Дерзай!

ИЗУЧАЕМ ARDUINO

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

bhv®

УДК 004.4
ББК 32.973.26-018.2

И39

И39 Изучаем Arduino (Руководство пользователя). — СПб.: БХВ-Петербург, 2021. — 64 с.: ил.
ISBN 978-5-9775-6739-8

Руководство содержит пошаговое описание выполнения экспериментов и проектов с использованием альтернативных компонентов, не описанных в прилагаемой книге.

Для начинающих радиолюбителей

Сборка электрических схем должна осуществляться точно в соответствии с приведенными пошаговыми инструкциями. За любое другое использование или изменение комплектации набора издательство ответственности не несет.

Все схемы и программы, представленные в руководстве, были тщательно проверены и испытаны. Тем не менее ошибки в описании и программном коде не могут быть полностью исключены.

ОСТОРОЖНО, СВЕТОДИОДЫ!

Информируйте ваших детей о мерах предосторожности при обращении с яркими светодиодами! Никогда не смотрите в упор на светодиод — это может вызвать повреждение сетчатки!

Малая яркость белых, синих, фиолетовых и ультрафиолетовых светодиодов в видимом диапазоне создает ложное впечатление о фактической безопасности для ваших глаз.

Символ перечеркнутого мусорного бака означает, что этот продукт должен утилизироваться только в специальные баки отдельно от бытовых отходов. Адреса ближайших пунктов приема опасных отходов вы можете уточнить в органах местного самоуправления.

Содержание

1. Введение

Поздравляем вас с приобретением учебного набора «Изучаем Arduino. Учебный набор + КНИГА»!

В состав набора входит популярная во всем мире книга Дж. Блума «Изучаем Arduino: инструменты и методы технического волшебства» (2-е изд.), микроконтроллеры Arduino Uno и ESP32S, электронные компоненты, двигатели и макетная плата для выполнения экспериментов, а также алюминиевое шасси с колесами для сборки самоходного робота (табл. 1.1).

В процессе чтения книги вы сможете собирать электрические схемы и проводить эксперименты, чтобы облегчить освоение изучаемого материала.

Вы сможете выполнить:

- более 50 экспериментов, приведенных в табл. 1.2;
- более 30 экспериментов, представленных на сайте разработчика Arduino **http://arduino.cc/en/Tutorial/HomePage**;
- собрать самоходного робота.

Набор будет интересен и полезен для различных категорий радиолюбителей:

- *начинающие* смогут сделать свои первые шаги в написании программного кода для электронных устройств — с подключения Arduino Uno к ПК и настройки среды разработки до получения информации с аналоговых и цифровых датчиков (*главы 1–6*);
- те, кто уже получил первый положительный опыт работы с Arduino и хочет *углубить свои знания*, смогут сразу приступить к изучению более сложного материала: использованию различных шин управления и передачи данных, программных и аппаратных прерываний, чтения/записи данных на карту SD и др. (*главы 7–13*);
- для более *опытных мейкеров* рассмотрены вопросы обмена данных по каналам Bluetooth, создание Web-сервера и получение информации в Интернете (*главы 14–17*).

Надеемся, что набор поможет вам быстро адаптироваться в мире разработки электронных устройств, и вы получите удовольствие, видя, как «оживают» ваши творения!

Что вам понадобится

Во-первых, необходимо иметь доступ к сети Интернет, чтобы скачать среду разработки Arduino IDE (Integrated Development Environment)[1], а также загрузить примеры программного кода (если вы не хотите вводить его вручную) и специальные библиотеки.

Во-вторых, понадобится хорошо освещенный стол, на котором вы будете проводить эксперименты, а также расположенный вблизи рабочего места настольный ПК или ноутбук для загрузки кода в Arduino.

[1] Ссылка для закачки: **http://arduino.ru/Arduino_environment**.

Внимание!

Помните, что вы работаете с электричеством (хотя и с низкими напряжениями и небольшими токами), и поэтому поверхность стола не должна быть металлической. А если уж так случилось, обязательно накройте ее токоизолирующим материалом (например, скатертью, бумагой и т.п.).

Приготовьте блокнот и ручку для разработки концепции и дизайна проекта, вычерчивания эскизов монтажных схем.

И наконец, самое главное, без чего вам не обойтись, — это энтузиазм и желание учиться, а набор даст необходимый импульс к развитию и реализации ваших творческих идей!

Состав набора

В состав набора входят компоненты, указанные в табл. 1.1.

Таблица 1.1. *Компоненты, входящие в набор*

№	Компоненты набора	Кол-во, шт.
	Контроллеры	
1	Плата, совместимая с Arduino Uno R3	1
2	Плата ESP32S NodeMCU	1
3	Кабель USB (A–B)	1
4	Кабель USB (A — MicroUSB)	1
	Элементы коммутации	
5	Плата макетная беспаечная, 400 контактов, 8,5×5,5 см	1
6	Набор проводов 65 шт. с разъемами «папа-папа»	1
7	Провод 20 см с разъемами «папа-папа»	20
8	Провод 20 см с разъемами «папа-мама»	20
	Резисторы, потенциометры	
9	Резистор 100 Ом 1/4 Вт	10
10	Резистор 220 Ом 1/4 Вт	10
11	Резистор 1 кОм 1/4 Вт	10
12	Резистор 4,7 кОм 1/4 Вт	10
13	Резистор 10 кОм 1/4 Вт	10
14	Потенциометр 10 кОм	1
15	Подстроечный потенциометр (триммер) 10 кОм	1
	Конденсаторы	
16	Конденсатор электролитический 10 мкФ, 50 В	2
17	Конденсатор керамический 0,1 мкФ	2
	Кнопки	
18	Кнопка тактовая 6×6×5 мм	5

Таблица 1.1 *(продолжение)*

№	Компоненты набора	Кол-во, шт.
	Светодиоды, ЖК-дисплеи	
19	Светодиод цветной 5 мм	8
20	Трехцветный RGB-светодиод диаметром 5 мм	1
21	Жидкокристаллический дисплей 1602	1
22	Четырехразрядный 7-сегментный дисплей	1
	Микросхемы	
23	Линейный стабилизатор напряжения L4940V5	1
24	Диод 1N4001	1
25	Транзистор PN2222 NPN BJT	1
26	Микросхема двойного H-моста TI L293D	1
27	Триггер Шмитта инвертирующий 74AHCT14	1
28	Микросхема 8-разрядного сдвигового регистра 74HC595	1
	Датчики	
29	Датчик температуры TMP36GT9Z	1
30	Фоторезистор	1
31	ИК-дальномер Sharp GP2Y0A21YK0F	1
32	Датчик температуры TC74A0-5.0VAT	1
33	Акселерометр LIS3DH	1
	Модули	
34	Шилд Arduino Uno для чтения/записи SD-карт	1
	Звуковые устройства	
35	Динамик 8 Ом	1
36	Пьезоэлектрический зуммер	1
	Двигатели	
37	Электродвигатель постоянного тока 3–6 В	1
38	Серводвигатель TowerPro SG90 9G	1
39	Шаговый двигатель 28YBJ-48 5B	1
40	Двигатель TT130 с редуктором	2
	Питание	
41	Разъем для батареи 9 В	1
42	Батарея 9 В	1
	Инструменты, вспомогательные устройства	
43	Монтажная диэлектрическая площадка для Arduino	1
44	Отвертка	1

Таблица 1.1 *(окончание)*

№	Компоненты набора	Кол-во, шт.
	Детали мобильного робота	
45	Алюминиевое шасси мобильного робота	1
46	Колеса	2
47	Крепежные винты (комплект)	1
	Книга, руководство	
48	Изучаем Arduino: инструменты и методы технического волшебства. 2-е изд. — СПб.: БХВ-Петербург. — 544 с.	1
49	Руководство пользователя	1

Эксперименты из книги

В табл. 1.2 перечислены эксперименты из книги Дж. Блума, которые вы сможете выполнить с помощью компонентов, входящих в состав набора.

Таблица 1.2. *Эксперименты из книги Дж. Блума «Изучаем Arduino: инструменты и методы технического волшебства»*

Глава из книги	Эксперимент	Стр.	Листинг
Глава 1. Начало работы и основные сведения о платформе Arduino	**Эксперимент 1.1.** Анализируем программу Blink	49	
Глава 2. Цифровые входы и выходы и широтно-импульсная модуляция	**Эксперимент 2.1.** Включение светодиода	57	2.1
	Эксперимент 2.2. Мигание светодиодом с разной частотой	62	2.2
	Эксперимент 2.3. Изменение яркости светодиода	64	2.3
	Эксперимент 2.4. Подключение кнопки к плате Arduino	69	2.4
	Эксперимент 2.5. Переключение светодиода с функцией устранения дребезга контактов кнопки	74	2.5
	Эксперимент 2.6. Управление трехцветным светодиодом	76	2.6
Глава 3. Считывание сигналов аналоговых датчиков	**Эксперимент 3.1.** Считывание выходного аналогового сигнала потенциометра	88	3.1
	Эксперимент 3.2. Подключение датчика температуры к плате Arduino	92	3.2
	Эксперимент 3.3. Использование переменных резисторов в качестве аналоговых датчиков	97	3.3

Таблица 2.1 *(продолжение)*

Глава из книги	Эксперимент	Стр.	Листинг
Глава 4. Использование транзисторов и управление электродвигателями постоянного тока	**Эксперимент 4.1.** Автоматическое управление скоростью вращения электродвигателя	113	4.1
	Эксперимент 4.2. Ручное управление скоростью вращения электродвигателя	114	4.2
	Эксперимент 4.3. Управление скоростью и направлением вращения электродвигателя	115	4.5
	Эксперимент 4.4. Управление мобильным шасси	125	4.6
Глава 5. Управление сервоприводами и шаговыми двигателями	**Эксперимент 5.1.** Управление сервоприводом с помощью потенциометра	141	5.1
	Эксперимент 5.2. Создание сканирующего дальномера	146	5.2
	Эксперимент 5.3. Управление шаговым электродвигателем stepper.ino	150	см. рук.*
	Эксперимент 5.4. Создаем одноминутный хронограф	159	—
Глава 6. Работаем со звуком	**Эксперимент 6.1.** Воспроизведение простой мелодии с помощью платы Arduino	177	6.2
	Эксперимент 6.2. Микропианино на Arduino	181	6.3
Глава 7. Последовательный интерфейс USB	**Эксперимент 7.1.** Тестирование функций print	192	7.1
	Эксперимент 7.2. Отображение данных в табличном виде	194	7.2
	Эксперимент 7.3. Возвращение платой Arduino получаемых данных	198	7.3
	Эксперимент 7.4. Отправка одиночных символов для управления светодиодом	200	7.4
	Эксперимент 7.5. Управление разноцветным светодиодом с помощью списка значений	202	7.5
	Эксперимент 7.6. Передача данных на компьютер	208	7.6
	Эксперимент 7.7. Прием данных по последовательному каналу и изменение цвета окна	209	7.7
	Эксперимент 7.8. Управление разноцветным светодиодом на Arduino с помощью мыши, подключенной к ПК	211	7.8

Таблица 2.1 *(продолжение)*

Глава из книги	Эксперимент	Стр.	Листинг
Глава 9. Сдвиговые регистры	**Эксперимент 9.1.** Подключение к плате Arduino сдвигового регистра и восьми светодиодов	236	9.1
	Эксперимент 9.2. Эффект «бегущего» светодиода	240	9.2
	Эксперимент 9.3. Гистограмма, реагирующая на изменение входных условий	241	9.3
Глава 10. Шина I²C	**Эксперимент 10.1.** Подключение к Arduino цифрового I²C-датчика температуры	255	10.1
	Эксперимент 10.2. Использование датчика температуры, сдвигового регистра и последовательной связи	262	10.2
	Эксперимент 10.3. Вывод на экран монитора ПК значений температуры	265	10.3
Глава 11. Шина SPI и библиотеки сторонних разработчиков	**Эксперимент 11.1.** Датчик ориентации на основе акселерометра	277	11.1
	Эксперимент 11.2. Аудиовизуальный музыкальный инструмент на основе трехкоординатного акселерометра	292	11.2
Глава 12. Взаимодействие с жидкокристаллическими дисплеями	**Эксперимент 12.1.** Вывод простого текста на ЖКД	300	12.1
	Эксперимент 12.2. Вывод на дисплей специальных символов и анимация	306	12.2
	Эксперимент 12.3. Управление регулятором температуры	311	12.8
Глава 13. Прерывания и другие специальные функции	**Эксперимент 13.1.** Аппаратные прерывания	330	13.1
	Эксперимент 13.2. Прерывания по таймеру	343	13.2
	Эксперимент 13.3. Музыкальный инструмент на прерываниях	345	13.3
Глава 14. Работа с картами памяти SD	**Эксперимент 14.1.** Проверка системы SD-карты	360	14.1
	Эксперимент 14.2. Чтение и запись на SD-карту	368	14.3
	Эксперимент 14.3. Чтение и запись на SD-карту с метками RTC	374	14.4
	Эксперимент 14.4. Регистратор прохождений через дверь	384	14.5

Таблица 2.1 (окончание)

Глава из книги	Эксперимент	Стр.	Листинг
Глава 16. Беспроводная связь Bluetooth	**Эксперимент 16.1.** Передача данных аналогового датчика по каналу BTLE	427	см. рук.**
	Эксперимент 16.2. Управление светодиодом по каналу BTLE	442	—
	Эксперимент 16.3. Управление светильником посредством Bluetooth	453	—
Глава 17. Wi-Fi и облачные хранилища	**Эксперимент 17.1.** Веб-сервер на плате ESP32 с подключенными к ней RGB-светодиодом и пьезозуммером	471	—
	Эксперимент 17.2. Создание веб-сервера	480	—
	Эксперимент 17.3. Получение текущей температуры и отображения ее на дисплее	498	—

* Для экспериментов с шаговым двигателем набор укомплектован двигателем 28YBJ-48 5В вместо мощного Nema 17, описанного в книге. Схема подключения, описание и скетчи для этих экспериментов находятся в руководстве.

