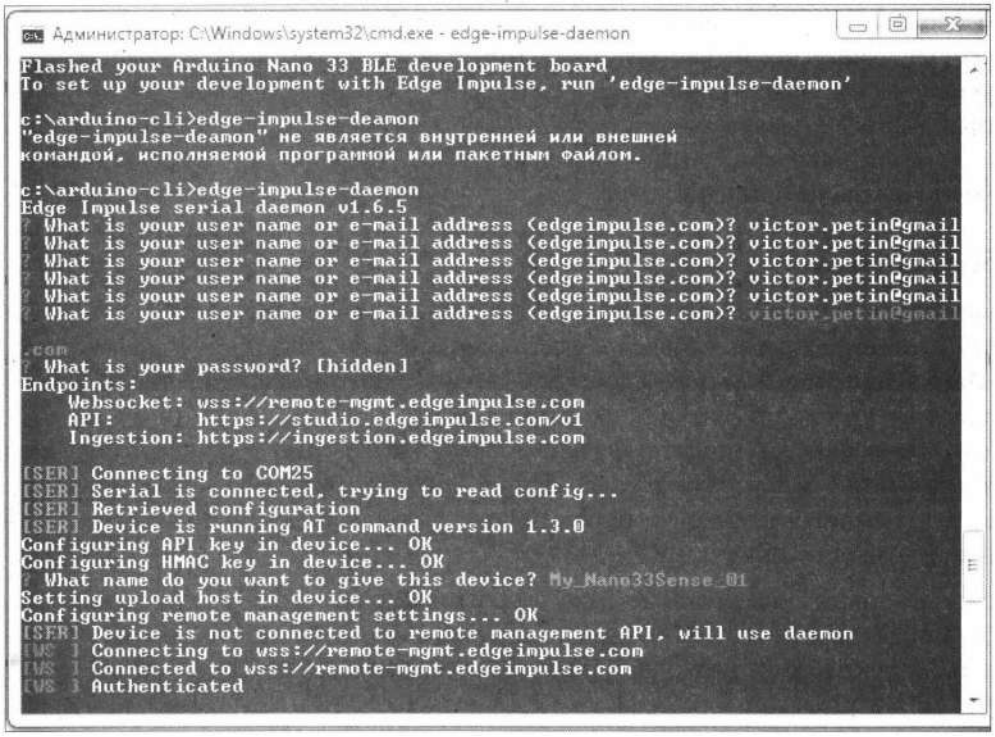


Рис. 6.80. Подключение платы к проекту Edge Impulse

Рис. 6.81. Проверка подключения платы к проекту в Edge Impulse

6.8.2. Сбор данных, тренировка и создание модели ML на платформе Edge Impulse

Теперь, когда все настроено, вы можете создать свою модель машинного обучения для системы распознавания упражнений, которая работает на микроконтроллере.

Это сложная задача, решаемая с помощью программирования на основе правил, поскольку люди не выполняют движения одинаково каждый раз. Но машинное обучение может легко справиться с этими изменениями. Мы получим высокочастотные данные с реальных датчиков (IMU-модуля), используем обработку сигналов для очистки данных, создадим классификатор нейронной сети и развернем полученную модель на устройстве.

Сбор данных

Подключите плату Arduino Nano 33 BLE Sense, подключенную к компьютеру, к API удаленного управления. Для этого перейдите на сайте Edge Impulse на вкладку **Data asquisition.** Мы будем собирать данные для четырех разных вариантов (упражнений): bicepscurl, lateralraise, overheadpress и idle ((бездействие). В разделе **Record new data** установите значения длительности для фрагмента получаемых данных и метку жеста. Нажмите на кнопку **Start sampling** и во время записи фрагмента постоянно выполняйте определенное упражнение. При этом важно делать непрерывные движения.

Запишите таким образом несколько фрагментов для каждого упражнения. Необработанные данные (RAW DATA) каждого фрагмента можно просмотреть: на рис. 6.82 показана запись для упражнения overheadpress, а на рис. 6.83 — для упражнения bicepscurl.

Модель может учиться только из того, что она будет видеть в обучающих данных, поэтому важно создать набор данных, представляющий фактические данные, которые модель будет видеть при развертывании. Если набор данных содержит данные

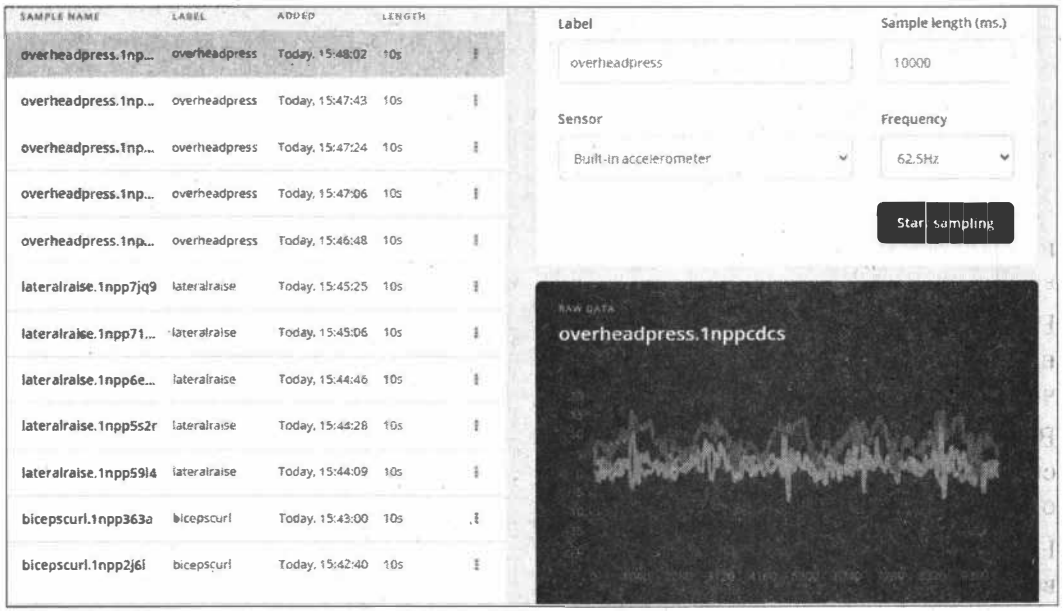


Рис. 6.82. Получаемые данные для упражнения overheadpress

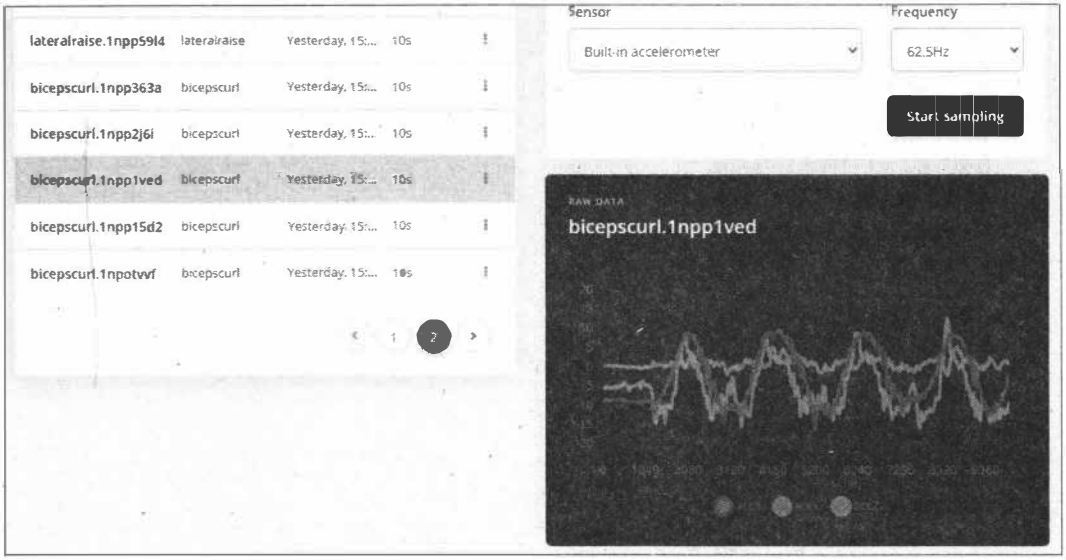


Рис. 6.83. Получаемые данные для упражнения bicepscurl

только от одного человека, модель узнает точные модели движения этого человека. Если набор данных содержит данные от многих людей, модель изучит общие характеристики каждого упражнения, которые существуют у разных людей.

С помощью тренировочного набора вы можете создать *импульс*. Импульс берет необработанные данные, разбивает их на более мелкие окна, задействует блоки обработки сигналов для извлечения функций, а затем использует обучающий блок для классификации новых данных. Блоки обработки сигналов всегда возвращают одинаковые значения для одного и того же ввода и используются для облегчения обработки необработанных данных, в то время как блоки обучения учатся на прошлом опыте.

Итак, разбиваем фрагменты на более мелкие окна (рис. 6.84). Настройка размера окна будет зависеть от продолжительности события, которое модель научится распознавать. Для данных о периодических движениях, таких как повторение упражнения с гантелями, размер окна должен включать как минимум одно полное выполнение упражнения.

Выберите блок обработки сигналов **Spectral Analysis** (рис. 6.85). Этот блок применяет фильтр, выполняет спектральный анализ сигнала и извлекает данные о его частоте и спектральной мощности (рис. 6.86).

Выберите теперь учебный блок **Neural Network** (рис. 6.87), который использует эти спектральные особенности и учится различать четыре класса: bicepscurl, lateralraise, overheadpress и idle (рис. 6.88).

Нейронная сеть классифицирует каждую выборку данных, выводя оценку достоверности для каждой метки, присутствующей в наборе обучающих данных. Входом в сеть является массив функций, который выводится из блока спектрального анализа. Тип и структуру сети можно настроить в меню классификатора **NN Classifier**.

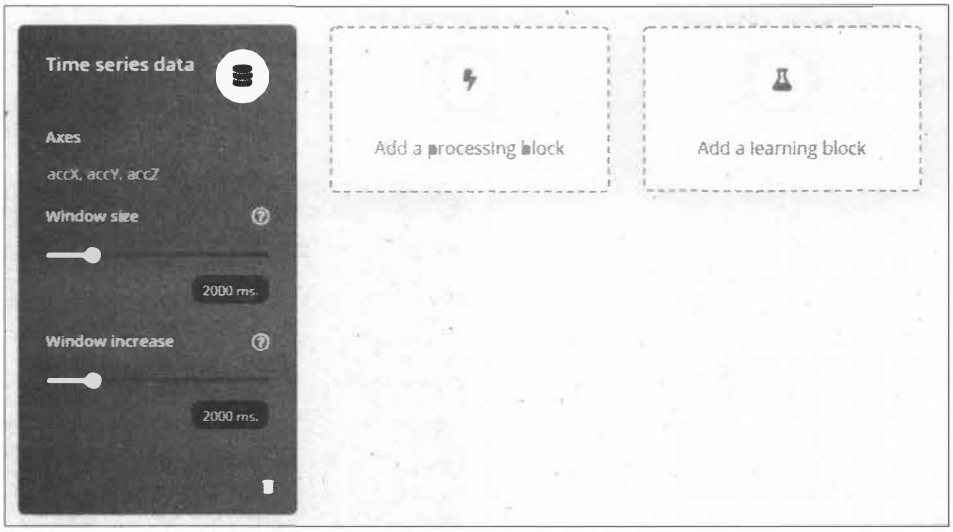


Рис. 6.84. Разбитие фрагментов на более мелкие окна

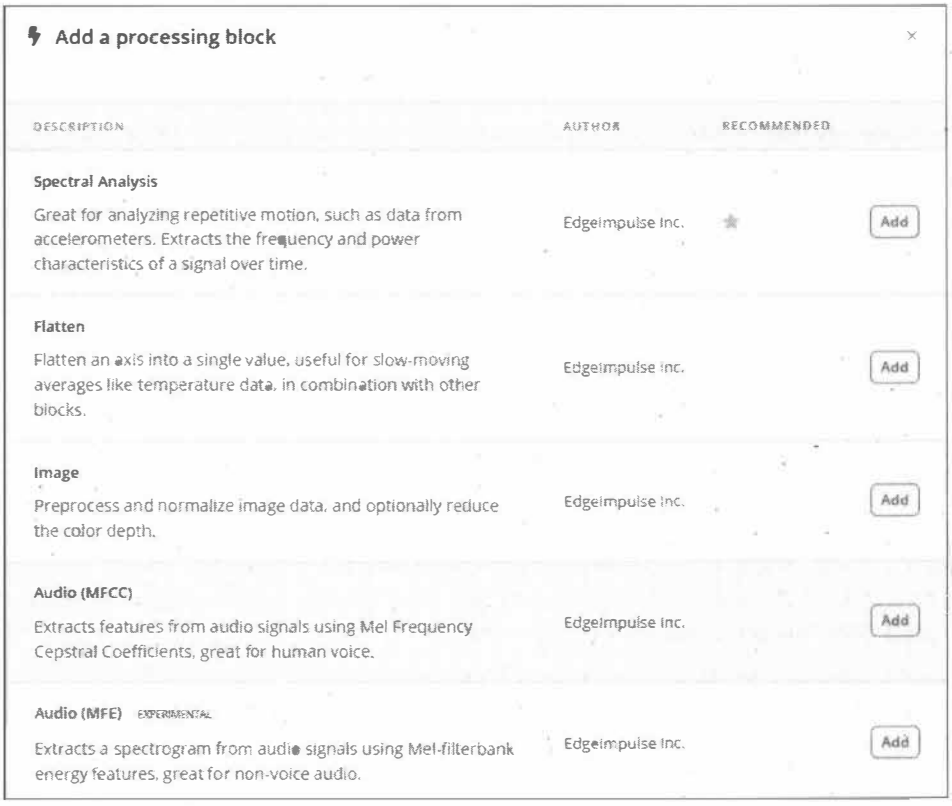


Рис. 6.85. Выбор блока обработки сигналов **Spectral Analysis**

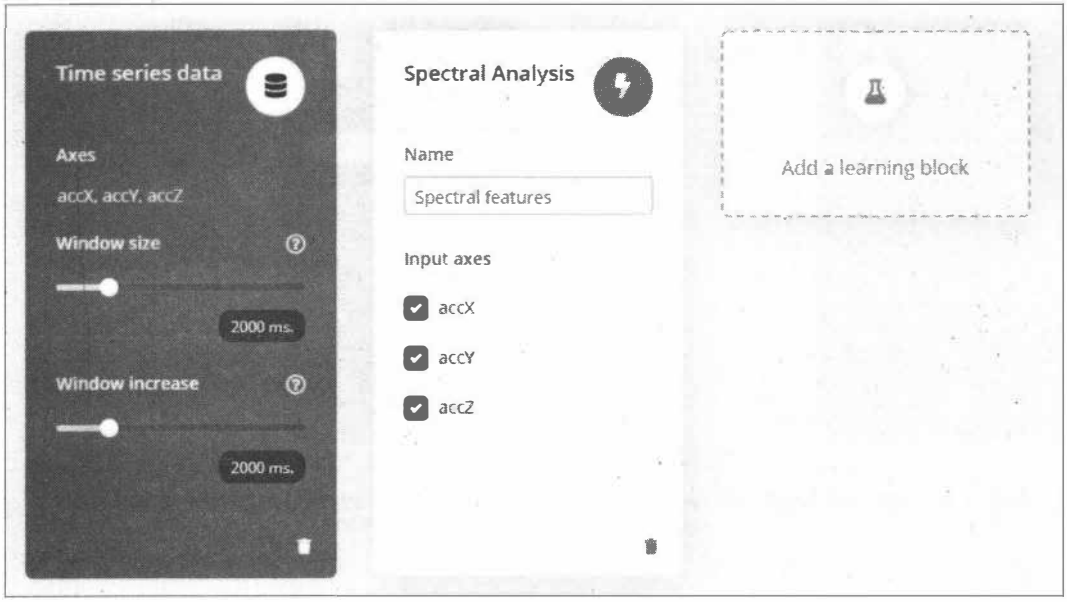


Рис. 6.86. Выполнение спектрального анализа сигнала и извлечение данных о его частоте и спектральной мощности

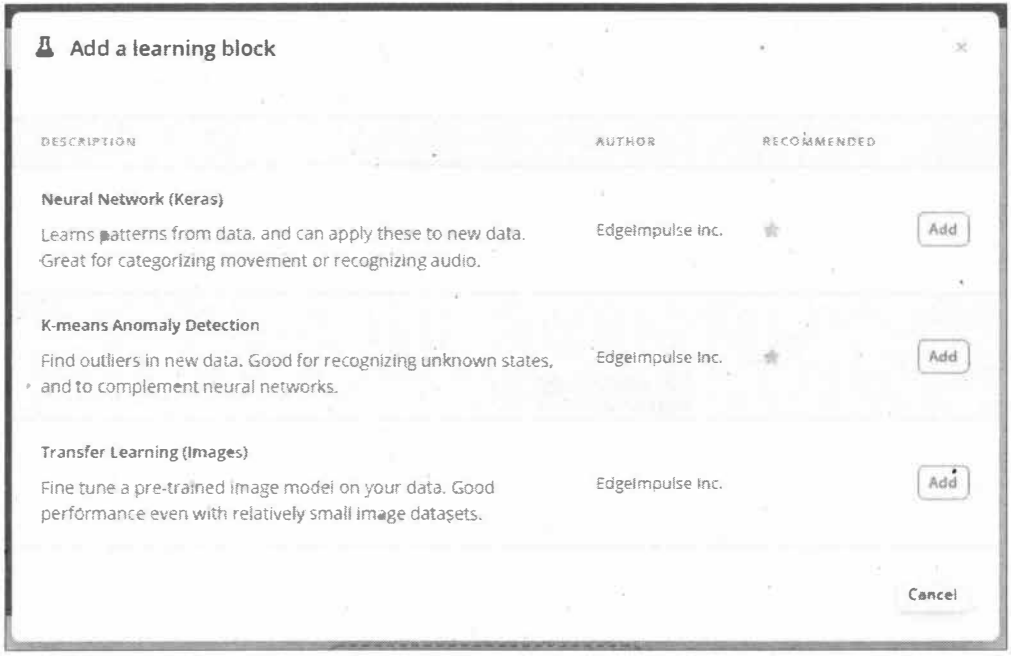


Рис. 6.87. Выбор учебного блока **Neural Network**

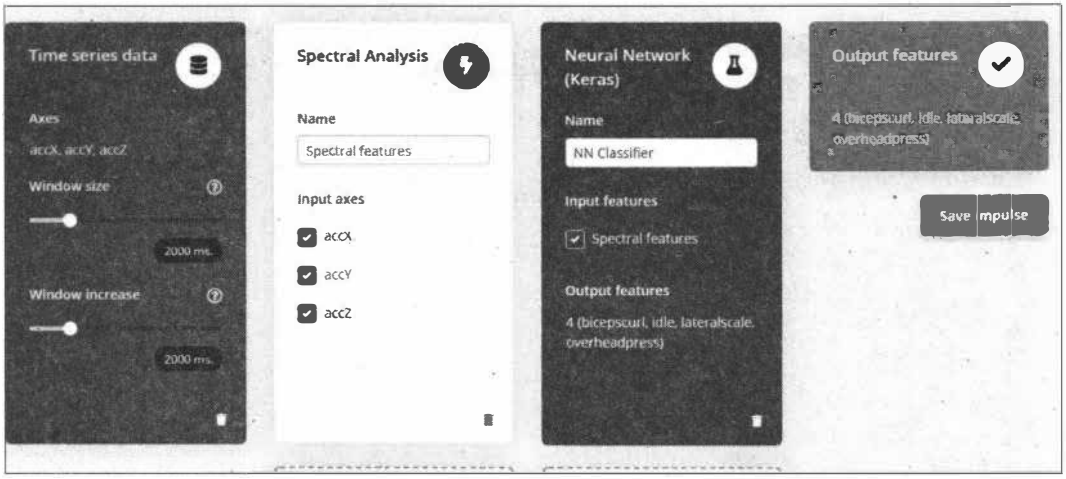


Рис. 6.88. Блок использует спектральные особенности сигналов и учится различать четыре класса: bicepscurl, lateralraise, overheadpress и idle

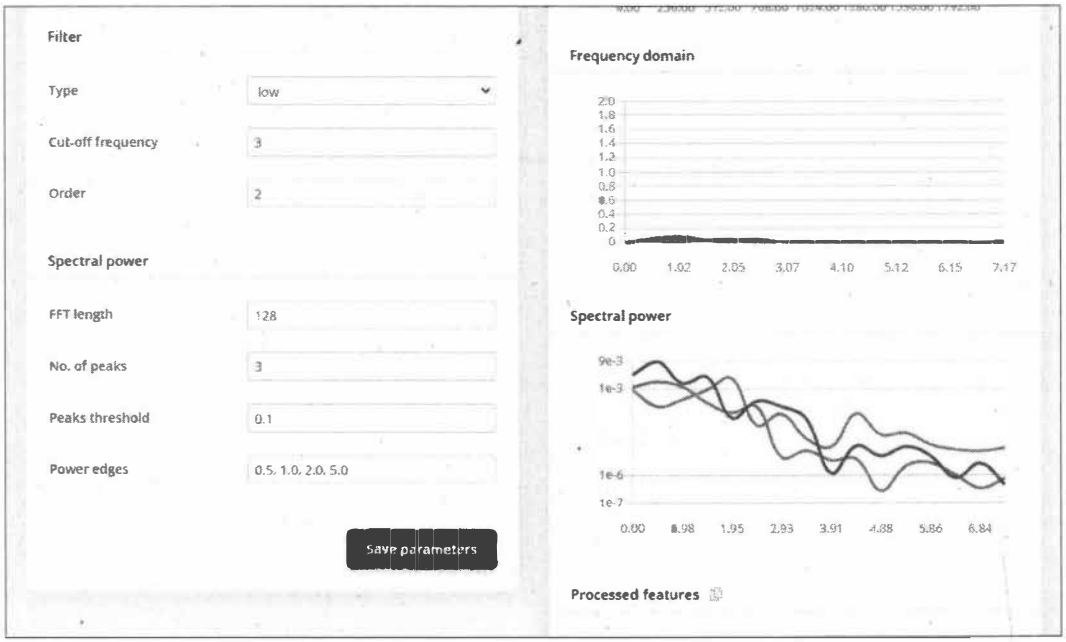


Рис. 6.89. Установка параметров обучения нейронной сети

Устанавливаем параметры обучения нейронной сети (рис. 6.89), запускаем обучение нейронной сети (рис. 6.90) и видим результат (точность распознавания) для обучающей выборки (рис. 6.91).

Инструмент тестирования моделей можно использовать для тестирования производительности модели по всему набору тестовых данных или по пакетам тестовых данных по мере необходимости (рис. 6.92). Он дает оценку точности, основанную

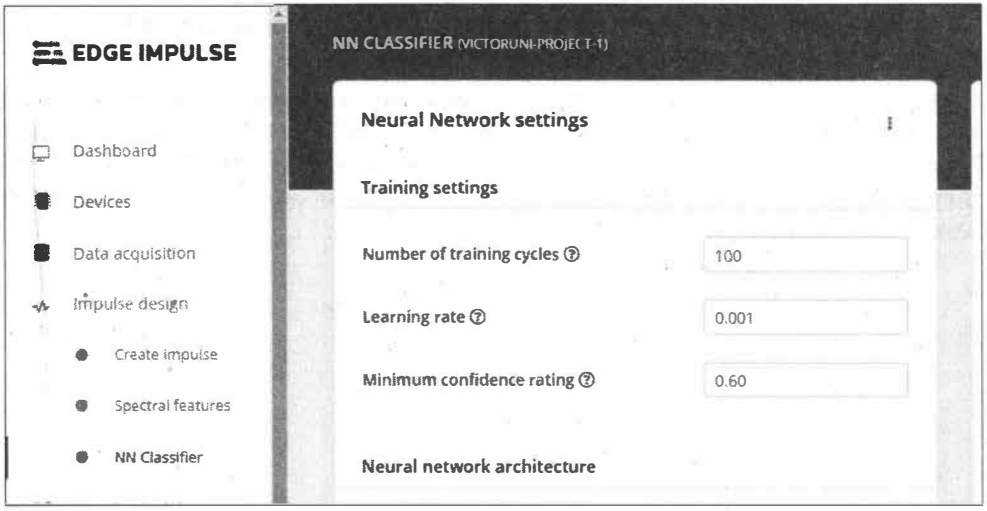


Рис. 6.90. Запуск обучения нейронной сети

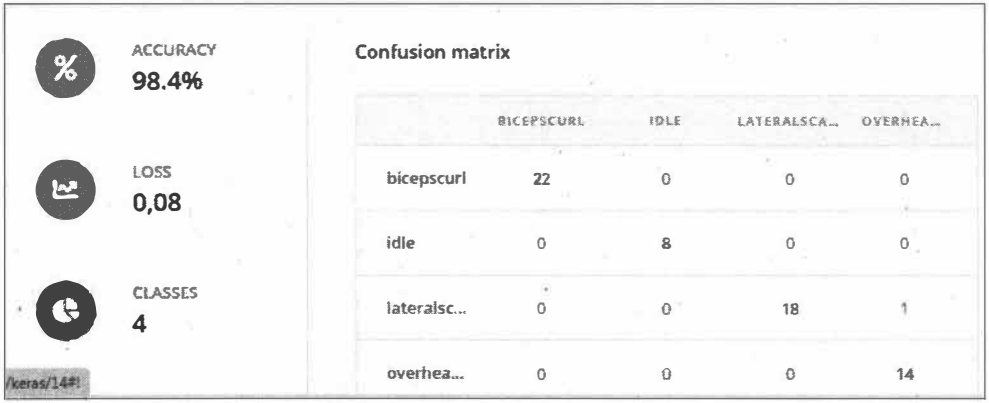


Рис. 6.91. Результаты обучения нейронной сети

Рис. 6.92. Запуск тестирования

на ожидаемом результате для каждого файла тестовых данных (рис. 6.93). Тестовые данные получаем в пункте меню **Live classification**.