** Для экспериментов в главах 16 и 17 книги используются дорогие и малораспространенные в России платы Feather 32U4 Bluefruit LE (с возможностями Bluetooth) и Adafruit Feather M0 Wi-Fi w/ATWINC1500 (с возможностями Wi-Fi). В наборе их заменяет популярный контроллер ESP32, который оборудован модулями Bluetooth BLE и Wi-Fi. Описание и скетчи для этих экспериментов находятся в руководстве.

2. Описание экспериментов

2.1. Подключение RGB-светодиодов

Существуют два типа светодиодов: с общим анодом и общим катодом (рис. 2.1). Светодиоды обоих типов выполняют одинаковые функции, но принцип их работы и схема подключения различаются.

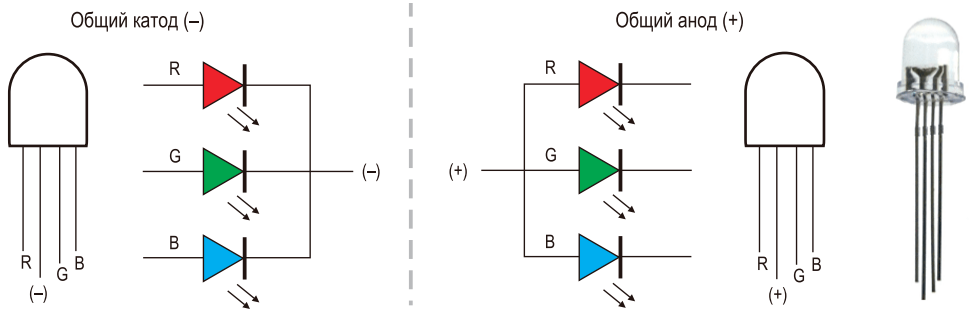


Рис 2.1. Трехцветный RGB-светодиод с общим катодом (слева) и общим анодом (справа)

В набор входит трехцветный RGB-светодиод с общим катодом. На рис. 2.2 показано подключение такого светодиода к плате Arduino. Сравните его с рис. 2.8 из книги. Как видно, длинный вывод RGB-светодиода (общий катод) подключен к земле (GND).

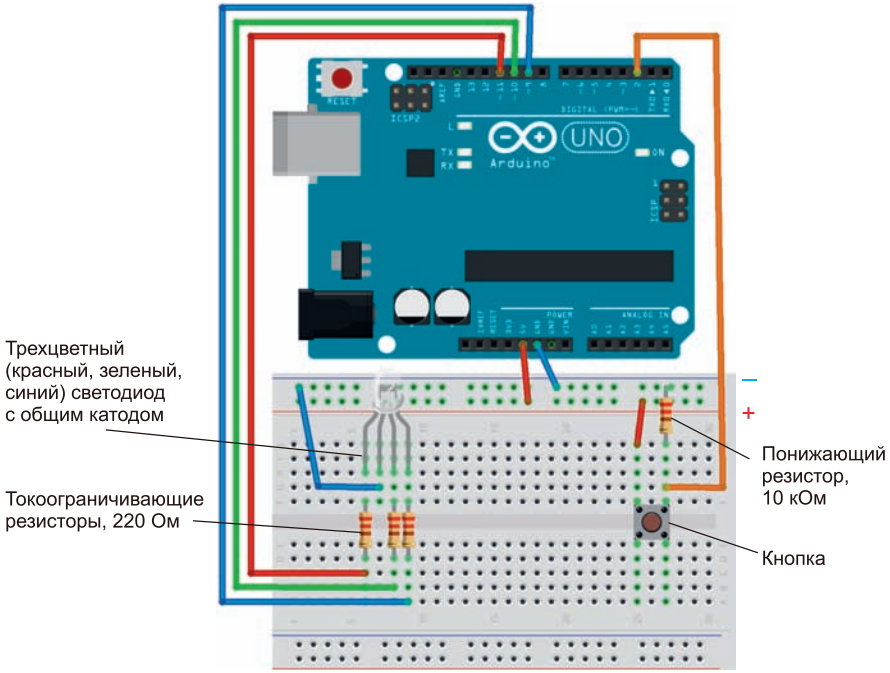


Рис. 2.2. Монтажная схема ночника на трехцветном светодиоде с общим катодом

Обратите внимание на подключение общего катода RGB-светодиода при выполнении других экспериментов (например, Эксперимент 7.5. Управление разноцветным светодиодом с помощью списка значений).

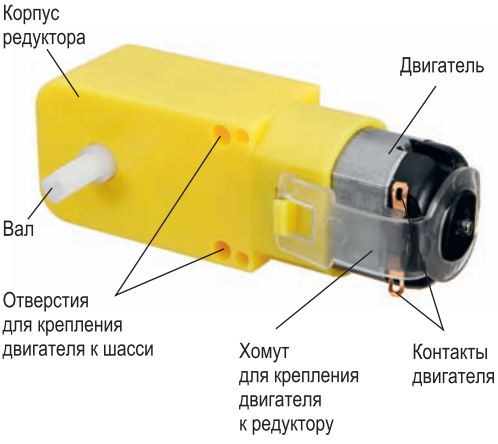
Обратите внимание на подключение общего катода RGB-светодиода при выполнении всех экспериментов с трехцветным светодиодом.

2.2. Электродвигатели постоянного тока

В экспериментах *главы 4* в книге применяется электродвигатель постоянного тока с рабочим напряжением 9 В. В набор же включен популярный в России редукторный двигатель ТТ130, рассчитанный на напряжение 3–6 В (рис. 2.3). Поэтому для его подключения следует использовать стабилизатор напряжения L7805CV (рис. 2.4).

Примечание

Принципиальная схема включения стабилизатора напряжения L7805CV показана в книге на рис. 4.9.



Напряжение питания: 3–6 В
Передаточное число: 1:48
Скорость вращения без нагрузки (3 В): 90 об./мин
Скорость вращения без нагрузки (6 В): 200 об./мин
Ток 150 мА при 120 об./мин без нагрузки (3 В)
Ток 160 мА при 250 об./мин без нагрузки (6 В)
Максимальный ток (при 6В): 670 мА
Крутящий момент (6 В): 0,8 кг/см
Вес: 30 г

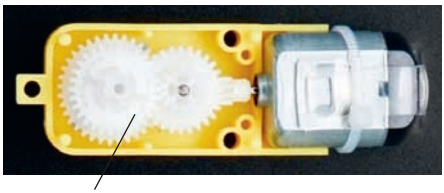


Рис. 2.3. Редукторный двигатель TT130

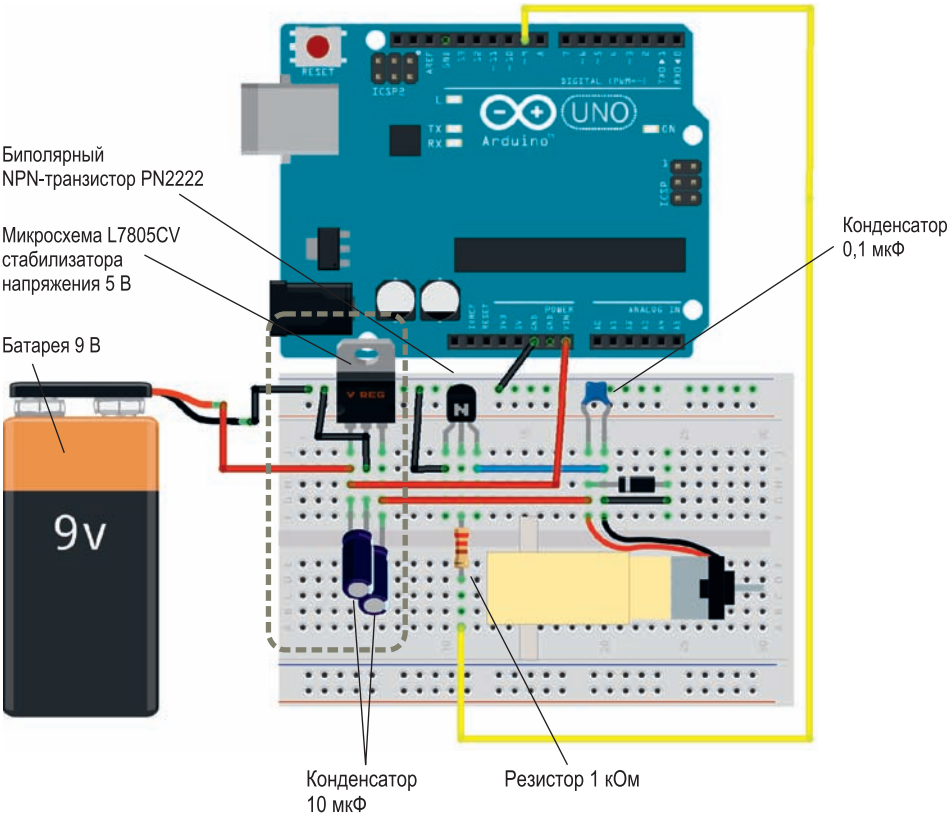


Рис. 2.4. Монтажная схема подключения электродвигателя TT130 к плате Arduino (питание от стабилизатора напряжения)

Для непродолжительных экспериментов с одним двигателем TT130 можно взять напряжение +5 В с платы Arduino, как показано на рис. 2.5 (максимальный выходной ток контакта **5V** составляет 800 мА).

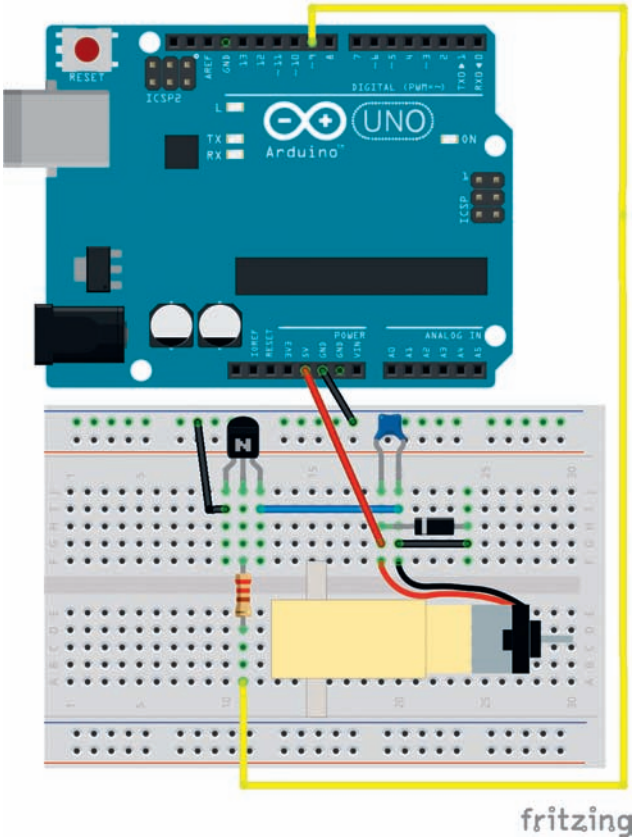
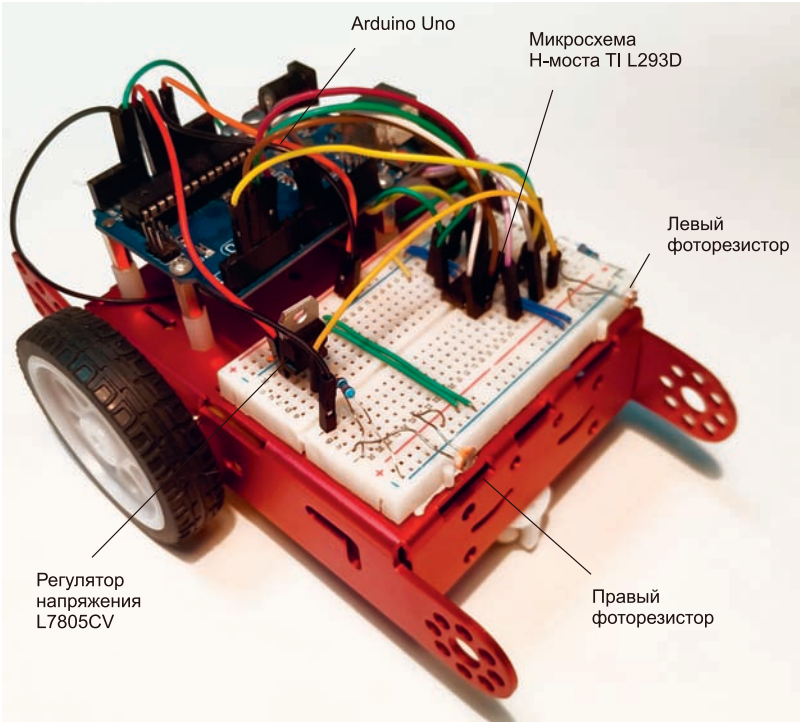


Рис. 2.5. Монтажная схема подключения электродвигателя TT130 к плате Arduino (питание от платы Arduino)

Внешний вид самоходного робота, собранного из компонентов набора, приведен на рис. 2.6.