Если результаты распознавания нас устраивают, необходимо сформировать код для разворачивания модели на устройстве Arduino Nano 33 BLE Sense. Сформируйте Arduino-библиотеку с созданной моделью: выберите пункт **Deployment | Arduino library** и нажмите кнопку **Build** — на компьютер скачается сформированная библиотека в ZIP-архиве (рис. 6.94).

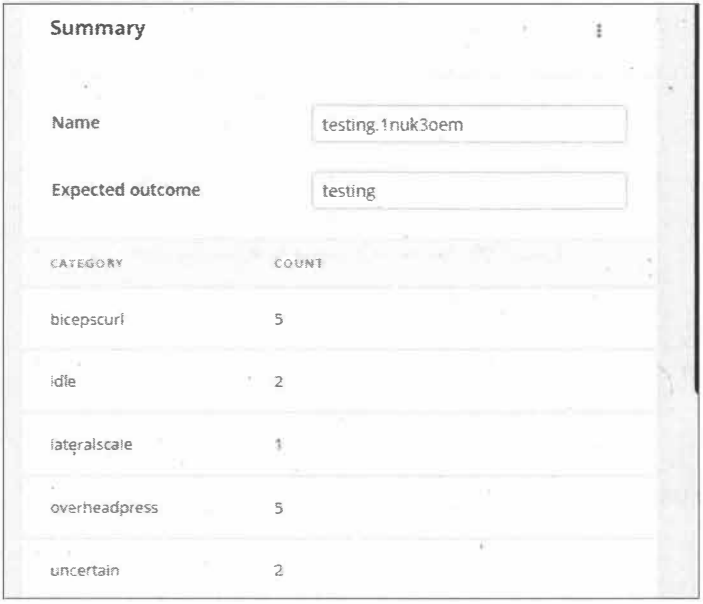


Рис. 6.93. Результаты тестирования

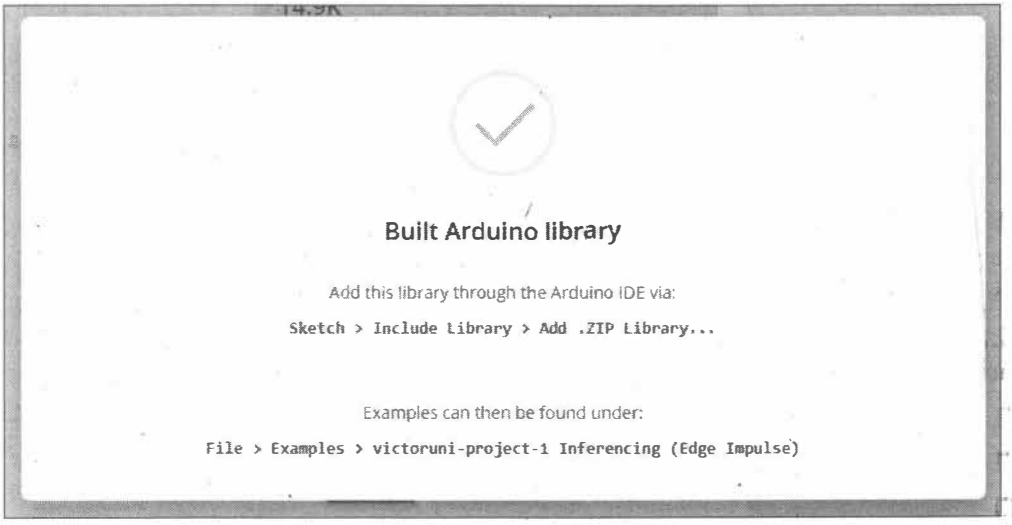


Рис. 6.94. Формирование и скачивание библиотеки

6.8.3. Создание скетча для отправки данных с платы Arduino Nano 33 BLE Sense по BLE

Приступим к созданию скетча отправки данных с Arduino Nano 33 BLE Sense при обнаружении выполненного упражнения. За основу скетча возьмем пример nano_ble_sense_accelerometer из скачанной библиотеки.

Сначала загрузите этот пример в плату и откройте монитор последовательного порта, чтобы посмотреть вывод данных при выполнении упражнений (рис. 6.95).

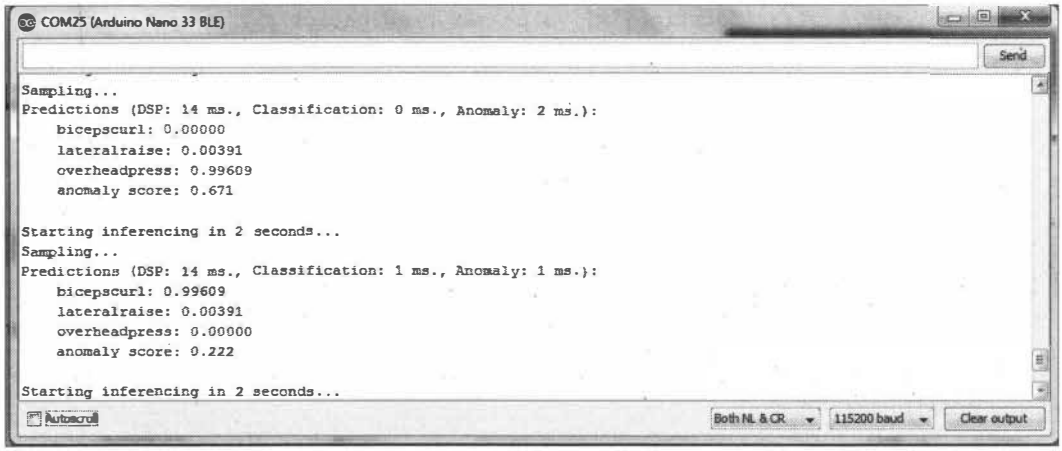


Рис. 6.95. Вывод данных в монитор последовательного порта при использовании гантели

Сохраните скетч под другим именем (dumbbell_01) и добавьте в него код подключения библиотеки ArduinoBLE:

```
#include <ArduinoBLE.h>
```

Определите BLE сервиса (uuid – 0x1826 Fitness Mashine):

```
BLEService fitnessBLEsense("1826");
```

И характеристики для упражнений (Traning status):

```
BLEIntCharacteristic counterChar("2AD3", BLERead | BLENotify);
```

Данные UUID для сервиса и характеристик берем в спецификации SIG для характеристик и сервисов (**https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20 Numbers%20Document.pdf**).

Для характеристики `counterChar` будем отправлять значения 1, 2, 3 или 4 в зависимости от определенного упражнения:

- 1 — bicepscurl;
- 2 — lateralraise;
- 3 — overheadpress;
- 4 — no или не опознано.

В процедуре `loop()` добавьте код определения максимального значения для результата классификатора объектов:

```
float max_predication=0.0;
    int result_predication=0;
    ei_printf("Predictions ");
    ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
        result.timing.dsp, result.timing.classification, result.timing.anomaly);
    ei_printf(": \n");
    for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
        if(result.classification[ix].value>max_predication)
           {
            result_predication=ix+1;
            max_predication=result.classification[ix].value;
           }
        ei_printf("    %s: %.5f\n", result.classification[ix].label,
                  result.classification[ix].value);
    }
   Serial.print("result_predication=");Serial.println(result_predication);
Serial.print("max_predication=");Serial.println  (max_predication);
```

Отправку данных по BLE осуществляем, если вероятность для какого-либо упражнения превышает 70%:

```
BLEDevice central = BLE.central();
if (central) {
   Serial.print("Connected to central: ");
   Serial.println(central.address());
   digitalWrite(LED_BUILTIN, HIGH);
   if   (central.connected()) {
       if(max_predication>0.7) {
         counter=result_predication;
         counterChar.writeValue(counter);
         //
         Serial.println("fitness BLEsense data:");
         Serial.print("counter=");Serial.print(counter);
         Serial.println("");
         //
       }
       else {
         counter=4;
         counterChar.writeValue(counter);
       }
   }
digitalWrite(LED_BUILTIN, LOW);
Serial.print("Disconnected from central: ");
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный код скетча можно найти в папке *projects\dumbbell_01* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату. На смартфоне запустите программу nrfConnect (рис. 6.96). Делайте упражнения и смотрите на данные, приходящие в сервис **Fitness Mashine**, характеристика **Traning status** (рис. 6.97).

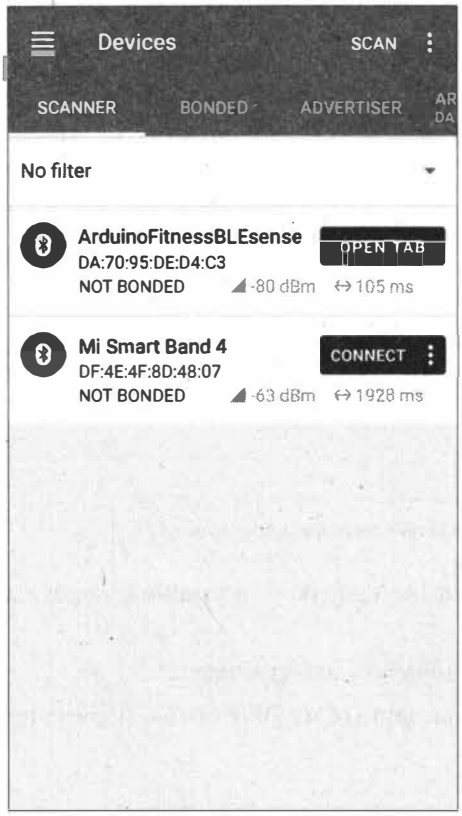


Рис. 6.96. Запуск программы nrfConnect

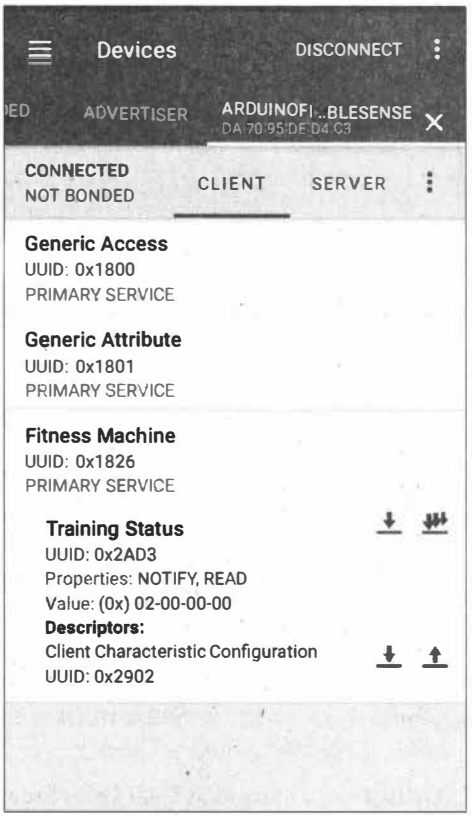


Рис. 6.97. Получение данных от «умной гантели» в программе nrfConnect

ЭЛЕКТРОННЫЙ АРХИВ

Полный код скетча можно найти в папке *projects\dumbbell_01* сопровождающего книгу электронного архива (см. *приложение*).

6.8.4. Создание программы для смартфона

Приложение для смартфона мы создадим в App Invertor 2 — онлайн-редакторе визуального программирования для Android, доступ к которому можно получить на странице: **http://ai2.appinventor.mit.edu**. После регистрации или авторизации (можно использовать профиль Google) попадаем в свой профиль программы, где создаем новый проект (рис. 6.98).

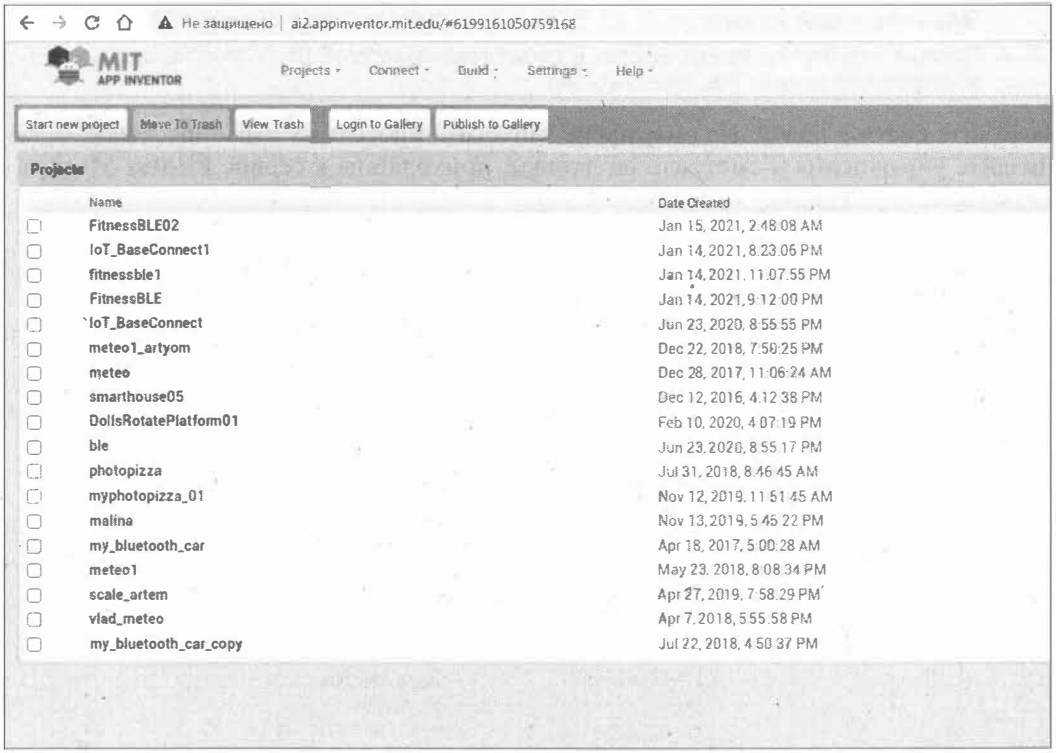


Рис. 6.98. Создание проекта в своем профиле программы App Invertor 2

Сначала в панели **Designer** формируем интерфейс нашего приложения, перетаскивая на экран необходимые компоненты.

Кроме визуальных компонентов необходимо добавить невизуальные:

- **Clock** — из раздела **Sensors** (для получения данных из Bluetooth с периодичностью, установленной в Clock);
- **Notifer** — из раздела **UserInterface**;
- **BluetoothBLE**.

Компонент BluetoothBLE не входит в стандартный набор компонентов, и его необходимо подгрузить — скачайте для этого со страницы расширений **https://mit-cml.github.io/extensions/** компонент BluetoothBLE.aix (edu.mit.appinventor.ble-20200828.aix). Для его установки выполните команду **Extension | Import extension**, в открывшемся окне выберите путь к скачанному компоненту и нажмите на кнопку **Import** — компонент **BluetoothBLE** появится в разделе **Extension** (рис. 6.99), и мы можем добавить его в интерфейс.

Включите в интерфейс метки для названия упражнений и поля для отображения количества упражнений. Добавьте кнопку для сброса количества упражнений. Кроме этих полей, понадобятся еще компоненты для организации подключения BLE, кнопки запуска и остановки сканирования BLE-устройств, поле списка для вывода обнаруженных при сканировании BLE-устройств (рис. 6.100).

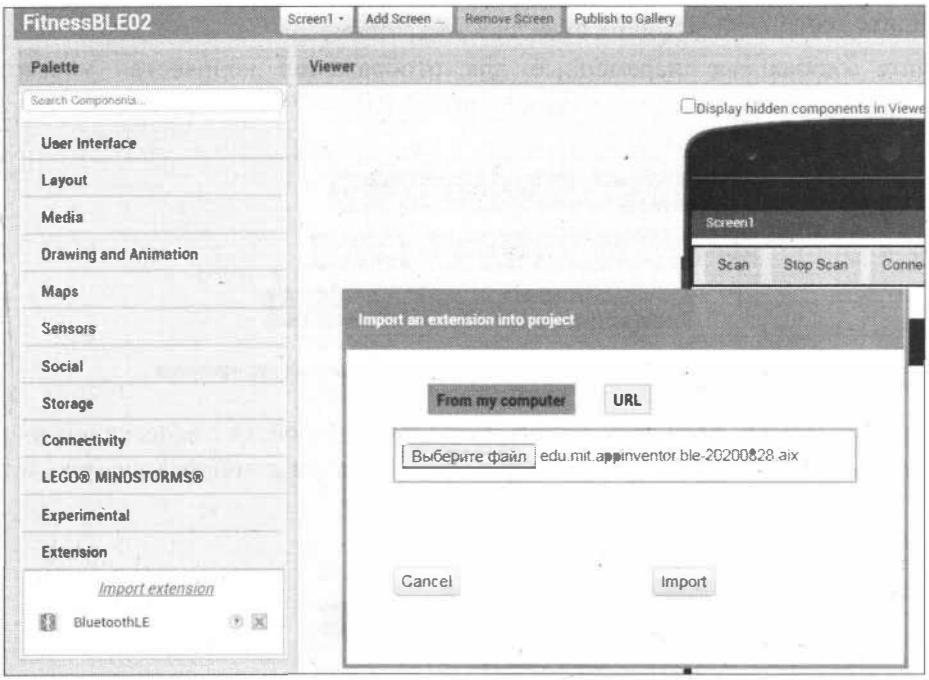


Рис. 6.99. Импорт компонента BluetoothBLE в App Invertor 2

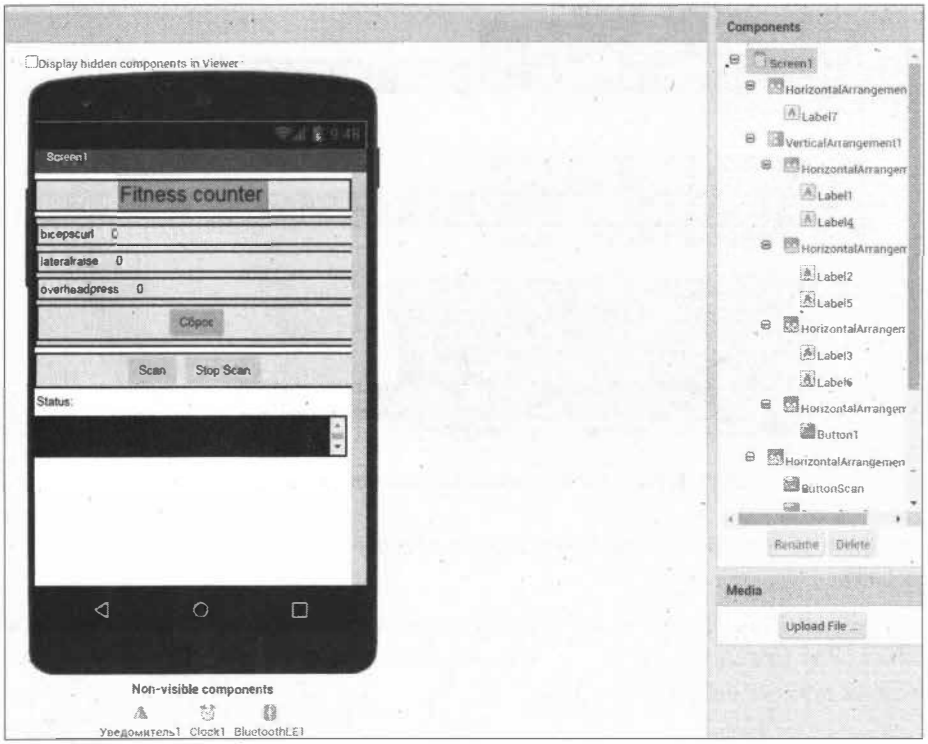


Рис. 6.100. Создание интерфейса приложения в панели **Designer**

Перейдите теперь для создания кода в раздел **Block**.

Создайте глобальные переменные для отображения количества упражнений: `bicepscurl`, `lateralraise`, `overheadpress` (рис. 6.101).

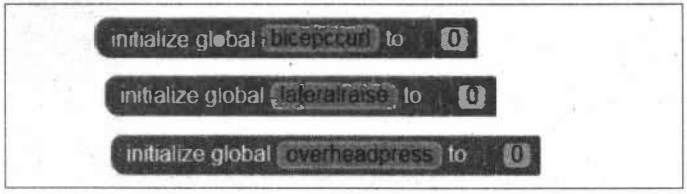


Рис. 6.101. Фрагмент кода создания глобальных переменных

По нажатию кнопки **Scan** запускается поиск BLE-устройств с занесением их в список `ListBLE`, а по нажатию кнопки **Stop Scan** — останов сканирования (рис. 6.102).

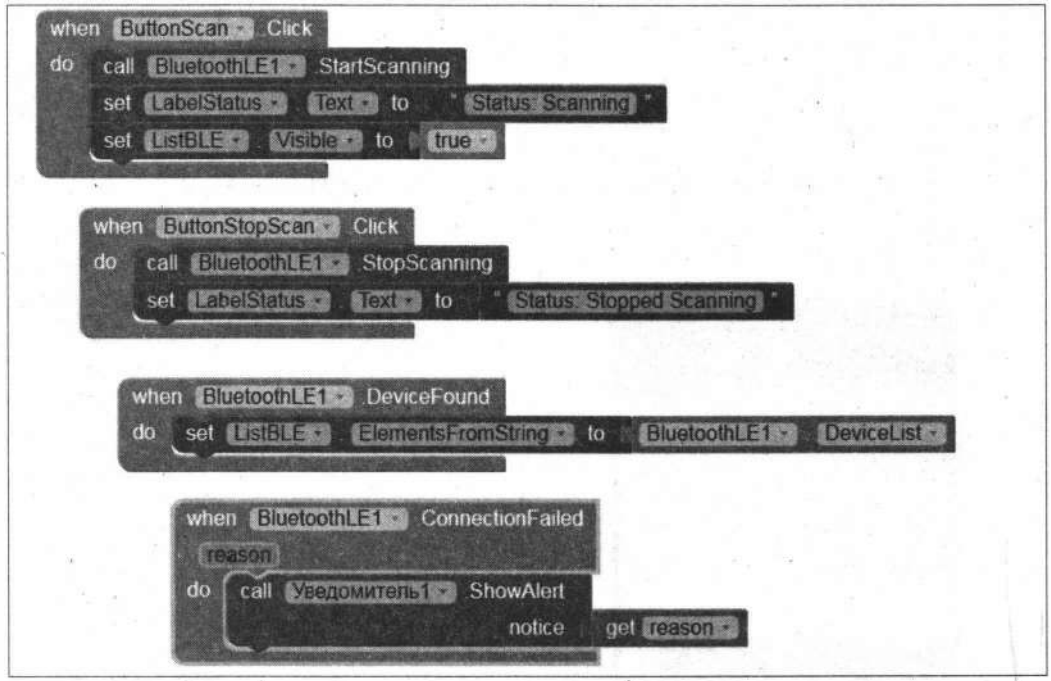


Рис. 6.102. Фрагмент кода сканирования BLE-устройств

Инициализация подключения производится при выборе BLE-устройства из списка (рис. 6.103).

Для получения и вывода данных у характеристики `counterChar` активируется свойство **NOTIFY** (Уведомление) — при помощи уведомлений мы получаем новые данные каждую секунду (рис. 6.104).