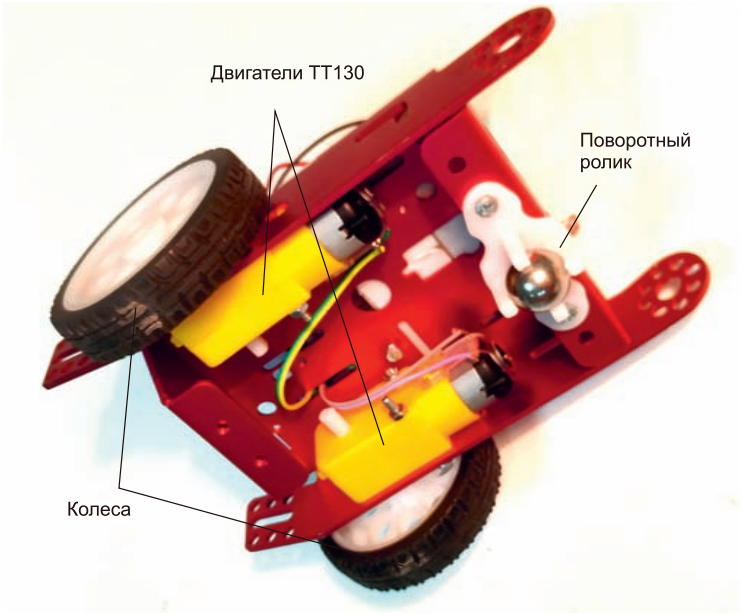


Рис. 2.6. Самоходный робот (вид сверху и снизу)

2.3. Управление шаговым электродвигателем 28YBJ-48 5B

В качестве шагового двигателя для экспериментов в *главе 5* книги рекомендован мощный и дорогой биполярный двигатель Nema 17, который широко применяется в механических системах точного позиционирования: станках с ЧПУ, принтерах (в том числе 3D) и роботах-манипуляторах.

В наборе для выполнения экспериментов прилагается двигатель 28BYJ-48, который часто используется в любительской робототехнике. Этот двигатель менее мощный и рассчитан на напряжение питания 5 В. В отличие от Nema 17, этот двигатель является униполярным (рис. 2.7).

Принцип работы биполярного двигателя показан на рис. 5.7 книги. Последовательное включение обмоток создает перемещающееся магнитное поле, которое заставляет ротор вращаться.

В схеме управления униполярным двигателем на средний вывод катушек постоянно подается напряжение (+5В для 28BYJ-48), а оставшиеся четыре вывода последовательно подключаются к общей шине. Это приводит к изменению направления протекания тока и как следствие смене полярности создаваемого магнитного поля, что в свою очередь приводит к вращению ротора двигателя.

Несмотря на то, что принципы работы двигателей различаются, вы можете в учебных целях подключить шаговый двигатель 28BYJ-48 к H-мосту TI L293D как показано на рис. 2.8.

На практике для управления шаговым двигателем 28BYJ-48 обычно используют драйвер на микросхеме ULN2003. На рис. 2.9 дана схема подключения двигателя и драйвера к плате Arduino Uno. Программа управления шаговым двигателем приведена в листинге 2.1.

На рис. 2.10 приведена схема хронографа, описанного в книге, где вместо шагового двигателя Nema 17 применяются 28BYJ-48 и драйвер ULN2003. Здесь в качестве примера показано питание шагового двигателя от внешнего источника 9 В. Программа управления хронографом приведена в листинге 2.2.

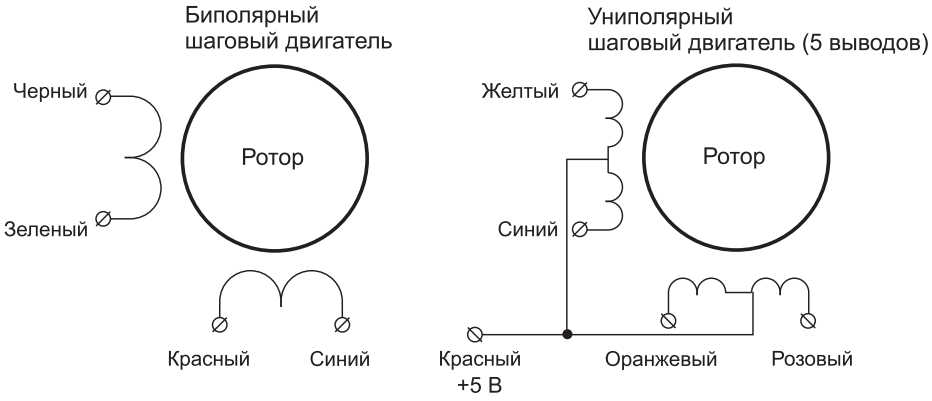


Рис. 2.7. Схемы биполярного и униполярного двигателей

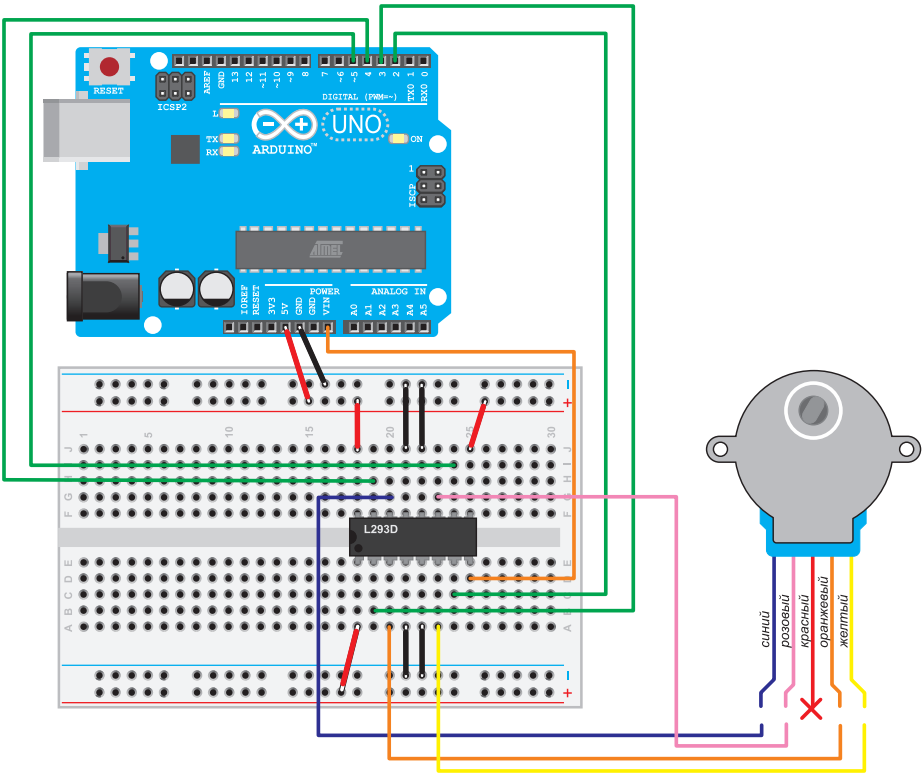


Рис. 2.8. Подключение шагового двигателя 28BYJ-48 к H-мосту TI L293D

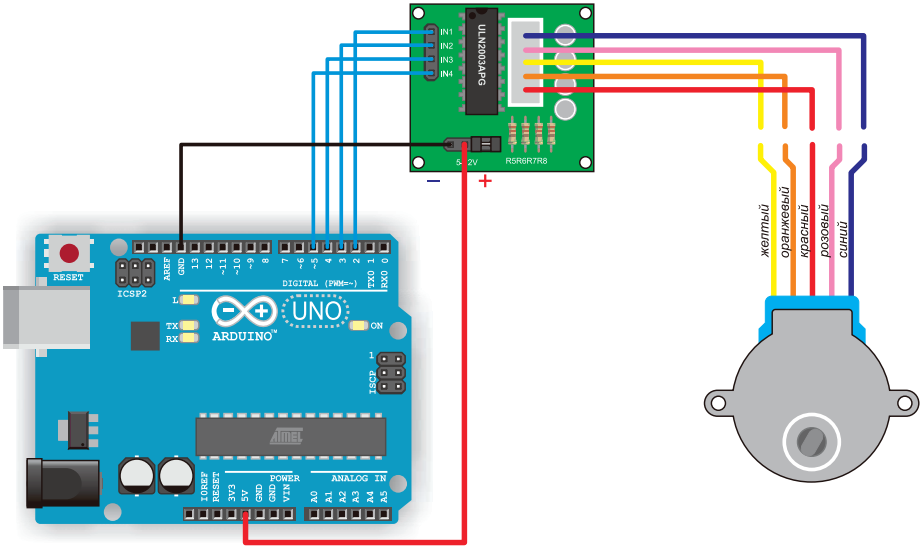


Рис. 2.9. Управление шаговым электродвигателем 28BYJ-48 с помощью драйвера ULN2003

Листинг 2.1. Простая программа для управления шаговым электродвигателем 28BYJ-48 (28BYJ-48-stepper.ino)

```
#include <Stepper.h>
// Шаговые электродвигатели 28BYJ-48 в режиме полного шага делают
// за один полный оборот вала 2048 шагов
const int STEPS_PER_REV = 2048; // 2048 шагов на оборот

// Создаем экземпляр класса Stepper и указываем
// количество шагов и контакты для подключения
  Stepper myStepper(STEPS_PER_REV, 2, 3, 4, 5);

void setup() {
  // Устанавливаем скорость вращения в об./мин
   myStepper.setSpeed(1);
}
void loop() {
 // Поворот вала на один шаг в одном направлении
   myStepper.step(STEPS_PER_REV);
    delay(500);
// Поворот вала на один шаг в другом направлении
   myStepper.step(-STEPS_PER_REV);
    delay(500);
}
```

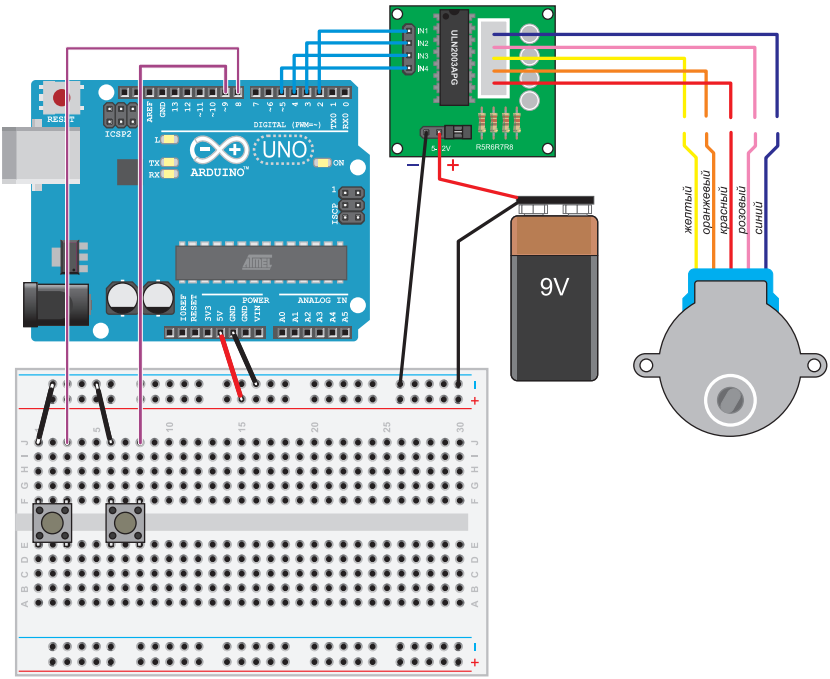


Рис. 2.10. Монтажная схема хронографа

Листинг 2.2. Программа одноминутного хронографа 28BYJ-48-crhonograph.ino (аналог листинга 5.7 из книги)

```
#include <Stepper.h>
// Шаговые электродвигатели 28BYJ-48 в режиме полного шага делают
// за один полный оборот вала 2048 шагов
const int STEPS_PER_REV = 2048; // 2048 шага/оборот
// Чтобы выполнить один оборот вала за одну минуту,
// вычисляем количество миллисекунд на каждый шаг:
// 60 секунд * 1000 мс / 2048 шагов = 30 мс/шаг
const int MS_PER_STEP = 30;
// Контакты для управления драйвером
const int COIL1_MC1 = 2;   // Контакт Arduino для подключения к контакту IN1
                           // драйвера ULN2003
const int COIL1_MC2 = 3;   // Контакт Arduino для подключения к контакту IN2
                           // драйвера ULN2003
const int COIL2_MC1 = 4;   // Контакт Arduino для подключения к контакту IN3
                           // драйвера ULN2003
const int COIL2_MC2 = 5;   // Контакт Arduino для подключения к контакту IN3
                           //драйвера ULN2003
// Контакты Arduino для считывания сигнала с кнопок
const int START = 8; // Кнопка Start
const int STOP = 9;  // Кнопка Stop
// Переменные для отслеживания
unsigned long last_time = 0;
unsigned long curr_time = 0;
int steps_taken = 0;
// Инициализируем библиотеку Servo.h — передаем ей контакты
// Arduino для управления переключателями H-моста
Stepper chronograph(STEPS_PER_REV, COIL1_MC1, COIL1_MC2,COIL2_MC1, COIL2_MC2);
void setup()
{
// Задаем высокую скорость вращения вала шагового двигателя
// для быстрого выполнения каждого шага
chronograph.setSpeed(1); // Включаем встроенные повышающие резисторы для контактов
// для снятия сигнала кнопок
pinMode(START, INPUT_PULLUP);
pinMode(STOP, INPUT_PULLUP);
}
void loop()
{
// Выполняем цикл, ожидая нажатия кнопки запуска Start
// Оператор цикла задерживает ход исполнения
// в этом месте, пока удовлетворяется указанное условие.
while(digitalRead(START) == HIGH);
```

```
last_time = millis(); // Получаем время при запуске
// Исполняем этот цикл, пока не истечет одна минута или не будет нажата кнопка Stop
while(digitalRead(STOP) == HIGH && steps_taken < STEPS_PER_REV)
{
curr_time = millis();
// Если прошло 300 мс, вращаем вал на один шаг
if(curr_time - last_time >= MS_PER_STEP)
{
chronograph.step(1);    // Вращаем вал на одни шаг
steps_taken++;          // Инкрементируем переменную счетчика шагов steps_taken
last_time=curr_time;    // Обновляем переменную last_time текущим временем
}
}
// Если мы в этой точке, значит, была нажата кнопка Stop или истекла одна минута
// Если не выполнен полный оборот, возвращаем стрелку в исходную точку.
if (steps_taken < STEPS_PER_REV) chronograph.step(-steps_taken);
// Сброс счетчика шагов
steps_taken = 0;
}
```

2.4. Шина SPI

В *главе 11* книги рассмотрена работа с шиной SPI на примере использования акселерометра LIS3DH на адаптерной плате компании Adafruit. В наборе имеется микросхема LIS3DSH, которая выполняет аналогичные функции и также поддерживает протокол SPI.