Обнуление данных для `bicepscurl`, `lateralraise`, `overheadpress` осуществляется по щелчку на кнопке **Сброс** (рис. 6.105).

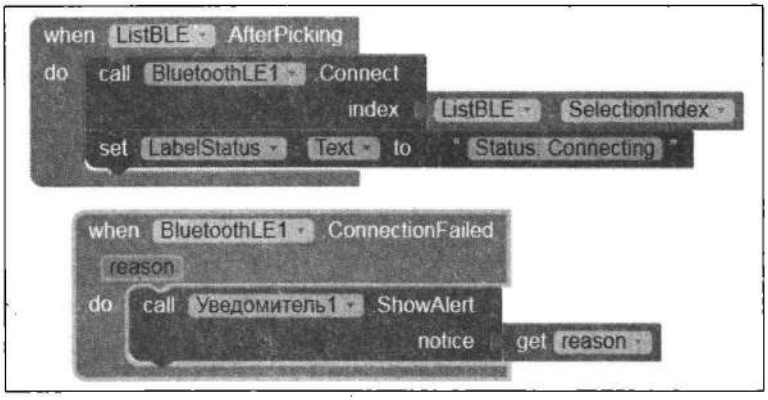


Рис. 6.103. Подключение к BLE-устройству

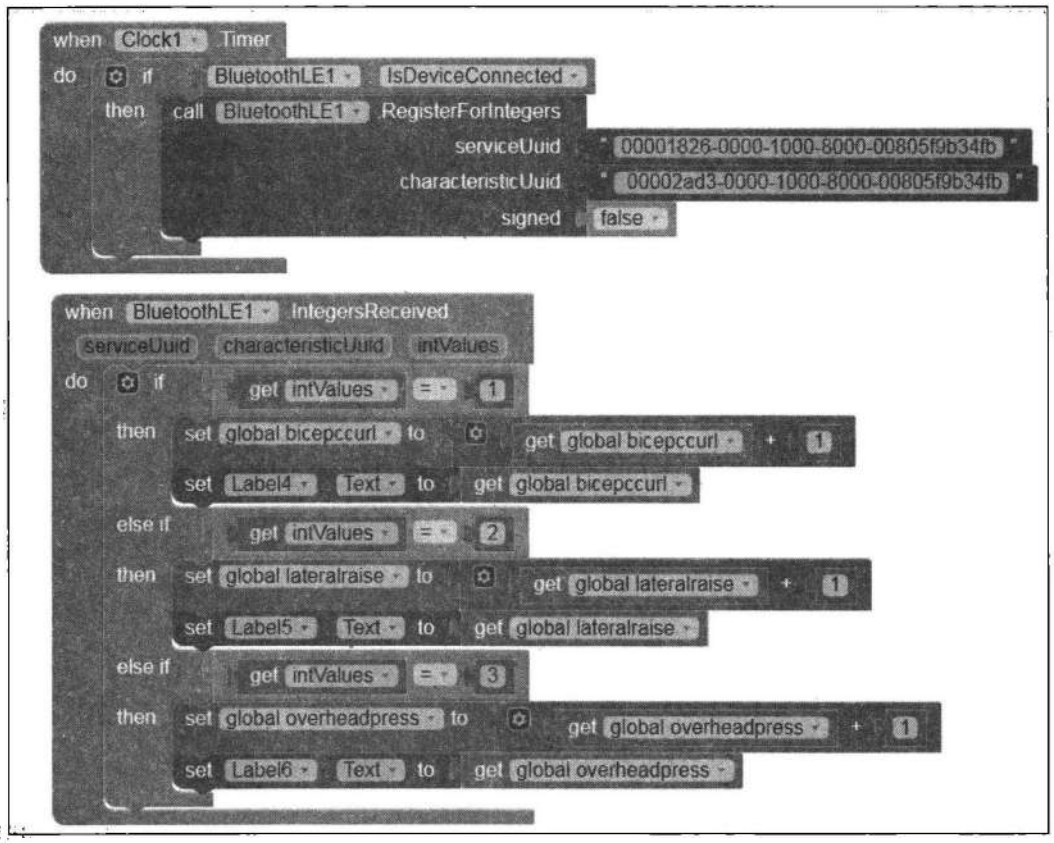


Рис. 6.104. Получение и вывод данных

В завершение командой **App** создаем приложение (рис. 6.106) и загружаем его в смартфон/планшет.

Запустите приложение на смартфоне/планшете (рис. 6.107), подключите его по Bluetooth к плате Arduino (рис. 6.108), и, выполняя упражнения, вы увидите отображение в нем получаемых от «умной гантели» данных (рис. 6.109).

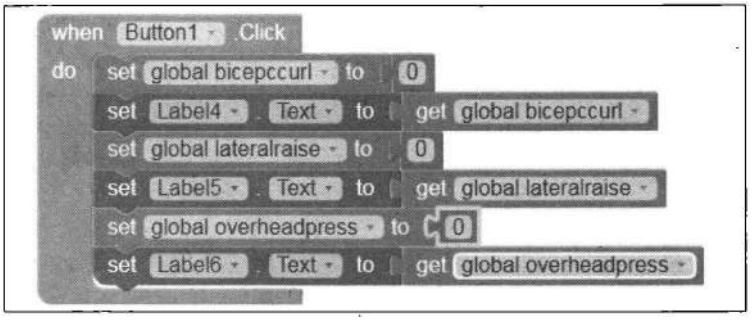


Рис. 6.105. Обнуление данных

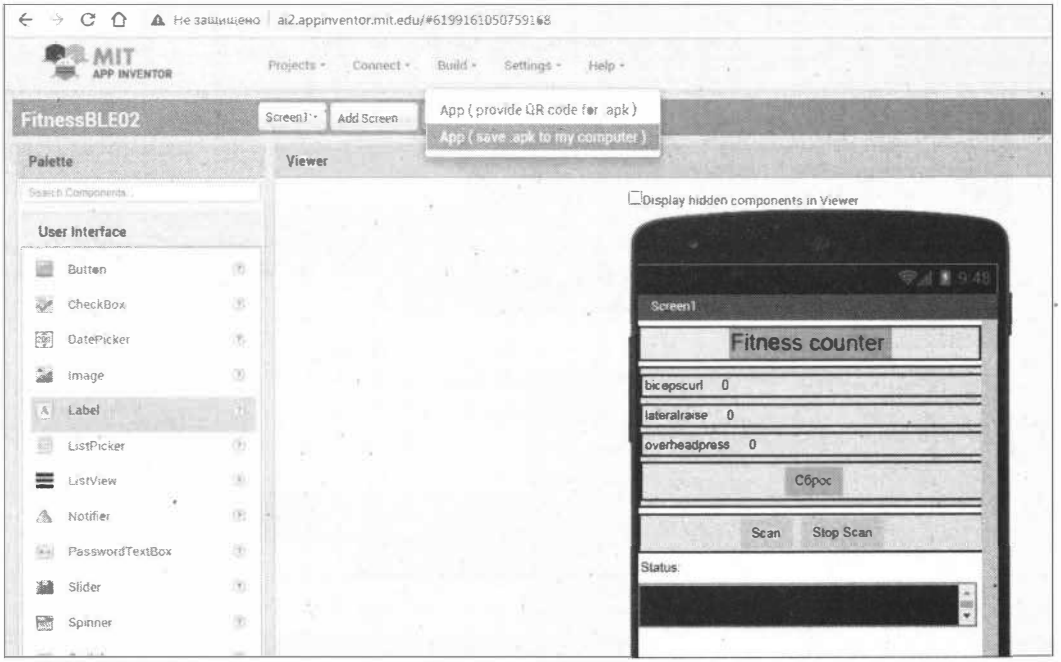


Рис. 6.106. Создание приложения

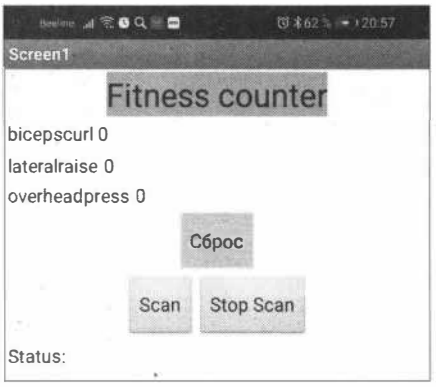


Рис. 6.107. Приложение запущено на планшете

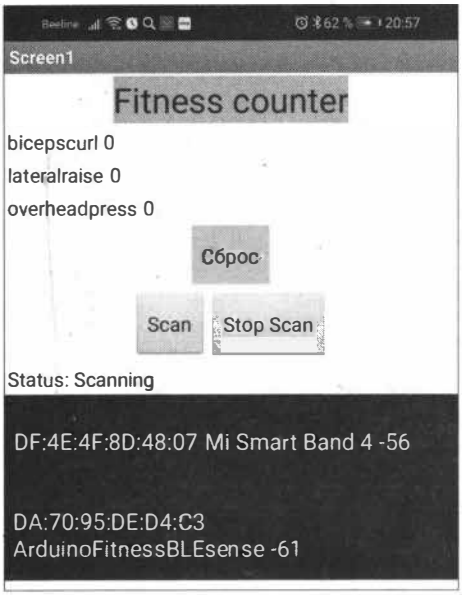


Рис. 6.108. Подключение приложения по Bluetooth к плате Arduino

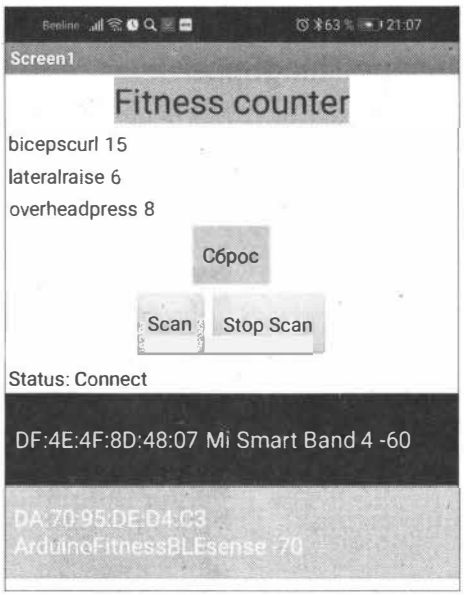


Рис. 6.109. Получение и отображение данных

6.9. Предсказатель погоды на основе данных, поступающих в сервис ThingSpeak

В *разд. 5.3.5* мы рассмотрели простое преобразование данных, поступающих в канал ThingSpeak. В этом разделе мы создадим предсказатель погоды, использующий данные, поступающие в сервис ThingSpeak, и воспользуемся для этого инструментами MATLAB.

По изменению значений атмосферного давления можно спрогнозировать, как изменится погода в ближайшие 12–24 часов. Если давление падает, то вероятно, что погода будет ухудшаться. Если давление растет, то это признак улучшения погоды. Однако само текущее значение атмосферного давления не может служить признаком возможного улучшения или ухудшения погоды — важна именно динамика его изменения. Прямой график без резких колебаний давления позволяет быть уверенным, что погода в ближайшие несколько часов останется той же. А о том, что погода скоро изменится, говорит резкое изменение атмосферного давления.

Так, если на улице солнечная погода, а график давления показывает резкое его падение, то это явный признак того, что либо пойдет дождь, либо небо закроет густая облачность. А если же на улице пасмурно и дождливо, то резкий подъем давления подскажет, что погода скоро станет солнечной.

Резким изменением давления, предвещающим бурю или дождь, является падение его значения на 200–300 Па/час. Так что наша задача — определять величину изменения давления в Па/час для данных, поступающих в сервис ThingSpeak.

6.9.1. Отправка данных атмосферного давления в сервис ThingSpeak

В качестве контроллера в этом проекте мы воспользуемся платой Arduino Nano 33 IoT с подключенным к ней датчиком BME280. Монтажная схема подключения показана на рис. 6.110.

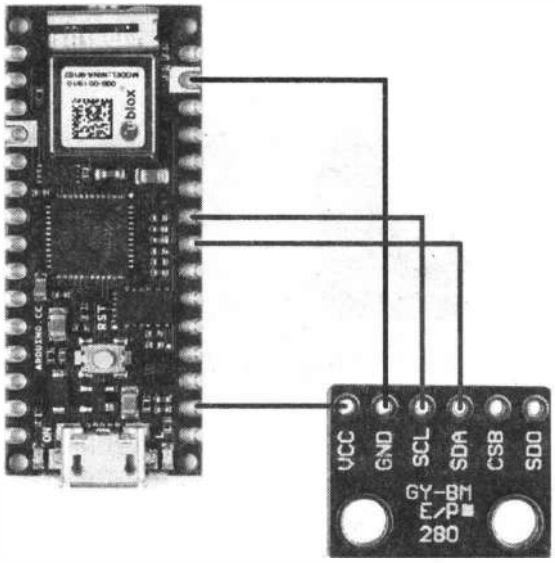


Рис. 6.110. Монтажная схема подключения датчика BME280 к плате Arduino Nano 33 IoT

Для работы с датчиком потребуется установить две библиотеки: Adafruit BME280 Library и Adafruit Sensor. Чтобы работать с датчиком по протоколу I^2C, необходимо определить I^2C-адрес датчика, т. к. он может иметь адрес `0x76` или `0x77`, и, возможно, внести изменения в файлы библиотеки. Более подробно эта процедура описана в *разд. 5.3.1*.

Напишем скетч, взяв за основу алгоритм, который используется при создании предсказателя погоды известного блогера Alexgyver.

Измерение давления производим один раз в 10 минут. Чтобы устранить «шум» измерений, выполняем 5 измерений и получаем среднее арифметическое, которое отправляем в канал ThingSpeak. Таким образом, мы будем получать 6 значений в час. Для получаемых значений построим график изменения давления с помощью линейной аппроксимации методом наименьших квадратов (рис. 6.111). Эта прямая и покажет значение изменения давления в Па/час.

Данные давления отправляем в сервис ThingSpeak по протоколу MQTT (в рассматриваемом примере — в поле **Field 2** канала **Arduino_Nano_33_IoT_1**), как показано на рис. 6.112.

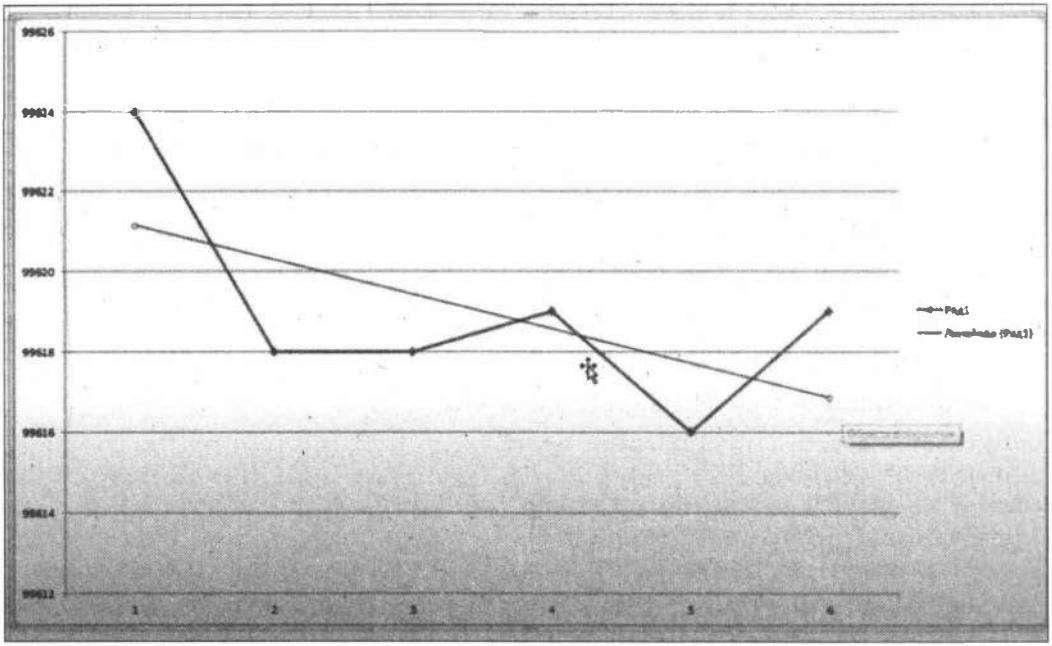


Рис. 6.111. Линейная аппроксимация методом наименьших квадратов

Рис. 6.112. Канал отправки данных для проекта в сервисе ThingSpeak

Подключаем необходимые библиотеки:

```
#include <SPI.h>
#include <WiFiNINA.h>
#include <PubSubClient.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
```

Создаем экземпляры объектов:

```
WiFiClient client;
Adafruit_BME280 bme; // I2C
PubSubClient mqttClient(client);
```

Вносим сведения о канале ThingSpeak и ключах:

```
const char* server = "mqtt.thingspeak.com";
// ThingSpeak Settings
char mqttUserName[] = "TSArduinoMQTTDemo";
char mqttPass[] = "0AJ9UGHFF763MPCI";
char writeAPIKey[] = "RKVTWSNB3NH7FVXC";
long channelID = 1215210;
int fieldNumber = 2;
// For random generation of client ID.
static const char alphanum[] ="0123456789"
          "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
          "abcdefghijklmnopqrstuvwxyz";
```

Интервал отправки данных на сервер:

```
unsigned long updateThingSpeakInterval = 10 * 60 * 1000;
```

В разделе `loop()` устанавливаем соединение MQTT и публикуем данные в канале через заданный интервал времени:

```
void loop()  {
  // Проверяем соединение MQTT
  if (!mqttClient.connected())
  {
    reconnect();
  }
  // Вызываем цикл непрерывно для установки соединения с сервером.
  mqttClient.loop();
  // публикация данных в ThingSpeak.
  if (millis() - lastConnectionTime > updateThingSpeakInterval)
  {
    mqttPublishFeed();.
  }
}
```

Функция `reconnect()` для подключения клиента Arduino к брокеру MQTT:

```
void reconnect()
{
  char clientID[9];
```

```
  // цикл подключения
  while (!mqttClient.connected())
  {
    Serial.print("Attempting MQTT connection...");
    // Генерация ClientID
    for (int i = 0; i < 8; i++) {
        clientID[i] = alphanum[random(51)];
    }
    clientID[8]='\0';

    // Подключение к брокеру
    if (mqttClient.connect(clientID,mqttUserName,mqttPass))
    {
      Serial.println("connected");
    } else
    {
      Serial.print(mqttClient.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}
```

Функция `mqttPublishFeed()` **предназначена для публикации данных датчика в канале ThingSpeak. С помощью этой функции вы можете публиковать сразу в несколько полей. В следующем примере выполняется публикация в поля** `field1` **(температура) и** `field2` **(атмосферное давление в Па):**

```
// отправка данных в канал ThingSpeak
void mqttPublishFeed() {
    t=bme.readTemperature();
    p=0;
    for(int i=0;i<5;i++) {
       p=bme.readPressure();
    delay(2000);
    }
    p=p/5;
    Serial.print("Temperature = ");
    Serial.print(t);
    Serial.println(" *C");
    Serial.print("Pressure = ");
    Serial.print(p);
    Serial.println(" Pa");

  // Формирование строки для отправки в ThingSpeak.
  String data = String("field1=") + String(t, DEC) + "&field2="
              + String(p, DEC);
```

```
  int length = data.length();
  const char *msgBuffer;
  msgBuffer=data.c_str();
  Serial.println(msgBuffer);

  // Создание топика и публикация в канал.
  String topicString = "channels/" + String( channelID ) +
          "/publish/"+String(writeAPIKey);
  length = topicString.length();
  const char *topicBuffer;
  topicBuffer = topicString.c_str();
  mqttClient.publish( topicBuffer, msgBuffer );
  lastConnectionTime = millis();
}
```

ЭЛЕКТРОННЫЙ АРХИВ

Полный код скетча можно найти в папке *projects\weatherForecaster\01* сопровождающего книгу электронного архива (см. *приложение*).

Загрузите скетч в плату Arduino, и через некоторое время вы увидите графики отправки значений в канал ThingSpeak (рис. 6.113).

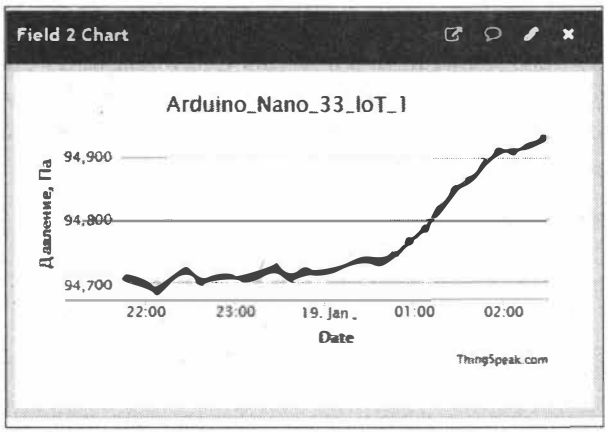


Рис. 6.113. Данные атмосферного давления, приходящие в канал ThingSpeak

6.9.2. Преобразование данных в MATLAB

Рассмотрим преобразование данных, поступивших в ThingSpeak. Для этого используются инструменты MATLAB. Создадим канал, в который будем получать параметры прямого изменения давления в час для 6 значений, вычисленных с помощью линейной аппроксимации методом наименьших квадратов.

Сценарий анализа MATLAB создается из кода шаблона: на странице канала ThingSpeak нажмите на кнопку **Matlab Analysis,** выберите шаблон **Custom (no starter code)** и нажмите на кнопку **Create** (рис. 6.114).

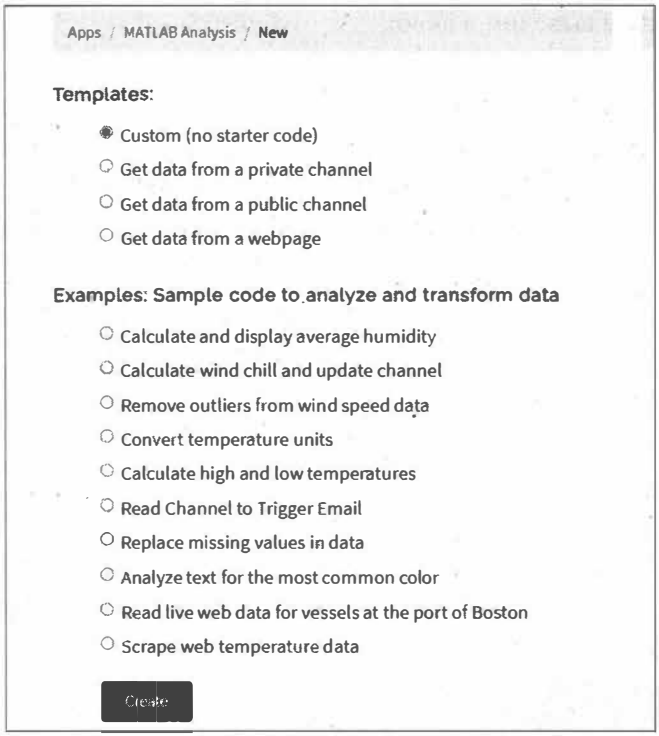


Рис. 6.114. Создание сценария MATLAB на основе шаблона Custom

В открывшееся окно **MATLAB Code** введите следующий код.

Объявляем переменные:

```
readChID = 1215210;
pressureFieldID = 2;
readAPIKey = 'UPX1IRGEQT5XJLBA'
```

где:

- `readChannelID` — ID канала для чтения данных;;
- `pressureFieldID` — номер поля данных давления (`Field2`);
- `readAPIKey` — для публичного канала оставляем пустое значение. Если чтение данных будет производиться из приватного канала, необходимо внести значение поля **Read API Keys** с вкладки **API keys**.

Получаем последние 6 данных поля `pressureFieldID (Field2)` из канала `readChannelID`:

```
pressure=thingSpeakRead(readChID,'Fields',pressureFieldID,'ReadKey',readAPIKey,
'NumPoints',6);
```

Поле значений времени 1 час для 6 точек представляем в виде:

```
hour=[0.0,0.2,0.4,0.6,0.8,1.0];
```

Применяем функцию `polifit()`:

```
s=polyfit(hour, pressure, 1);
```

Полученные данные выводим в окно:

```
display(s,'polifit result');
display(num2str(s(1)));
display(num2str(s(2)));
```

Нас интересует коэффициент наклона Па/час — значение `s(1)`.

Сохраните и запустите сценарий кнопкой **Save and Run**. Результат показан на рис. 6.115.

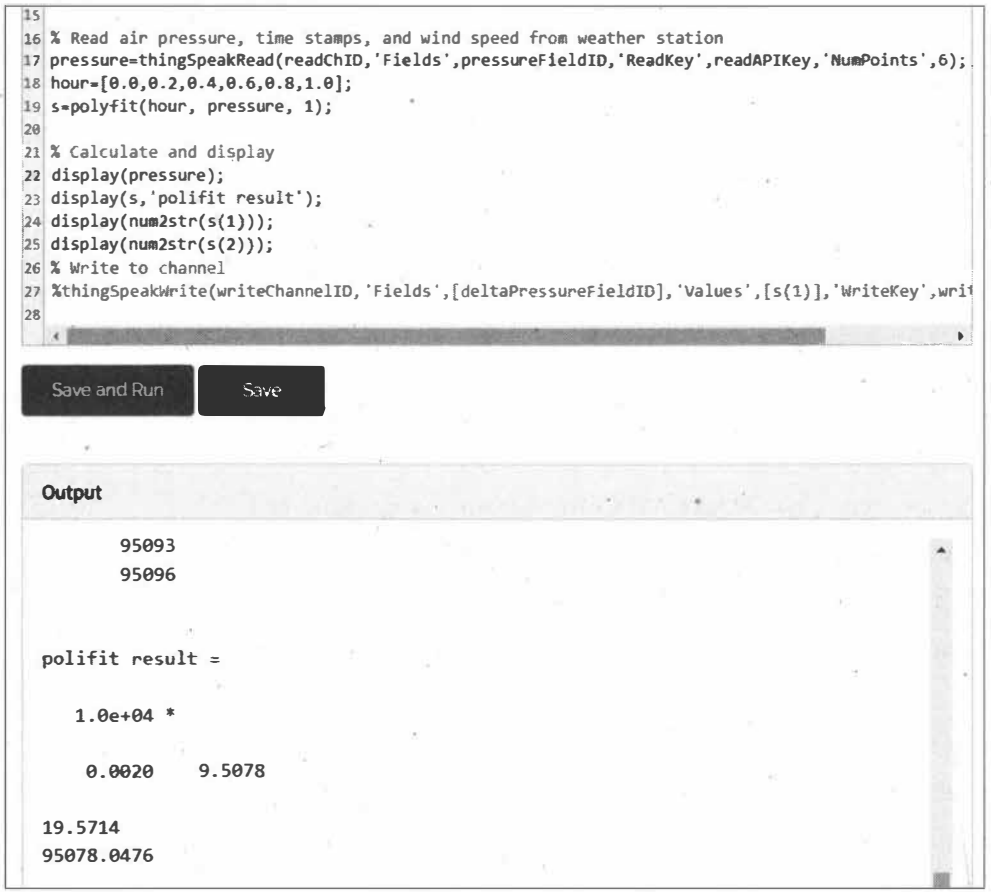


Рис. 6.115. Результат выполнения сценария MATLAB

Теперь добавим отправку конвертированных данных в другой канал. Активируем для этого в канале Arduino_Nano_33_IoT еще одно поле: **Field 5 — deltaPa/hour** (рис. 6.116).

Объявляем переменные для канала Arduino_Nano_33_IoT:

```
writeChannelID = 1131248;
deltaPressureFieldID = 5;
writeAPIKey = 'D110B1A6867IUZO9';
```

и отправляем полученное значение `deltaPa/hour` **в канал Arduino_Nano_33_IoT:**