Для подключения акселерометра LIS3DSH воспользуемся библиотекой SPI, встроенной в среду разработки Arduino IDE. Соберите схему, показанную на рис. 2.11, и загрузите скетч листинга 2.3.

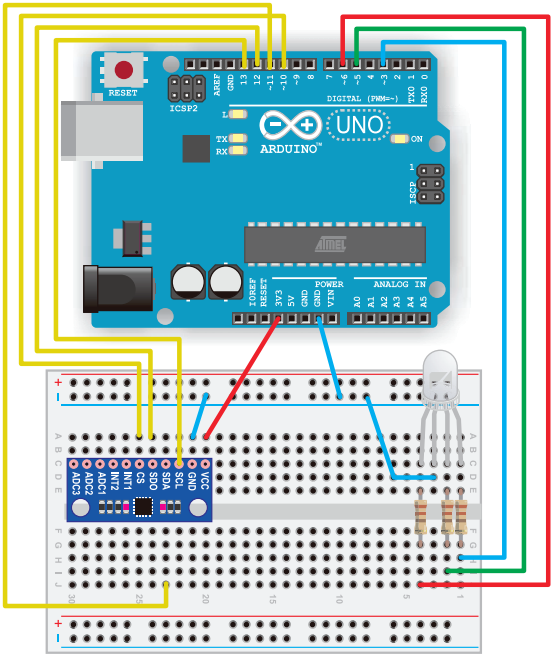


Рис. 2.11. Монтажная схема подключения акселерометра LIS3DSH и RGB-светодиода к плате Arduino Uno

Листинг 2.3. Программа orientation.ino для датчика ориентации на основе акселерометра LIS3DSH (аналог листинга 11.1 из книги)

```
// Программа определяет ориентацию устройства, используя показания оси Z
// акселерометра LIS3DSH
#include <SPI.h>

// Определяем назначение контактов платы Arduino
  const int RED_PIN = 6;
  const int GREEN_PIN = 5;
  const int CS_Pin = 10;

// Контакты Arduino аппаратного интерфейса SPI
// заданы по умолчанию
// pin 11 as MOSI (SDI),
// pin 12 as MISO (SDO) ,
// pin 13 as clock (SPC)

// Определение скорости, порядка отправки битов информации и режима передачи данных
SPISettings settingsA(2000000, MSBFIRST, SPI_MODE3);

int x,y,z;
// (4000/65535) milli-g per digit for +/-2g full scale using 16 bit digital output
float K=0.061;

void setup() {

  Serial.begin(9600);
  pinMode (CS_Pin, OUTPUT);  // Выбор ведомого устройства (регистра) SPI

  digitalWrite(CS_Pin, HIGH);    // Конец передачи по шине SPI
  SPI.begin();                   // Инициализация интерфейса SPI
  SPI.beginTransaction(settingsA);

  digitalWrite(CS_Pin, LOW);     // Начало передачи данных по шине SPI
  SPI.transfer(0x20);            // Обращение к регистру 'Control register 4'
  SPI.transfer(0x7F);            // Настройка его  для записи значений x, y, z акселерометра
  digitalWrite(CS_Pin, HIGH);    // Конец передачи по шине SPI

 // Задаем выходной режим работы для контактов светодиодов
   pinMode(RED_PIN, OUTPUT);
   digitalWrite(RED_PIN, HIGH);
   pinMode(GREEN_PIN, OUTPUT);
   digitalWrite(GREEN_PIN, HIGH);
```

```
}
void loop() {
  delay(1000);
  digitalWrite(CS_Pin, LOW);                // Начало передачи данных по шине SPI
  SPI.transfer(0xA8);                       // Принимаем байт по шине SPI
  x = SPI.transfer(0) | SPI.transfer(0)<<8; // x acceleration
  y = SPI.transfer(0) | SPI.transfer(0)<<8; // y acceleration
  z = SPI.transfer(0) | SPI.transfer(0)<<8; // z acceleration
  digitalWrite(CS_Pin, HIGH);               // Конец передачи по шине SPI
  Serial.println("x=" + String(K*x)+" mg \ty=" + String(K*y)+" mg  \tz=" + String(K*z)+" mg");

  // Проверяем, "вверх ногами" устройство или нет
  if (z < 0){
  digitalWrite(RED_PIN, LOW);
  digitalWrite(GREEN_PIN, HIGH);
  }
  else {
   digitalWrite(RED_PIN, HIGH);
   digitalWrite(GREEN_PIN, LOW);
  }
// Получаем новые данные каждые 100 мс
delay(100);
}
```

2.5. Радиосвязь Bluetooth

В проектах *главы 16* книги Дж. Блума «Изучаем Arduino…» описано применение платы Feather 32U4 Bluefruit LE, разработанной американской компанией Adafruit. Эта плата не получила широкого распространения в России, но на рынке предлагается много других устройств BTLE/BLE (BlueTooth LE — беспроводная технология Bluetooth с низким энергопотреблением), совместимых с Arduino.

Популярная и недорогая альтернатива — плата ESP32 от компании Espressif. Эта плата кроме Bluetooth поддерживает Wi-Fi, а также оборудована датчиками прикосновения и температуры. (Подробное описание платы ESP32 и способ ее подключения к Arduino IDE приведены в Приложении к руководству.)

Чип ESP32 может работать как в режиме «классического» Bluetooth BR/EDR (Basic Rate/Enhanced Data Rate), называемого также Bluetooth Classic, так и в экономичном режиме BLE, еще называемом Bluetooth Smart (табл. 2.1). «Классический» режим проще в программной реализации. Далее приведены эксперименты из книги, адаптированные как для Bluetooth Classic, так и для Bluetooth BLE.

Таблица 2.1. *Сравнение Bluetooth BR/EDR и Bluetooth BLE/BLTE*

Характеристика	Bluetooth BR/EDR	Bluetooth BLE/BLTE
Диапазон частот, ГГц	2,402–2,480	2,402–2,480
Потребляемая мощность, Вт	1	0,01–0,5
Применение	Используется для передачи файлов и других данных	Отлично подходит для маяков, смарт-часов, фитнес-трекеров и т.д.
Протокол	1.0, 2.*x*, 3.0	4.0, 5.0
Топологии сети	Точка-точка (включая пикосеть)	Точка-точка (включая пикосеть) Broadcast Mesh

2.5.1. Применение Bluetooth BR/EDR платы ESP32

Сначала рассмотрим особенности подключения и применение «классического» Bluetooth BR/EDR.

Установка связи между платой ESP32 и смартфоном

Прежде всего подключите плату ESP32 к порту USB ПК и настройте среду Arduino IDE для работы с ESP32, как описано в Приложении к руководству (см. раздел «Установка и настройка Arduino IDE для работы с ESP32»).

Мы будем передавать текстовые сообщения от смартфона на плату ESP32 и обратно по каналу Bluetooth BR (рис. 2.12).

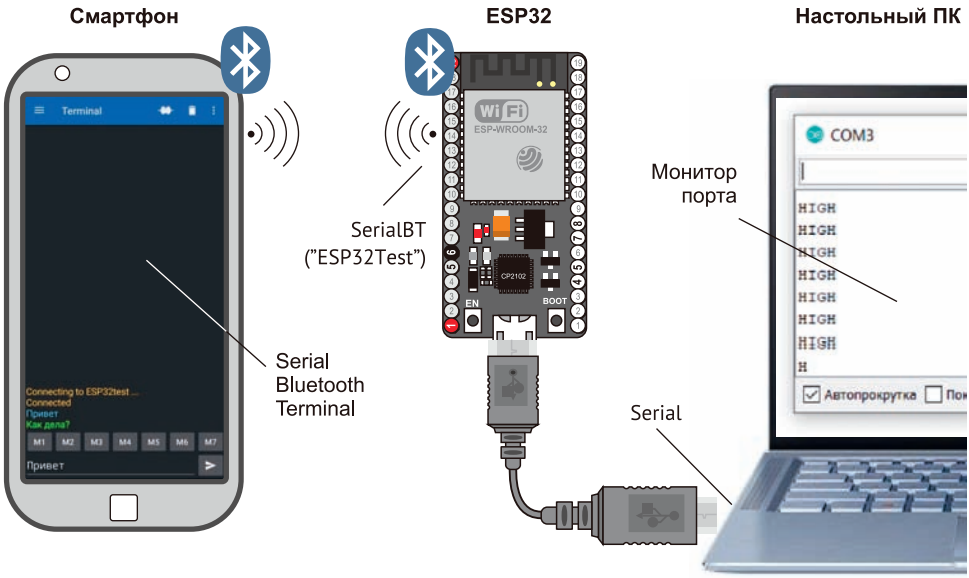
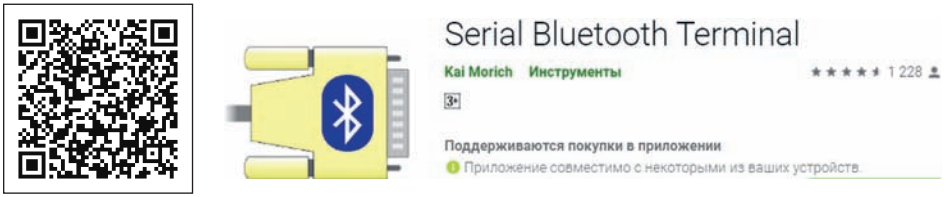


Рис. 2.12. Обмен информацией между ESP32 и смартфоном

Для взаимодействия между смартфоном и модулем ESP32 по каналу Bluetooth BR/EDR выполните следующие действия.

1. Установите на смартфон терминальную программу для связи с модулем Bluetooth, например **Serial Bluetooth Terminal**.



2. Откройте Arduino IDE и выберите учебный пример SerialtoSerialBT (**Файл | Примеры | BluetoothSerial**). Этот скетч устанавливает двустороннюю последовательную связь по каналу Bluetooth между двумя устройствами (рис. 2.13). Его код приведен в листинге 2.4.

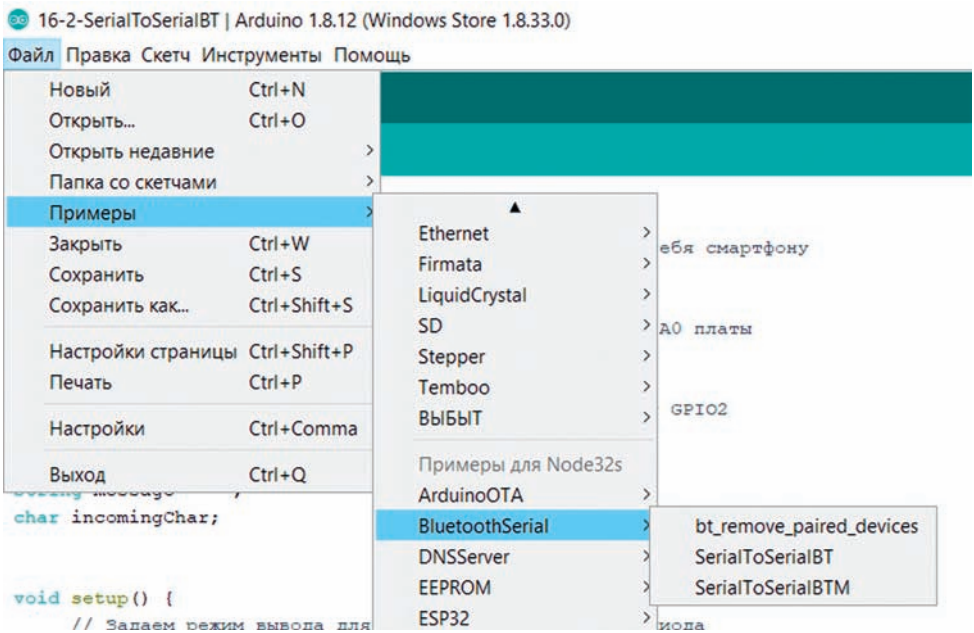


Рис. 2.13. Открываем скетч SerialtoSerialBT

Листинг 2.4. Подключение смартфона к ESP32 по каналу BT BR

#include "BluetoothSerial.h"	Подключение библиотеки BluetoothSerial
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED) #error Bluetooth is not enabled! Please run `make menuconfig' to and enable it #endif	Проверка включения Bluetooth
BluetoothSerial SerialBT;	Создание экземпляра BluetoothSerial с именем SerialBT
void setup() {	
Serial.begin(115200);	Инициализация последовательной связи со скоростью 115 200 бод
SerialBT.begin("ESP32test"); //Bluetooth device name	Инициализация последовательного модуля Bluetooth на плате ESP32 и передача в качестве аргумента имени устройства Bluetooth. В примере указано имя ESP32test, но вы можете его изменить
Serial.println("The device started, now you can pair it with bluetooth!");	Сообщение о готовности устройства к работе
}	
void loop() {	Начало цикла

if (Serial.available()) {
SerialBT.write(Serial.read());
}
	В первом операторе if мы проверяем, получены ли байты в последовательном порту. Если есть, то передаем эту информацию через Bluetooth на подключенное устройство.
	Оператор SerialBT.write () отправляет данные, используя последовательный интерфейс Bluetooth, а оператор Serial.read () возвращает данные, полученные через последовательный порт.

if (SerialBT.available()) {
Serial.write(SerialBT.read());
}
	Следующий оператор if проверяет, доступны ли байты для чтения через последовательный порт Bluetooth. Если они есть, мы запишем эти байты в Монитор порта

delay(20);	Задержка 20 мс
}	

3. Загрузите код в плату ESP32. Убедитесь, что вы выбрали правильную плату и COM-порт. После загрузки кода откройте последовательный **Монитор порта** со скоростью 115 200 бод. Нажмите кнопку **EN** на плате ESP32. Через несколько секунд вы получите сообщение: *“The device started, now you can pair it with bluetooth!”*» (рис. 2.14).

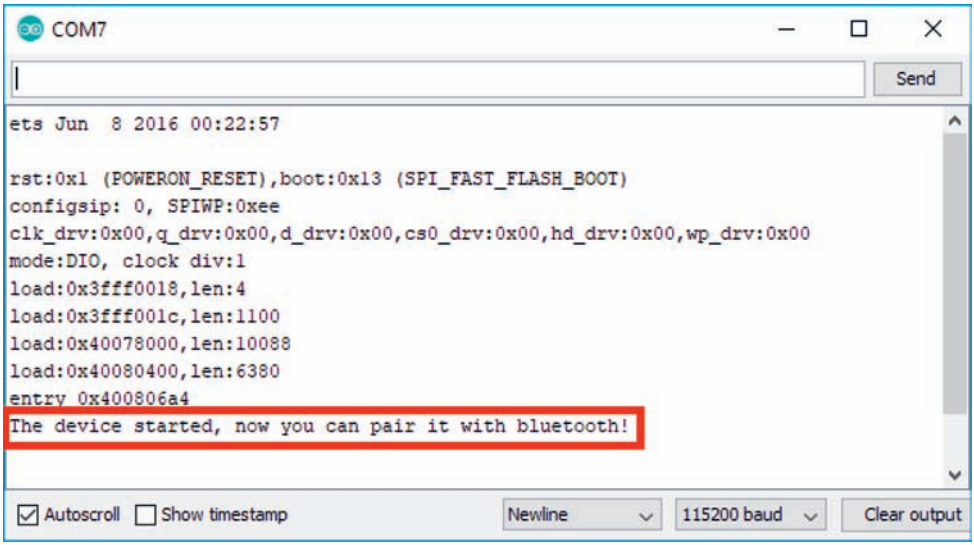
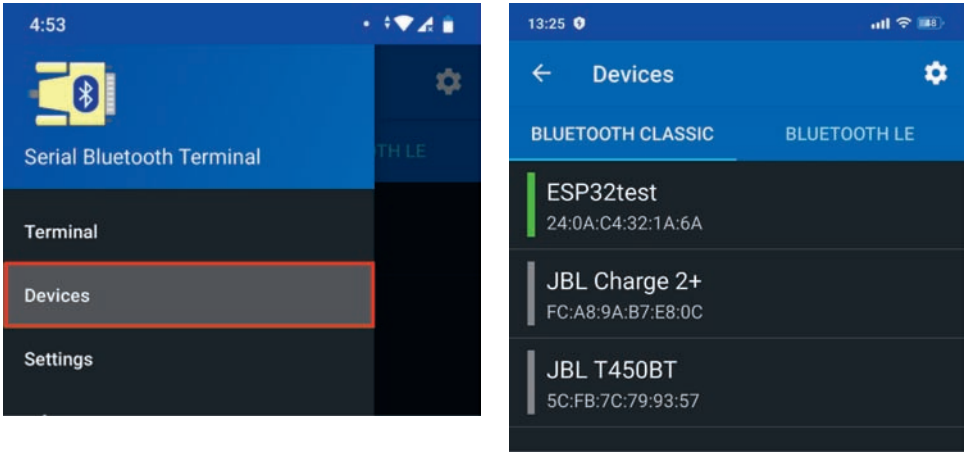


Рис. 2.14. Устройство (ESP32) запущено, теперь его можно подключить к Bluetooth (смартфона)

4. Откройте на смартфоне приложение **Serial Bluetooth Terminal**. Перейдите к разделу **Devices**, и на вкладке **Bluetooth Classic** вы увидите обнаруженные поблизости устройства (рис. 2.15).



а *б*

Рис. 2.15. Обнаруженные на смартфоне устройства Bluetooth

Если среди обнаруженных устройств нет нашего модуля **ESP32test**, перейдите в настройки смартфона, проверьте включен ли Bluetooth, обновите список доступных Bluetooth-устройств и выберите сопряжение с **ESP32test** (рис. 2.16).

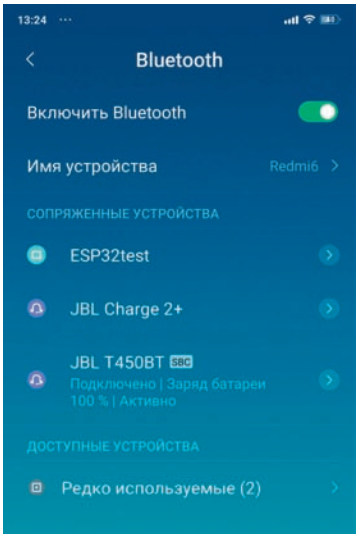


Рис. 2.16. Выберите обнаруженное устройства ESP32test

5. Затем вернитесь к приложению **Serial Bluetooth Terminal**. Нажмите на иконку вверху, чтобы подключиться к ESP32. Вы должны получить сообщение Connected (рис. 2.17).

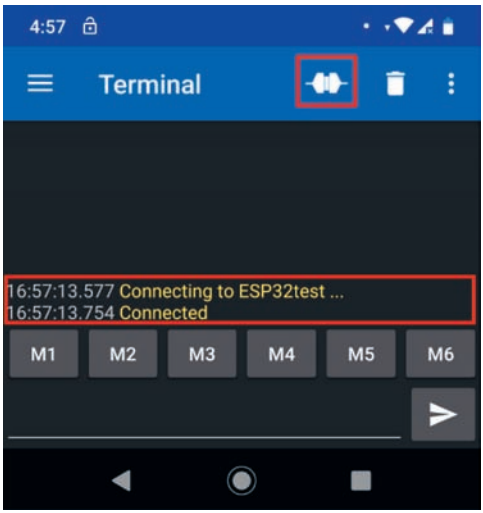


Рис. 2.17. ESP32 и смартфон сопряжены

6. После этого введите что-нибудь в приложении **Serial Bluetooth Terminal**, например, «Привет». Вы должны немедленно получить это сообщение в окне **Монитора порта** Arduino IDE (рис. 2.18).

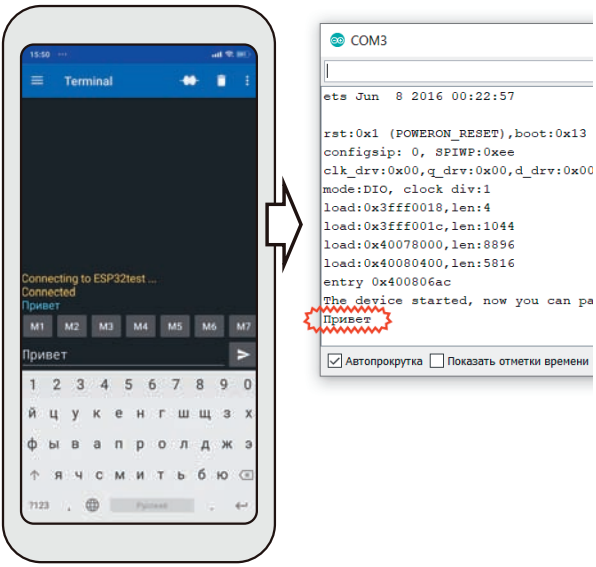


Рис. 2.18. Переданное из Терминала сообщение отобразилось в **Мониторе порта**

Вы также можете обмениваться данными между вашим **Монитором порта** и смартфоном. Напечатайте что-нибудь в верхней панели **Монитора порта** и нажмите кнопку **Отправить**. Вы должны немедленно получить это сообщение в приложении **Serial Bluetooth Terminal** (рис. 2.19).

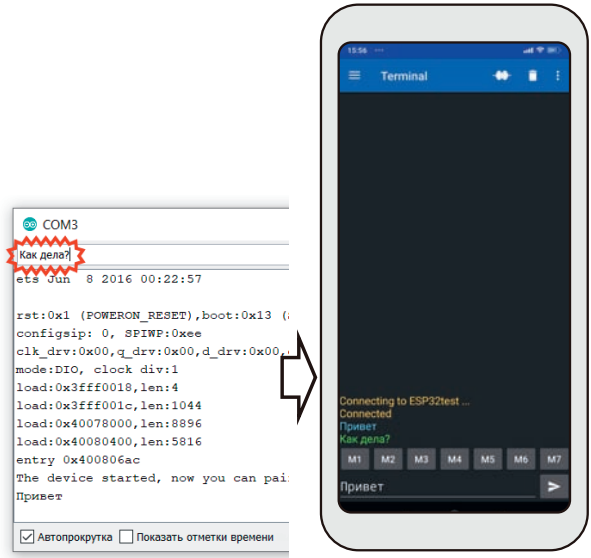


Рис. 2.19. Переданное из **Монитора порта** сообщение отобразилось в Терминале

Передача данных аналогового датчика на смартфон по каналу Bluetooth BR

Рассмотрим подключение и получение данных от самого простого аналогового датчика — потенциометра (см. Эксперимент 16.1). Подключите его к плате ESP32, как показано на монтажной схеме на рис. 2.20. Внешние контакты потенциометра подключаются к шинам **3,3 В** и **GND** макетной платы, а средний контакт — к контакту A0 (GPIO36) платы ESP32. Мы запрограммируем нашу плату передавать значение выходного сигнала этого потенциометра по каналу Bluetooth на **Монитор порта** (листинг 2.5).

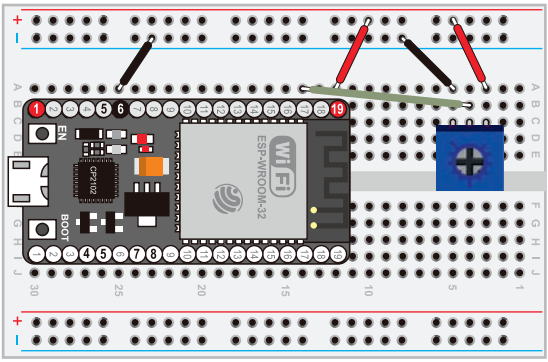


Рис. 2.20. Подключение потенциометра

Листинг 2.5. Программа BTBR_sensor.ino для передачи данных аналогового датчика по каналу BTBR (аналог листинга 16.5 из книги, адаптированный для ESP32)

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run 'make menuconfig' to and enable it
#endif
BluetoothSerial SerialBT;
// Имя, под которым устройство BTLE будет представлять себя смартфону
const String BTLE_NAME = "Jeremy's Sensor";
// Выходной контакт потенциометра подключаем к контакту A0 платы
const int POT = A0;
// Встроенный светодиод NodeMCU-32S подключен к контакту GPIO2
const int STATUS_LED = 2;

void setup() {
// Задаем режим вывода для контакта красного светодиода
  pinMode(STATUS_LED, OUTPUT);
// Задаем режим ввода для контакта потенциометра
  pinMode(POT, INPUT);
 // Будем выводить информацию для отладки в окно монитора порта.
```

```
  Serial.begin(115200);
  SerialBT.begin(BTLE_NAME);        // Имя устройства Bluetooth
  Serial.println("The device started, now you can pair it with bluetooth!");
}
void loop() {
// Считываем значение выходного сигнала потенциометра
   int val = analogRead(POT);
// Выводим полученное значение в окно монитора порта
   Serial.print(F("Analog Value: "));   // Аналоговое значение:
   Serial.print(val);
   Serial.println(F("\tSending..."));

// Отправляем данные модулю BTLE для дальнейшей передачи по радиоканалу
// Мигаем светодиодом в процессе передачи
   digitalWrite(STATUS_LED, HIGH);
   SerialBT.println(val);

// Ждем подтверждения получения данных модулем BT
  if (SerialBT.available()) {
  Serial.write(SerialBT.read());
  Serial.println("OK!");
   digitalWrite(STATUS_LED, LOW);
 }
  SerialBT.println(val);
  delay(20);
}
```

Передача данных со смартфона на ESP32 по каналу BTBR

Теперь, когда мы знаем, как передавать данные по радиоканалу BTBR с платы ESP32 на смартфон, рассмотрим, как передавать данные обратно со смартфона на плату ESP32 (см. Эксперимент 16.2). Для демонстрации этой возможности проделайте следующее:

1. Подключите светодиод к контакту 10 платы ESP32, не забывая при этом включить последовательно токоограничивающий резистор номиналом 220 или 150 Ом (рис. 2.21).
2. Загрузите в плату ESP32 код из листинга 2.6.
3. Запустите на смартфоне приложение **Serial Bluetooth Terminal** и подключитесь к плате, как делали ранее.
4. Теперь вы можете вводить на смартфоне различные фразы, указанные в табл. 16.1 книги, и светодиод будет реагировать на ваши команды.

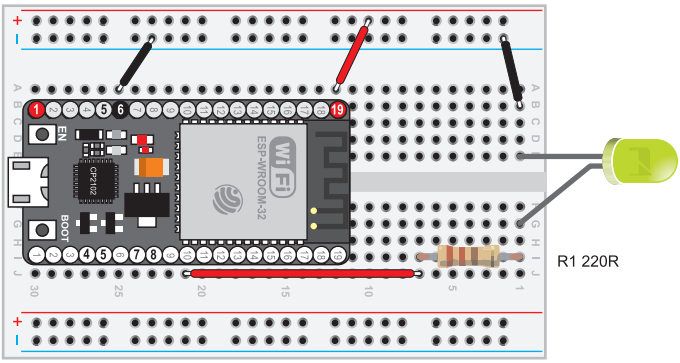


Рис. 2.21. Подключение светодиода

Листинг 2.6. Программа BTBR_led.ino для управления светодиодом по каналу BTBR