```
thingSpeakWrite(writeChannelID,'Fields',[deltaPressureFieldID],'Values',{s(1)},
'WriteKey',writeAPIKey);
```

Сохраняем и запускаем сценарий — произойдет однократная отправка данных в канал Arduino_Nano_33_IoT (рис. 6.117).

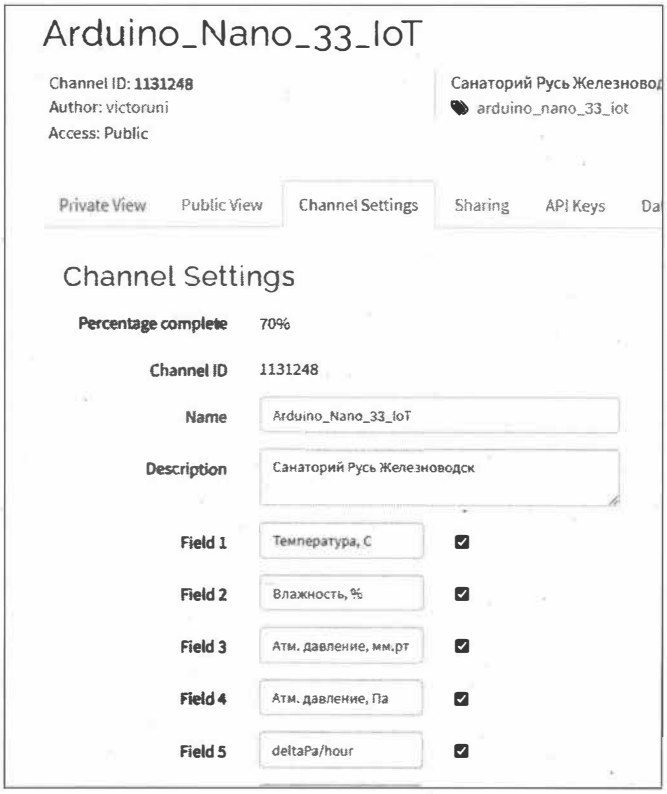


Рис. 6.116. Новое поле deltaPa/hour в канале ThingSpeak

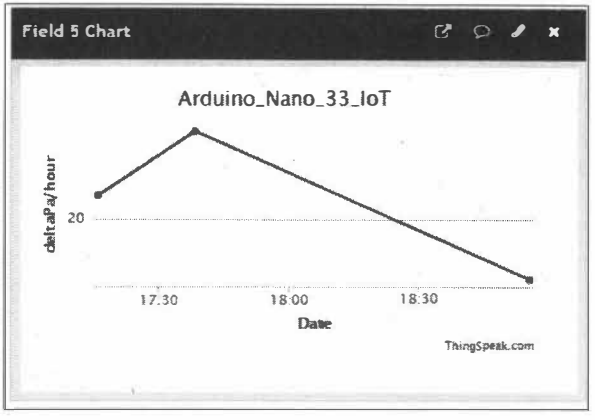


Рис. 6.117. Данные deltaPa/hour в канале Arduino_Nano_33_IoT

Теперь необходимо настроить периодичность отправки данных (периодичность запуска сценария). Командой **Apps | TimeControl | New TimeControl** создайте новое значение **NewTimeControl** и задайте для него периодичность запуска сценария. Затем в поле **Code to execute** выберите наш сценарий **weatherForecaster** и сохраните настройки, нажав на кнопку **Save TimeControl** (рис. 6.118).

Теперь данные в канале Arduino_Nano_33_IoT будут появляться каждые 10 минут (рис. 6.119).

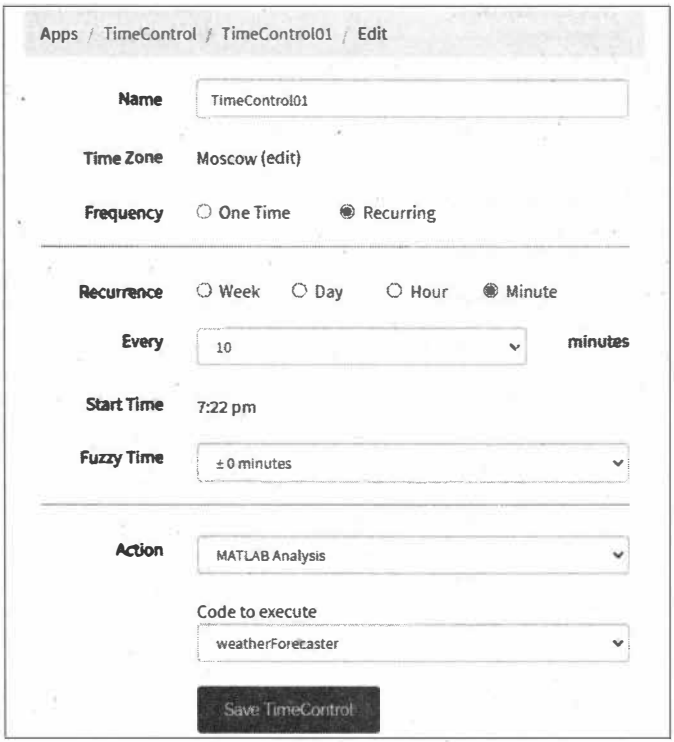


Рис. 6.118. Создание периодичности запуска сценария

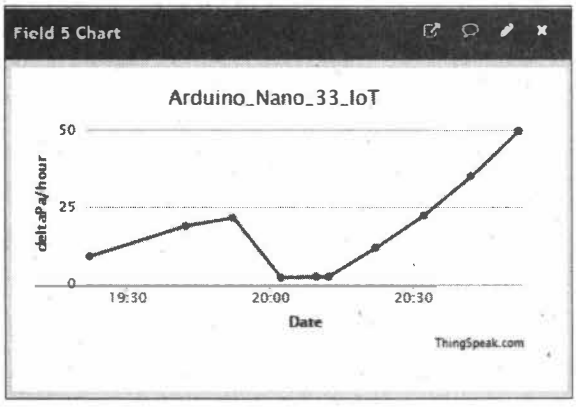


Рис. 6.119. Поступление данных **deltaPa/hour** в канал Arduino_Nano_33_IoT с заданной периодичностью

6.9.3. Добавление виджета

Для добавления виджета перейдите в канал, на вкладке **Public View** (или **Private View**) нажмите на кнопку **Add Widgets**, в открывшемся окне выберите необходимый виджет (рис. 6.120) и нажмите на кнопку **Next**.

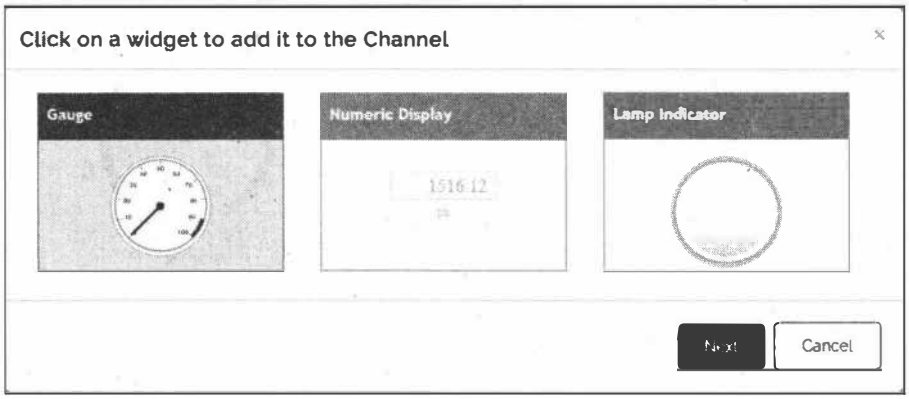


Рис. 6.120. Добавление виджета в канал ThingSpeak

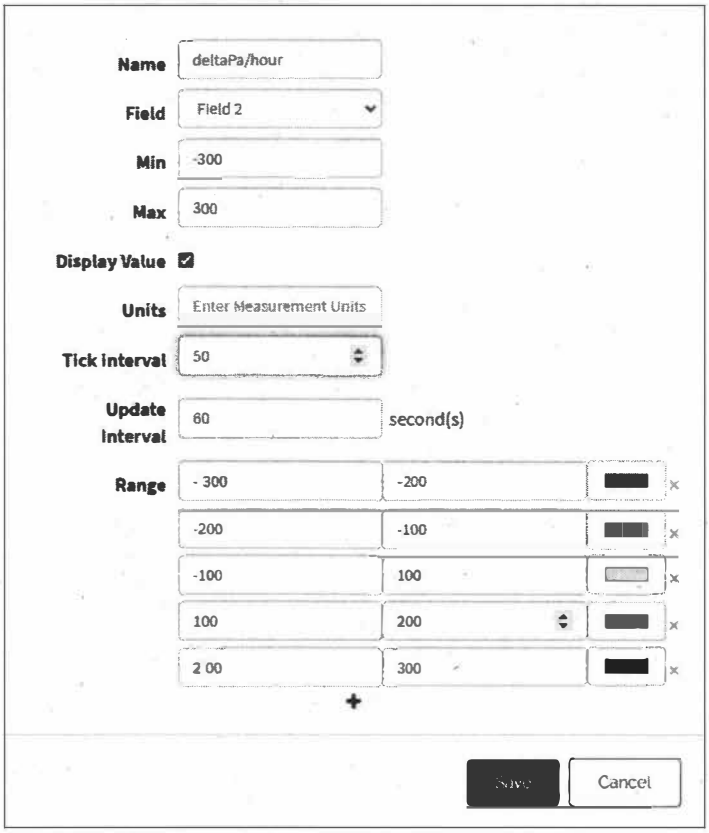


Рис. 6.121. Настройка параметров виджета

В следующем открывшемся окне (рис. 6.121) настройте параметры виджета: выберите поле канала для отображения, интервалы значений, интервал и сохраните введенные значения, нажав на кнопку **Create**. Виджет появится в канале (рис. 6.122).

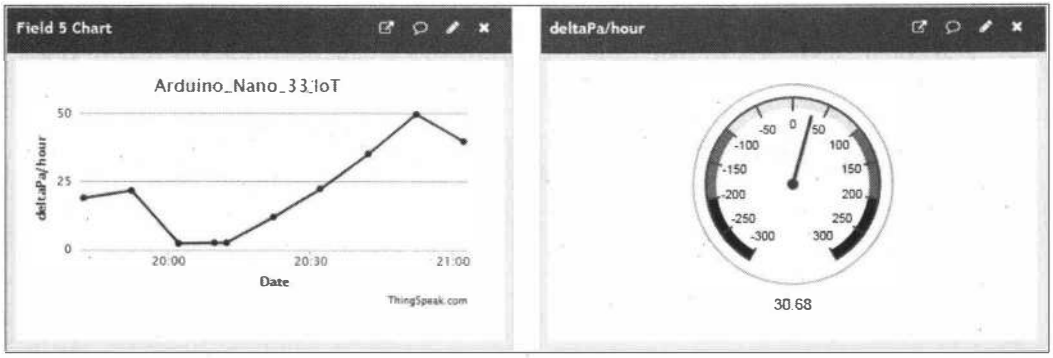


Рис. 6.122. Виджет добавлен в канал

Значение виджета, близкое к нулю, не предвещает изменения погоды. При значениях от –300 до –200 и от 200 до 300 ждите скорого изменения погоды.

Если виджет создан в **Public View**, можно предоставить доступ для его просмотра другим пользователям. По ссылке вида **https://www.thingspeak.com/channels/<channelID>/widgets/<widgetID>** все желающие увидят страницу с виджетом, показанную на рис. 6.123.

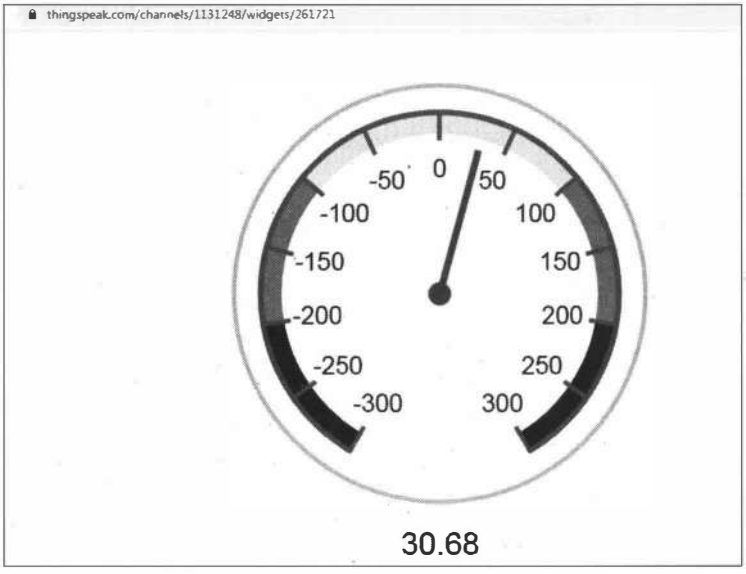


Рис. 6.123. Страница для просмотра виджета

Заключение

Интернет вещей (Internet of things) — это новое и стремительно развивающееся направление. Интернет-вещи призваны сделать нашу жизнь еще более функциональной и удобной. Интернет вещей трансформирует привычные для нас объекты в совершенно новые инструменты, позволяющие собирать, агрегировать и анализировать большое количество информации об окружающей нас действительности. Устройства с постоянным подключением к Интернету могут следить практически за всеми возможными измеримыми параметрами. Интернет вещей предоставит возможности для анализа и наблюдения за любыми показателями в реальном времени. И, что самое важное, возможна реализация функции контроля и анализа получаемых показателей, а также управления удаленными устройствами.

По оценкам аналитиков, объем подключений Интернета вещей достигнет к 2022 году огромной величины в 19 миллиардов. Так что ценность Интернета вещей будет расти в геометрической прогрессии.

Мы рассмотрели вопросы реализации идей Интернета вещей на известных и самых популярных платформах: Arduino, ESP и Raspberry Pi.

Практическая реализация полученных при прочтении книги знаний поможет вам участвовать в построении мира Интернета вещей не только в качестве потребителя, но и разработчика.

Желаю вам творческих успехов.

Виктор Петин

ПРИЛОЖЕНИЕ

Описание электронного архива

Электронный архив с материалами, сопровождающими книгу, можно скачать с FTP-сервера издательства «БХВ» по ссылке **ftp://ftp.bhv.ru/9785977567558.zip**, а также со страницы книги на сайте **https://bhv.ru**.

В архиве находятся следующие папки:

- \drivers — необходимые для устройств драйверы;
- \examples — исходники примеров *глав 3–5*;
- \projects — исходники проектов из *главы 6*;
- \libraries — библиотеки Arduino, используемые в примерах и проектах книги и не включенные в среду разработки Arduino IDE;
- \prgs — необходимые для работы программы.

Предметный указатель

А

Б

В

Г

Д

З

И

К

М

Н

О

П

www.bhv.ru

Момот М.

Мобильные роботы на базе ESP32 в среде Arduino IDE

Отдел оптовых поставок:
e-mail: opt@bhv.ru

Вы узнаете, как:

- создавать роботов на шаговых двигателях;
- управлять роботом со смартфона через Bluetooth и WiFi;
- взаимодействовать с роботом удаленно через Интернет;
- передавать изображение от робота на смартфон;
- подключить к роботу смартфон в качестве IP-камеры для видеонаблюдения;
- сконструировать балансирующего робота.

Эта книга создана как руководство для начинающих конструкторов, людей, которым нравится конструировать, энтузиастов Arduino, желающих освоить новые высокоскоростные контроллеры ESP32. Описано конструирование мобильных роботов на платформе ESP32, позволяющей реализовывать как простейшие, так и достаточно интеллектуальные устройства. А применение шаговых моторов позволяет создавать очень интересные высокоточные устройства, например робота с памятью траектории. Все проекты выполнены на единой двухколесной базе из простых в изготовлении деталей. В каждый новый проект вносятся небольшие схемные изменения и оригинальное программное наполнение, что совершенно преображает поведение роботов. Программирование ведется в традиционной среде Arduino IDE.

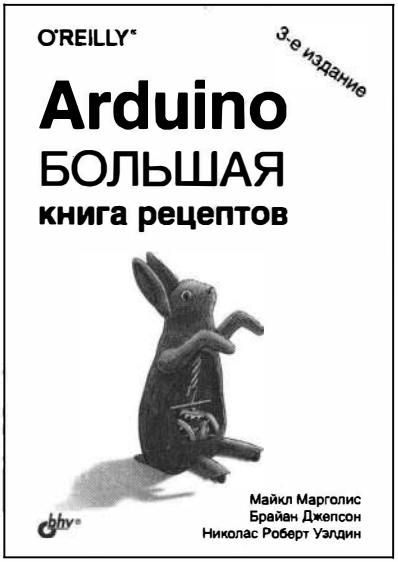
Момот Михаил Викторович, сотрудник Томского политехнического университета, основатель и директор фирмы «Юрга-Технологии-Инновации», занимающейся разработкой роботов. Увлекается робототехникой, поклонник и пропагандист проекта Arduino с 2014 года. Основатель неформального клуба робототехников «Лига роботов ЮТИ ТПУ», объединяющего школьников, студентов, преподавателей и энтузиастов. Автор книги «Мобильные роботы на базе Arduino».

www.bhv.ru

Марголис М., Джепсон Б., Уэлдин Н. Р.

Arduino. Большая книга рецептов, 3-е изд.

Отдел оптовых поставок:
e-mail: opt@bhv.ru



Книга поможет вам

- Быстро войти в курс дела по работе с платой Arduino и применению основных приемов программирования
- Изучить основные способы считывания цифровых и аналоговых сигналов
- Использовать платформу Arduino с различными популярными датчиками и другими устройствами ввода
- Выводить текст и графику на дисплеи, генерировать звуки и управлять электродвигателями разных типов
- Дистанционно управлять устройствами, включая телевизоры и бытовые приборы
- Обучиться методам измерения времени и использования пауз в исполнении кода
- Применять продвинутые методы программирования и управления памятью

Хотите создавать устройства, способные взаимодействовать с внешним миром? Этот справочник будет идеальным пособием для любого, кто хочет экспериментировать с популярной платформой Arduino. Он содержит подробное описание решений свыше 200 практических задач по созданию различных устройств и приспособлений, включая проекты для Интернета вещей, мониторинга окружающей среды, системы для определения местонахождения и положения в пространстве, а также устройств, которые реагируют на касание, звук, тепло и освещение.

Все примеры третьего издания обновлены для версии 1.8 среды Arduino IDE. Каждое решение включает в себя программный код с подробными комментариями, его анализ и обсуждение возможных проблем. Такой подход позволит вам сразу начать создавать, расширять и улучшать свои проекты, независимо от уровня технической подготовки, будь вы инженер, разработчик, деятель искусств, студент или «мейкер-самоделкин».

Майкл Марголис, работает в области вычислений реального времени и имеет свыше 30 лет опыта на руководящих должностях в компаниях Sony, Microsoft и Lucent/Bell Labs.

Брайан Джепсон, работает контент-менеджером в компании LinkedIn Learning, где руководит курсами инженерно-технического проектирования.

Николас Роберт Уэлдин, работает в британском благотворительном центре исследований и разработок Rix Centre. Проводит исследования в области мультимедийных технологий, которые помогают людям с ограниченными возможностями обучения.

Электроника

Новые возможности
Arduino, ESP, Raspberry Pi
в проектах IoT

Петин Виктор Александрович, профессиональный программист. Круг интересов: робототехника, электроника, программирование. Имеет более 50 публикаций в сетевых изданиях. Автор книг «Микрокомпьютеры Raspberry Pi. Практическое руководство», «Проекты с использованием контроллера Arduino», «Arduino и Raspberry Pi в проектах Internet of Things» и двух книг в области практического веб-программирования.

- Arduino MKR, Arduino Nano 33 IoT, ESP, Raspberry pi Zero
- Технологии Ethernet, WiFi, GPRS/GSM, BLE, LoRa для связи устройств IoT
- Протоколы IoT (HTTP, MQTT)
- Облачные IoT-платформы Arduino IoT Cloud, Narodmon, ThingSpeak, Blynk, The Things Network (TTN)
- Практические примеры IoT-проектов

Эта книга о том, как создавать проекты Интернета вещей (IoT, Internet of Things) на базе традиционных плат Arduino и новых плат Arduino серий MKR и Nano 33, плат ESP и Raspberry Pi. Показано, как организовать доступ устройств к сети Интернет и другим устройствам с помощью технологий Ethernet, WiFi, GPRS, BLE, LoRa. Рассмотрен обмен данными с облачными платформами Arduino IoT Cloud, Narodmon, ThingSpeak, Blynk b и открытой LoRaWAN-сетью The Things Network (TTN).

Большая часть книги посвящена созданию практических проектов: собственный MQTT-сервер, табло на матрице для отображения биржевых котировок в реальном времени, GPS-трекер и онлайн-сервис поиска стоянок с использованием Яндекс.Карт, сканер штрих-кода с отправкой результатов в облако, IoT-принтер для печати курсов валют, бесконтактный измеритель температуры с отправкой данных в облако, предсказатель погоды на основе данных, поступающих в сервис ThingSpeak, проекты с элементами машинного обучения на платформе TinyML и другие.

Новые возможности, новые проекты. Не опоздай!

bhv®

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru

ISBN 978-5-9775-6755-8