```
#include "BluetoothSerial.h"
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run 'make menuconfig' to and enable it
#endif
BluetoothSerial SerialBT;
// Имя, под которым устройство BTLE будет представлять себя смартфону
const String BTLE_NAME = "Jeremy's LED";
// Встроенный светодиод NodeMCU-32S подключен к контакту GPIO2
const int STATUS_LED =2;
// Управляемый светодиод подключен к контакту 5
const int CTRL_LED = 5;
// Переменные для отслеживания состояния светодиода
bool led_state = LOW;
String cmd = "";
String reply = "";

void setup() {
// Задаем режим вывода для контакта красного светодиода
   pinMode(STATUS_LED, OUTPUT);
   pinMode(CTRL_LED, OUTPUT);
// Будем выводить информацию для отладки в окно монитора порта.
 Serial.begin(115200);
 SerialBT.begin(BTLE_NAME); //имя Bluetooth
 Serial.println("The device started, now you can pair it with bluetooth!");
}
void loop() {
// Если есть входящие данные, считываем их и выполняем анализ.
  if (SerialBT.available()){
    char incomingChar = SerialBT.read();
 // Считываем символы во входящем буфере, пока не обнаружим символ новой строки.
```

```
  if (incomingChar != '\n'){
    cmd += String(incomingChar);
  }
  else{
    cmd = "";
  }
  Serial.write(incomingChar);
 }
 // Преобразовываем все буквы строки в строчные, чтобы упростить распознавание
 // команды
 cmd.toLowerCase();
 // Ищем в строке ключевое слово red или led и продолжаем анализ строк, содержащих их
 if (cmd.indexOf(F("red")) != -1 || cmd.indexOf(F("led")) != -1){
 // Строка содержит ключевое слово on
    if (cmd.indexOf(F("on")) != -1){
      led_state = HIGH;
      reply = F("OK! The LED has been turned on.");
      //OK! Светодиод включен
    }
    // Строка содержит ключевое слово off
    else if (cmd.indexOf(F("off")) != -1) {
      led_state = LOW;
      reply = F("OK! The LED has been turned off.");
   // OK! Светодиод выключен
    }
    // Строка содержит ключевое слово toggle
    else if (cmd.indexOf(F("toggle")) != -1) {
      led_state = !led_state;
      if (led_state) reply = F("OK! The LED has been toggled on.");
   // OK! Светодиод перключен
      else reply = F("OK! The LED has been toggled off.");
    }
    // Строка содержит ключевое слово red или led, но никаких других ключевых слов
    else{
      if (led_state) reply = F("The LED is currently on.");
   // Светодиод включен
      else reply = F("The LED is currently off.");
   // Светодиод выключен
    }
    // Set the LED state
    Serial.println(reply);
    digitalWrite(CTRL_LED, led_state);
 }
 else {
    reply = F("Command not understood."); // Неизвестная команда
 }
 delay(20);
}
```

2.5.2. Применение Bluetooth BLE платы ESP32

Технология BLE предполагает использование двух типов устройств — сервера и клиента. Плата ESP32 может действовать как клиент или как сервер.

Сервер объявляет о своем существовании, чтобы его могли найти другие устройства, и содержит данные, которые клиент может прочитать. Клиент сканирует близлежащие устройства и, когда находит сервер, который ищет, он устанавливает соединение и прослушивает входящие данные. Это называется двухточечным общением (рис. 2.22).

Рис. 2.22. Взаимодействие сервера и клиента

Плата ESP32 может выступать как в качестве сервера BLE, так и в качестве клиента BLE. В библиотеке BSP ESP32 для Arduino IDE есть несколько примеров BLE для ESP3 (рис. 2.23). Эта библиотека устанавливается по умолчанию при установке ESP32 в Arduino IDE.

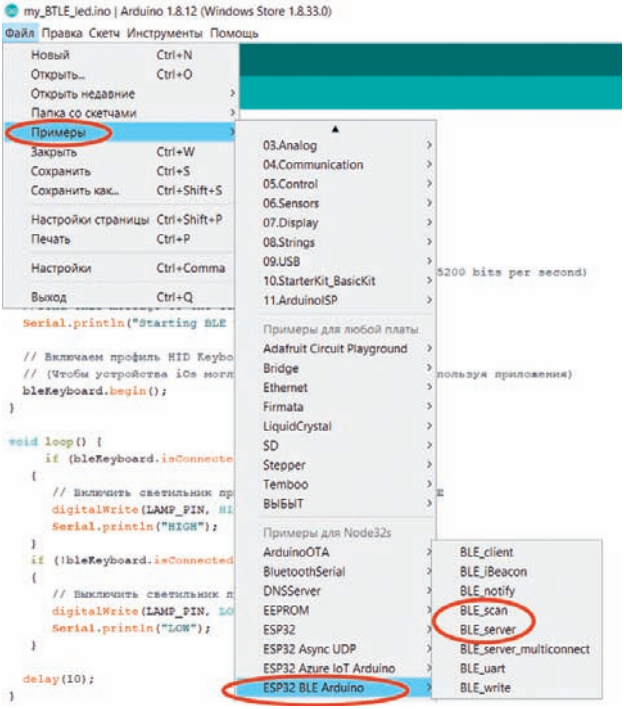


Рис. 2.23. Примеры BLE для ESP3

Вы можете создать веб-сервер (BLE_server) и веб-клиент (BLE_client, BLE_scan) на базе ESP32. Подробно можно прочитать здесь:

http://wikihandbk.com/wiki/ESP32:Примеры

https://iotbyhvm.ooo/esp32-ble-with-dht11/

https://iotbyhvm.ooo/esp32-bluetooth-low-energy-ble-on-arduino-ide/

https://www.instructables.com/id/ESP32-BLE-Android-App-Arduino-IDE-AWESOME/

https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/

Рассмотрим проект, объединяющий в себе эксперименты, которые были рассмотрены ранее с классическим Bluetooth BR/EBR. Будем передавать на смартфон/iPhone показания потенциометра и включать/выключать светодиод по команде с телефона.

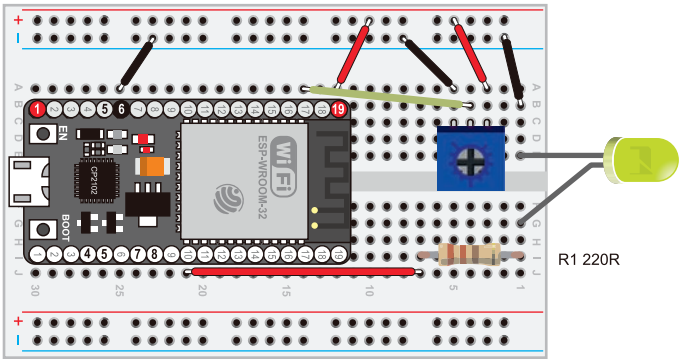
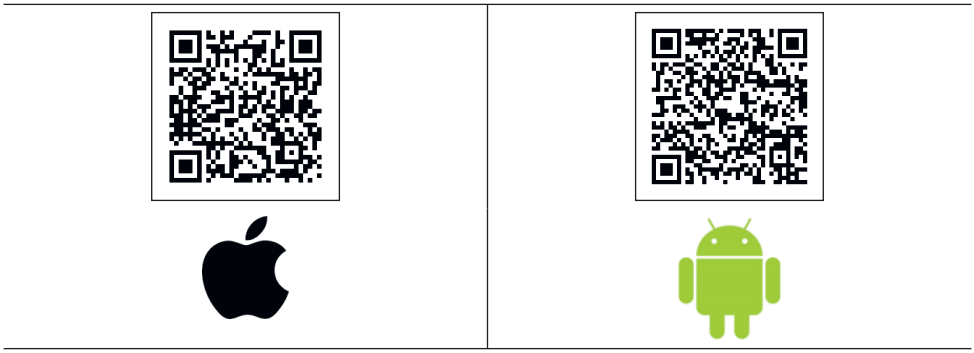


Рис. 2.24. Подключение светодиода и резистора к ESP32

1. Соберите монтажную схему, показанную на рис. 2.24.
2. Скачайте на мобильный телефон приложение **nRF Connect for Mobile** для сопряжения мобильного телефона с внешним BLE-устройством.



3. Загрузите из электронного архива скетч, приведенный в листинге 2.7. Скетч сопровождается подробными комментариями, и вы сможете по аналогии подключить впоследствии к плате другие датчики и внешние устройства.

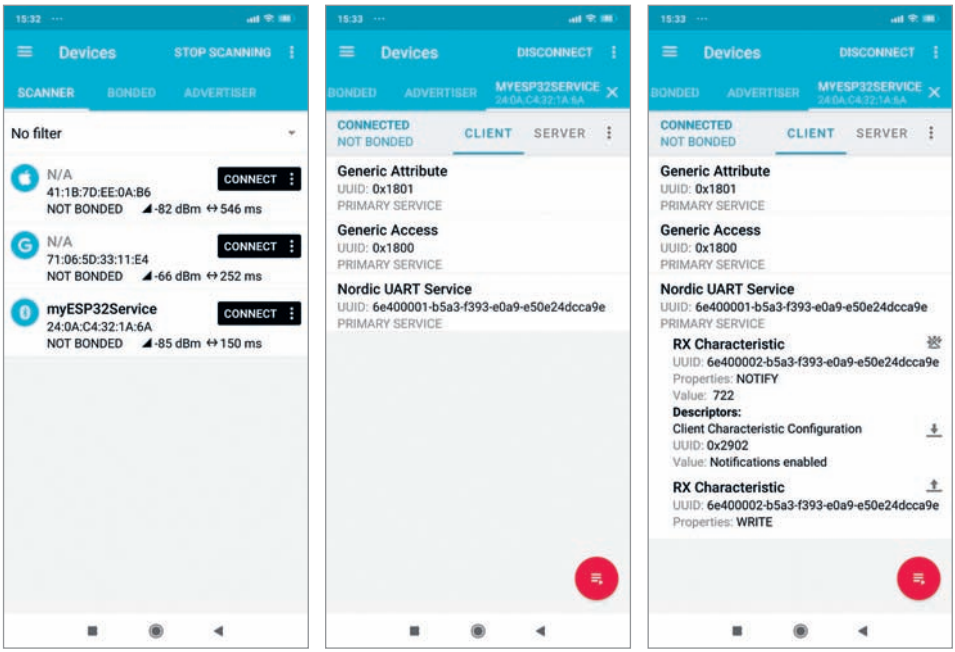
Листинг 2.7. Программа BLE_sensor_led.ino для управления светодиодом и получения данных с аналогового датчика по каналу BLE

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include <iostream>
#include <string>

 BLECharacteristic * pCharacteristic;
 BLEServer *pServer = NULL;
 bool deviceConnected = false;

// Светодиод подключен к контакту GPIO5
 const int STATUS_LED = 5;
// Встроенный светодиод NodeMCU-32S подключен к контакту GPIO2
 const int CONNECT_LED = 2;
// Выходной контакт потенциометра подключаем к контакту A0
 const int POT = A0;
// Определяем UUID для сервисов и характеристик
#define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define DHTDATA_CHAR_UUID "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

// Функция обратного вызова, которая обрабатывает состояние соединения Bluetooth:
// устанавливает флаг "deviceConnected" в состояние true или false,
// когда вы подключаетесь или отключаетесь от ESP32
class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };
    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

// Функция обратного вызова, которая обрабатывает
// получение данных, получаемых от клиента (телефона)
class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
      std::string rxValue = pCharacteristic->getValue();
      if (rxValue.length() > 0) {
        Serial.println("**********");
        Serial.print("Received Value: ");
        for (int i = 0; i < rxValue.length(); i++)
```

```
        Serial.print(rxValue[i]);
        Serial.println();

// Здесь я добавил оператор "if", который включает или выключает светодиод
// в зависимости от того, какое письмо отправлено (1 - вкл., 0 - выкл.)
        if (rxValue[0] == '1') {
          Serial.print("Turning ON!");
          digitalWrite(STATUS_LED, HIGH);
        }
        else if (rxValue.find('0') != -1) {
          Serial.print("Turning OFF!");
          digitalWrite(STATUS_LED, LOW);
        }
        Serial.println();
        Serial.println("*********");
      }
    }
};

  void setup() {
// Задаем режим вывода для светодиодов
     pinMode(STATUS_LED, OUTPUT);
     pinMode(CONNECT_LED, OUTPUT);
// Задаем режим ввода для контакта потенциометра
      pinMode(POT, INPUT);
      Serial.begin(115200);
// Инициализируем ESP32 как устройство BLE и устанавливаем его имя:
    BLEDevice::init("myESP32Service");
// Создаем сервер BLE
    pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());
// Создаем службу BLE, используя сервис UUID
    BLEService * pService = pServer-> createService (SERVICE_UUID);
// Добавим характеристики
    pCharacteristic = pService->createCharacteristic(
                CHARACTERISTIC_UUID_RX,
                BLECharacteristic::PROPERTY_NOTIFY
              );
      pCharacteristic->addDescriptor(new BLE2902());
      BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                 CHARACTERISTIC_UUID_RX,
                                 BLECharacteristic::PROPERTY_WRITE
                               );
      pCharacteristic->setCallbacks(new MyCallbacks());
// Запускаем службу
    pService-> start ();
```

```
// Запускаем "обнаружение" ESP32
  pServer-> getAdvertising () -> start ();
  Serial.println ("Waiting for a client to connect ...");
 }

 void loop () {
    if (deviceConnected) {
// Включаем встроенный светодиод на плате ESP32
    digitalWrite(CONNECT_LED, HIGH);
// Считываем значение выходного сигнала потенциометра
    int val = analogRead(POT);
// Выводим полученное значение в окно монитора порта
    Serial.print(F("Analog Value:")); // Аналоговое значение:
    Serial.print(val);
    Serial.println(F("\tSending..."));
// Преобразуем значение val и присваиваем его характеристике
    char valString [2];
    dtostrf (val, 1, 2, valString);
    char valDataString [16];
    sprintf (valDataString, "% d", val);
    pCharacteristic-> setValue (valDataString);
// Отправить значение характеристики на телефон!
    pCharacteristic-> notify ();
    Serial.print ("*** Sent Data:");
    Serial.print (valDataString);
    Serial.println ("***");
  }
   delay (1000);
// Выключаем встроенный светодиод на плате ESP32,
// если нет соединения
   digitalWrite(CONNECT_LED, LOW);
 }
```

4. Откройте приложение **nRF Connect for Mobile**, нажмите кнопку **Scanning**, и телефон должен обнаружить ваше устройство ESP32 под именем **myESP32Service**. Нажмите кнопку **Connect**. Телефон подключится к плате ESP32, и при этом на ней загорится встроенный светодиод.
5. Нажмите на строку **Nordic UART Service** и увидите значение аналогового сигнала в поле **Value** (на рис. 2.25 это значение «722»). Измените значение сопротивления потенциометра, и цифры изменятся.
6. Нажмите на последнюю строку, которая имеет **Properties** (Свойство) **WRITE**.
7. Появится диалоговое окно, в котором введите заглавную латинскую букву *A*, и светодиод загорится, аналогично введите заглавную латинскую букву *B*, и светодиод погаснет.



а *б* *в*

Рис. 2.25. Обмен информацией между смартфоном и платой ESP32

Управление светильником посредством Bluetooth

В книге рассмотрен интересный проект, в котором используется способность мобильных телефонов (iPhone и Android) осуществлять сопряжение с устройствами BLTE (в нашем случае ESP32), если эти устройства предоставляют так называемый профиль HID (Human Interface Device — человеко-машинный интерфейс). Например, любая беспроводная клавиатура имеет профиль HID, и если на устройстве BTLE включить службу HID Keyboard, то оно будет представлять себя как беспроводную клавиатуру. Всякий раз когда мобильный телефон (iPhone и Android) будет находиться в зоне действия сопряженного устройства, он будет автоматически подключаться к нему, не требуя никаких действий с вашей стороны. Мы создадим программу для ESP32, которая будет ожидать такое подключение и включать подключенный к плате светодиод, когда вы приблизите к нему смартфон на определенное расстояние (10–15 м).

1. Подключите к плате ESP32 светодиод, как показано на рис. 2.21.
2. Установите библиотеку клавиатуры BleKeyboard для ESP32. Этой библиотеки нет в Диспетчере библиотек, потому скачайте ее по ссылке **https://github.com/T-vK/ESP32-BLE-Keyboard#installation** и подключите к Arduino IDE **Скетч | Подключить библиотеку | Добавить .ZIP библиотеку**.
3. Загрузите в плату код из листинга 2.8.

Листинг 2.8. Программа BLE_Keyboard_led.ino для управления светодиодом и получения данных с аналогового датчика по каналу BLE

```
#include <BleKeyboard.h>
// Выбор имени клавиатуры Bluetooth (которая отображается в меню Bluetooth
// вашего устройства)
BleKeyboard bleKeyboard;
// Контакт для управления реле светильника
const int LAMP_PIN = 5;
void setup() {
  pinMode(LAMP_PIN, OUTPUT);
  digitalWrite(LAMP_PIN, LOW);

 // Открываем последовательный порт, устанавливаем скорость 115200 бит/с
 Serial.begin(115200);
 // Вывести сообщение на Монитор порта
 Serial.println("Starting BLE work!");

 // Включаем профиль HID Keyboard, чтобы устройства iOs/Android могли распознавать
 // модуль ESP32, не используя специального приложения)
 bleKeyboard.begin();
}
void loop() {
    if (bleKeyboard.isConnected()){
// Включить светодиод (реле светильника) при подключении устройства BTLE
     digitalWrite(LAMP_PIN, HIGH);
     Serial.println("HIGH");
  }
  if (!bleKeyboard.isConnected()){
// Выключить светодиод (реле светильника) при отключении устройства BTLE
     digitalWrite(LAMP_PIN, LOW);
     Serial.println("LOW");
  }
 delay(10);
}
```

4. Включите смартфон/iPhone и подключите его в настройках Bluetooth. По умолчанию имя устройства "ESP32 BLE Keyboard". Светодиод загорится. Если вы удалите телефон от платы ESP32 на расстояние около 10 м, светодиод погаснет.

Внимание!

Данное программное обеспечение является учебным и не оснащено средствами безопасности для предотвращения несанкционированного подключения к устройству Bluetooth и управления им.

3. Управление платой ESP32 через Интернет

3.1. Подготовка оборудования для управления вводом-выводом

Установите микроконтроллер ESP32 в макетную плату. Подключите к плате RGB-светодиод с общим катодом и пьезозуммер (или обычный динамик) как показано на рис. 3.1.

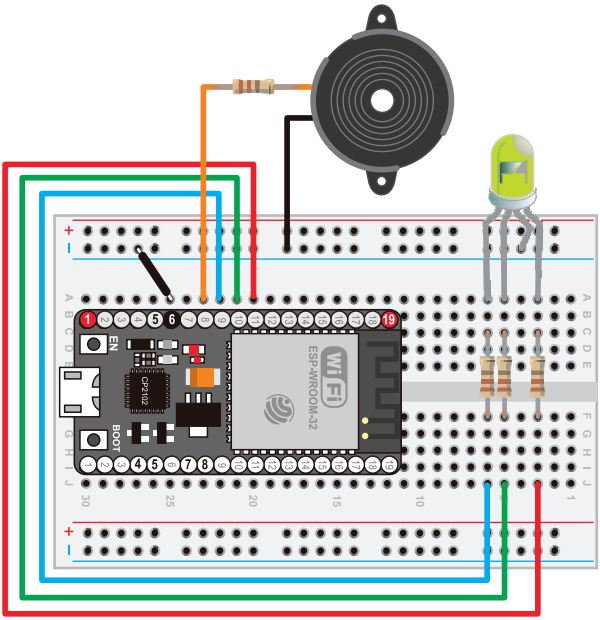


Рис. 3.1. Веб-сервер на плате ESP32 с подключенными к ней RGB-светодиодом и пьезозуммером

Аноды RGB-светодиода нужно подключить через токоограничивающие резисторы номиналом 220 Ом к контактам P25, P26 и P27 (GPIO25, GPIO26 и GPIO27) платы ESP32, а пьезозуммер (или динамик) — к контакту P14 (GPIO14).

3.2. Программа сервера для платы ESP32

Задачу создания программы веб-сервера для исполнения на нашей плате ESP32 будем решать пошагово.

1. Подключаем плату ESP32 к вашей локальной сети через Wi-Fi-соединение и получаем для нее IP-адрес.
2. Разработаем максимально простой веб-сервер, чтобы разобраться, как выглядят запросы HTTP, как выполнять их анализ и как отвечать на них.

3. Разработаем простую веб-страницу, которая будет посылать команды на наш веб-сервер ESP32.
4. Интегрируем веб-страницу и немного кода для управления оборудованием в нашу программу веб-сервера, чтобы получить полнофункциональный проект.

3.2.1. Подключение платы ESP32 к сети Wi-Fi

Подключите плату ESP32 к ПК, как показано в Приложении П1, и загрузите скетч, приведенный в листинге 3.1.

> **Внимание!**
>
> Вместо строк **ВВЕДИТЕ СЮДА ИМЯ SSID СВОЕЙ СЕТИ** и **ВВЕДИТЕ СЮДА ПАРОЛЬ ДЛЯ СВОЕЙ СЕТИ** введите реальные значения вашей сети Wi-Fi.

Листинг 3.1. Программа connect_to_wifi.ino для подключения к сети Wi-Fi

```
#include <WiFi.h>

// Учетные данные сети WiFi
const char* ssid     = " ВВЕДИТЕ СЮДА ИМЯ SSID СВОЕЙ СЕТИ";
const char* password = " ВВЕДИТЕ СЮДА ПАРОЛЬ ДЛЯ СВОЕЙ СЕТИ";

// Обозначаем состояние подключения через встроенный светодиод
const int ONBOARD_LED = 2;

WiFiServer server(80);

void setup()
{
   // Запускаем последовательный интерфейс
   Serial.begin(115200);

   // Задаем режим работы из состояния контактов
  pinMode(ONBOARD_LED, OUTPUT);
  digitalWrite(ONBOARD_LED, LOW);
  delay(10);

   // Сначала подключаемся к сети Wi-Fi

   Serial.println();
   Serial.println();
   Serial.print("Connecting to "); // Подключаемся к:
   Serial.println(ssid);

   WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");                    // Подключились!
  Serial.println("WiFi connected.");

  digitalWrite(ONBOARD_LED, HIGH);  // При установке подключения включаем
                                    // встроенный светодиод

  Serial.println("IP address: ");        // IP-адрес этой платы ESP32:
  Serial.println(WiFi.localIP());

  server.begin();

}

void loop(){
  // Ничего не делаем
}
```

После успешной загрузки скетча в плату ESP32 откройте **Монитор порта** и нажмите на плате кнопку **EN**. Вы увидите сообщение наподобие показанного на рис. 3.2. В нижней строке вы увидите IP-адрес платы ESP32 в вашей сети.

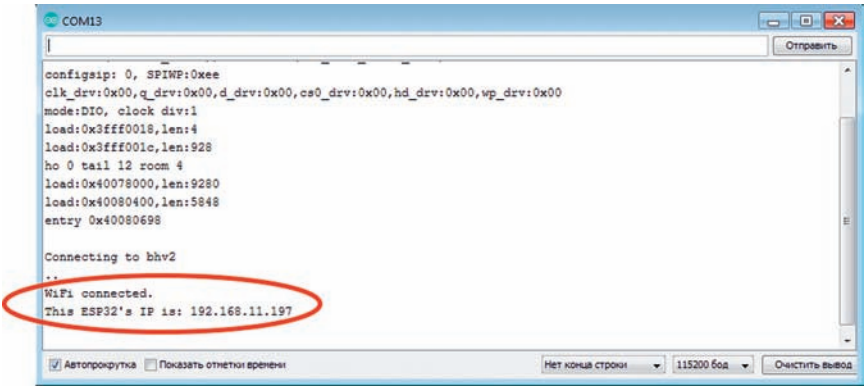


Рис. 3.2. Загрузка скетча прошла успешно

Примечание

Не забудьте при загрузке программы на плату ESP32 нажать на плате кнопку **BOOT** (см. рис. П2.11) и для **Монитора порта** установите скорость 115 200 бод.

3.2.2. Программа минимального веб-сервера

Подключив плату ESP32 к сети Wi-Fi, мы можем реализовать простой HTTP-сервер, который ожидает входящие запросы и отвечает на них, подтверждая их получение.

Загрузите на плату скетч bare_minimum_server.ino (листинг 3.2, который похож на листинг 17.2 из книги, только адаптирован для платы ESP32).

Листинг 3.2. Программа bare_minimum_server.ino минимального веб-сервера

```
#include <WiFi.h>

// Учетные данные сети Wi-Fi
const char* ssid     = " ВВЕДИТЕ СЮДА ИМЯ SSID СВОЕЙ СЕТИ ";
const char* password = " ВВЕДИТЕ СЮДА ПАРОЛЬ ДЛЯ СВОЕЙ СЕТИ ";

// Обозначаем состояние подключения через встроенный светодиод
const int ONBOARD_LED = 2;

WiFiServer server(80);

void setup()
{
   // Запускаем последовательный интерфейс
   Serial.begin(115200);

   // Задаем режим работы из состояния контактов
  pinMode(ONBOARD_LED, OUTPUT);
  digitalWrite(ONBOARD_LED, LOW);
  delay(10);

   // Подключаемся к сети Wi-Fi

   Serial.println();
   Serial.println();
   Serial.print("Connecting to ");  // Подключаемся к:
   Serial.println(ssid);

   WiFi.begin(ssid, password);
   while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
   }

   Serial.println("");                    // Подключились!
   Serial.println("WiFi connected.");

   digitalWrite(ONBOARD_LED, HIGH);  // При установке подключения
                                     // включаем встроенный светодиод
// Выводим в окно монитора порта IP-адрес, полученный через протокол DHCP
```

```
  IPAddress ip = WiFi.localIP();
  Serial.print("This Arduino's IP is: ");     // IP-адрес этой платы Arduino:
  Serial.println(ip);
  Serial.println("");

   server.begin();
}

void loop(){
   // Запускаем сервер, ожидающий входящие подключения клиентов
  WiFiClient client = server.available();

  // Подключился ли клиент (браузер)?
  if(client)
  {
    // Пока клиент подключен, считываем в цикле входящие строки запроса
    while (client.connected())
    {
      // Считываем по одной строке входящих данных за раз
      String incoming_line = "";
      // Используем цикл do-while, чтобы не начинать проверять форматирования строки,
      // пока строка не получит свой первый символ
      do
      {
        while(!client.available());          // Ожидаем поступления следующего байта
        char c = client.read();              // При поступлении считываем его
        incoming_line += c;                  // и добавляем в конец текущей строки
      }
      while (!incoming_line.endsWith("\r\n"));

      Serial.print(incoming_line);           // Выводим в монитор порта только что полученную
                                             // строку
      // Если строка пустая (содержит только символы возврата каретки и новой строки),
      // тогда мы полностью получили весь входящий запрос
      if (incoming_line == "\r\n")
      {
        // Подтверждаем получение запроса, отправляя действительный код ответа
        client.println("HTTP/1.1 200 OK");
        client.println("Content-type:text/html");
        client.println();

        // Теперь можно закрыть подключение
        delay(50);
        client.stop();
      }
    }
  }
}
```

Загрузите скетч из листинга 3.2 в свою плату ESP32, а затем откройте среду Arduino IDE и запустите **Монитор порта**. Далее запустите на компьютере веб-браузер и введите в его строку адреса URL, который веб-сервер на ESP32 отобразил в окне **Монитора порта**. Обратите внимание на то, что ваш компьютер должен быть в той же самой локальной сети, что и ваша плата ESP32.

В браузере должна отобразиться пустая страница с белым фоном (поскольку веб-сервер отправил ему только код ответа 200 без всяких данных). В окне **Монитора порта** должны выводиться запросы браузера, получаемые веб-сервером (рис. 3.3).

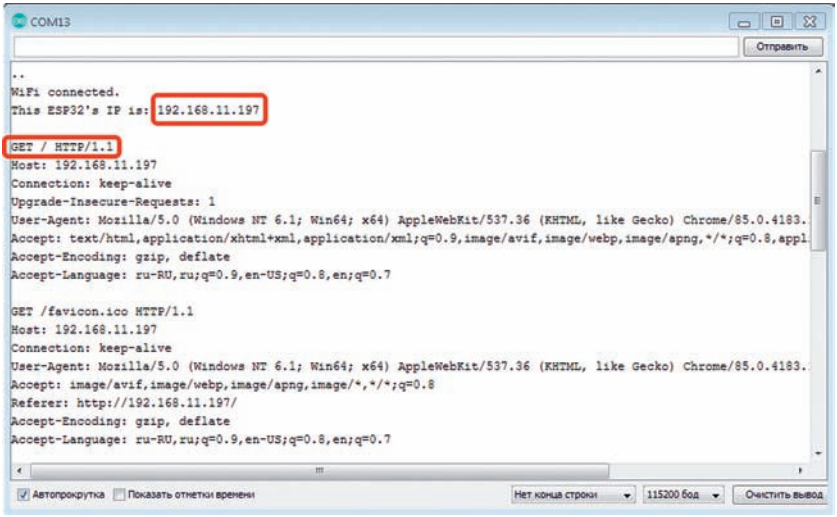


Рис. 3.3. Отображение в окне **Монитора порта** сообщений о запуске веб-сервера на ESP32 и данные получаемых им запросов

3.2.3. Разрабатываем простую веб-страницу

С помощью текстового редактора (например, Notepad) набираем текст веб-страницы (листинг 3.3) и сохраняем в виде файла с расширением html (например, server_form.html).

Листинг 3.3. Простая веб-страница server_form.html

```
<form action='' method='get'>
  <input type='hidden' name='L' value='5' />
  <input type='submit' value='Toggle Red' />
</form>

<form action='' method='get'>
  <input type='hidden' name='L' value='10' />
  <input type='submit' value='Toggle Green' />
</form>
```

```
<form action='' method='get'>
  <input type='hidden' name='L' value='11' />
  <input type='submit' value='Toggle Blue' />
</form>

<form action='' method='get'>
  <input type='range' name='S' min='0' max='1000' step='100' value='0'/>
  <input type='submit' value='Set Frequency' />
</form>
```

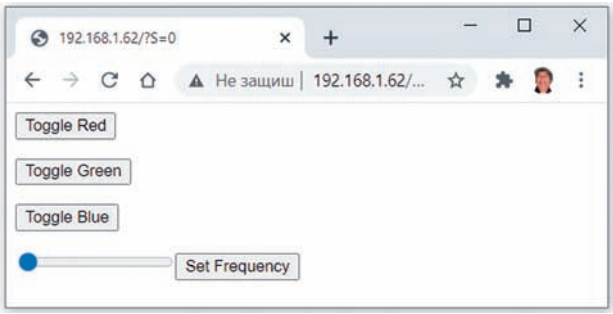


Рис. 3.4. Веб-страница server_form.html

3.2.4. Собираем весь код вместе для создания веб-сервера

Код, приведенный в листинге 3.4, реализует всю функциональность, описанную в предыдущих разделах. Он является аналогом скетча web_control_server.ino (см. листинг 17.4 книги).

Листинг 3.4. Программа esp32-web_control_server.ino веб-сервера на ESP32

```
// Веб-сервер на ESP32  для управления светодиодами и динамиком
// Некоторые части кода адаптированы из книги Arduino Example Code авторства
// Тома Айго (Tom Igoe) и книги Дж. Блума "Изучаем Arduino. Секреты технического
// мастерства"

#include <WiFi.h>
#include <SPI.h>

// Учетные данные сети Wi-Fi
const char* ssid     = "ВВЕДИТЕ СЮДА ИМЯ SSID СВОЕЙ СЕТИ";
const char* password = "ВВЕДИТЕ СЮДА ПАРОЛЬ ДЛЯ СВОЕЙ СЕТИ";

// Обозначаем состояние подключения через встроенный светодиод
const int ONBOARD_LED = 2;
// Контакты, которыми будет управлять форма HTML
```

```
const int RED = 25;
const int GREEN = 26;
const int BLUE = 27;
const int SPEAKER = 14;

// Сервер ожидает подключения на порту 80 (стандартный порт HTTP)
WiFiServer server(80);

void setup()
{
    // Запускаем последовательный интерфейс
    Serial.begin(115200);

    // Задаем режим работы из состояния контактов
   pinMode(ONBOARD_LED, OUTPUT);
   digitalWrite(ONBOARD_LED, LOW);
   pinMode(RED, OUTPUT);
   digitalWrite(RED, LOW);
   pinMode(GREEN, OUTPUT);
   digitalWrite(GREEN, LOW);
   pinMode(BLUE, OUTPUT);
   digitalWrite(BLUE, LOW);
   delay(10);

// Конфигурируем пьезоизлучатель (динамик)
  ledcSetup(3, 8000, 12);
  // Инициализируем пьезоизлучатель (динамик)
  ledcAttachPin(SPEAKER, 3);

    // Подключаемся к сети Wi-Fi

    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");            // Подключаемся к:
    Serial.println(ssid);

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }

    Serial.println("");                        // Подключились!
    Serial.println("WiFi connected.");

    digitalWrite(ONBOARD_LED, HIGH);  // При установке подключения включаем
                                      // встроенный светодиод
```

```
// Выводим в окно монитора порта IP-адрес, полученный через протокол DHCP
  IPAddress ip = WiFi.localIP();
  Serial.print("This Arduino's IP is: ");  // IP-адрес этой платы Arduino:
  Serial.println(ip);
  Serial.println("");

  server.begin();

}

void loop(){
  // Запускаем сервер, ожидающий входящие подключения клиентов
  WiFiClient client = server.available();

  // Подключился ли клиент (браузер)?
  if(client)
  {
    // Пока клиент подключен, считываем в цикле входящие строки запроса
    String command = "";
    while (client.connected())
    {
      // Считываем по одной строке входящих данных за раз
      String incoming_line = "";
      // Используем цикл do-while, чтобы не начинать проверять форматирование строки,
      // пока строка не получит свой первый символ
      do
      {
        while(!client.available());   // Ожидаем поступления следующего байта
        char c = client.read();       // При поступлении считываем его
        incoming_line += c;           // и добавляем в конец текущей строки
      } while (!incoming_line.endsWith("\r\n"));

      Serial.print(incoming_line); // Выводим в монитор порта только что полученную
                                   // строку
      // Выполняем действие, указанное в запросе GET
      // Анализируем и извлекаем данные из строк, выглядящих как: "GET /?L=10 HTTP/1.1"
      if (incoming_line.startsWith("GET /?"))
      {
        // Команда выглядит как L=10
        command = incoming_line.substring(6,incoming_line.indexOf(" HTTP/1.1"));
      }

      // Если строка пустая (содержит только символы возврата каретки и новой строки),
      // тогда мы полностью получили весь входящий запрос
      if (incoming_line == "\r\n")
```

```
{
  // Отвечаем на все полученные полные запросы, выдавая нашу страницу с формой
  // Код ответа 200: Запрос страницы был получен и понят
  client.println("HTTP/1.1 200 OK");
  client.println("Content-type:text/html");
  client.println();

  // Кнопка переключения состояния красного светодиода
  client.print("<form action='' method='get'>");
  client.print("<input type='hidden' name='L' value='" + String(RED) + "' />");
  client.print("<input type='submit' value='Toggle Red' />");
  client.print("</form>");

  // Кнопка переключения состояния зеленого светодиода
  client.print("<form action='' method='get'>");
  client.print("<input type='hidden' name='L' value='" + String(GREEN) + "' />");
  client.print("<input type='submit' value='Toggle Green' />");
  client.print("</form>");

  // Кнопка переключения состояния синего светодиода
  client.print("<form action='' method='get'>");
  client.print("<input type='hidden' name='L' value='" + String(BLUE) + "' />");
  client.print("<input type='submit' value='Toggle Blue' />");
  client.print("</form>");

  // Ползунок установки частоты воспроизводимого на динамике сигнала
  client.print("<form action='' method='get'>");
  client.print("<input type='range' name='S' min='0' max='1000' step='100' value='0'/>");
  client.print("<input type='submit' value='Set Frequency' />");
  client.print("</form>");

  // Здесь можно добавить дополнительные элементы формы для управления
  // другими устройствами

  // Завершаем пустой строкой
  client.println();

  // Теперь можно закрыть подключение
  delay(50);
  client.stop();

  // Если была получена команда, исполняем ее
  if (command.startsWith("L="))
  {
```

```
            int led_pin = command.substring(2).toInt();
            Serial.print("TOGGLING PIN: ");  //Переключаем контакт:
            Serial.println(led_pin);
            Serial.println("");
            digitalWrite(led_pin, !digitalRead(led_pin));
          }
          else if (command.startsWith("S="))
          {
            int speaker_freq = command.substring(2).toInt();
            Serial.print("SETTING SPEAKER FREQUENCY TO: ");
            //Устанавливаем частоту воспроизводимого сигнала равной:
            Serial.println(speaker_freq);
            Serial.println("");
            activateBuzzer(true,speaker_freq);
            if (speaker_freq == 0) noTone(SPEAKER);
            else tone(SPEAKER, speaker_freq);
          }
          // Здесь можно вставить дополнительные операторы 'else if' для обработки других
          // команд
        }
      }
    }
  }

  void activateBuzzer(boolean state, int freq){
    // Если переменная state равна true, то активируем пьезоизлучатель (динамик)
    // с заданной частотой freq
    if(state) ledcWriteTone(3, freq);
  }
```

3.3. Взаимодействие с интерфейсом Web-API

3.3.1. Работа с интерфейсом Web-API для получения метеоданных

Для получения на микроконтроллер ESP32 метеоданных из Интернета можно воспользовался интерфейсом Web-API открытого проекта OpenWeatherMap. Для этого необходимо:

1. Создать учетную запись на сайте **https://openweathermap.org**.
2. Скопировать сгенерированный для вас ключ API (**API Keys**).
3. Установить в Arduino IDE библиотеку **Arduino_JSON by Arduino**.
4. Загрузить на плату ESP32 скетч (листинг 3.5, который является аналогом листинга 17.5 книги; скетч web_weather.ino).

Листинг 3.5. Программа клиента esp32_web_weather.ino для получения текущих метеорологических данных

```
#include <WiFi.h>
#include <HTTPClient.h>
#include <Arduino_JSON.h>
  String json;
  String request;
// Учетные данные сети Wi-Fi
const char* ssid     = "ВВЕДИТЕ СЮДА ИМЯ SSID СВОЕЙ СЕТИ";
const char* password = "ВВЕДИТЕ СЮДА ПАРОЛЬ ДЛЯ СВОЕЙ СЕТИ";

const String API_KEY = "ВВЕДИТЕ СЮДА СВОЙ КЛЮЧ API, ПОЛУЧЕННЫЙ НА САЙТЕ
OPENWEATHERMAP.ORG";
const String HOST = "http://api.openweathermap.org";
const String URL = "/data/2.5/weather?q=";
const String CITY = "Saint Petersburg,ru";

// Обозначаем состояние подключения через встроенный светодиод
  const int ONBOARD_LED = 2;

void setup() {
 // Задаем режим работы и состояние контактов
   pinMode(ONBOARD_LED, OUTPUT);
   digitalWrite(ONBOARD_LED, LOW);

// Запускаем последовательный интерфейс
  Serial.begin(115200);
// Подключаемся к Wi-Fi
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi..");
  }
  Serial.println("Connected to the WiFi network");

  Serial.println("Формируем строку запроса");
  request = HOST+
          "/data/2.5/weather?q=" + CITY +
         "&appid=" + API_KEY;
}
void loop() {
  if ((WiFi.status() == WL_CONNECTED)) {  // Проверяем текущее состояние подключения
    digitalWrite(ONBOARD_LED, HIGH);      // При установке подключения включаем
                                          // встроенный светодиод
```

```
    HTTPClient http;
    http.begin(request);          // Определяем URL
    int httpCode = http.GET();  // Формируем запрос
    if (httpCode > 0) {           // Проверяем код возврата
       json = http.getString();
       Serial.println(httpCode);
       Serial.println(json);
     }
    else {
      Serial.println("Error on HTTP request");
    }
    http.end(); // Высвобождаем ресурсы
  }
 // Обнаружив данные JSON, считываем их в строчную переменную
  JSONVar api_object = JSON.parse(json);
  Serial.println("Raw JSON:");
  Serial.println(api_object);
  double temp = (double) api_object["main"]["temp"];
  Serial.print("Temperature = ");
  Serial.print(temp-273.15);
  delay(30000);
}
```

Результаты выполнения этого скетча для Санкт-Петербурга приведены на экране **Монитора порта** (рис. 3.5).

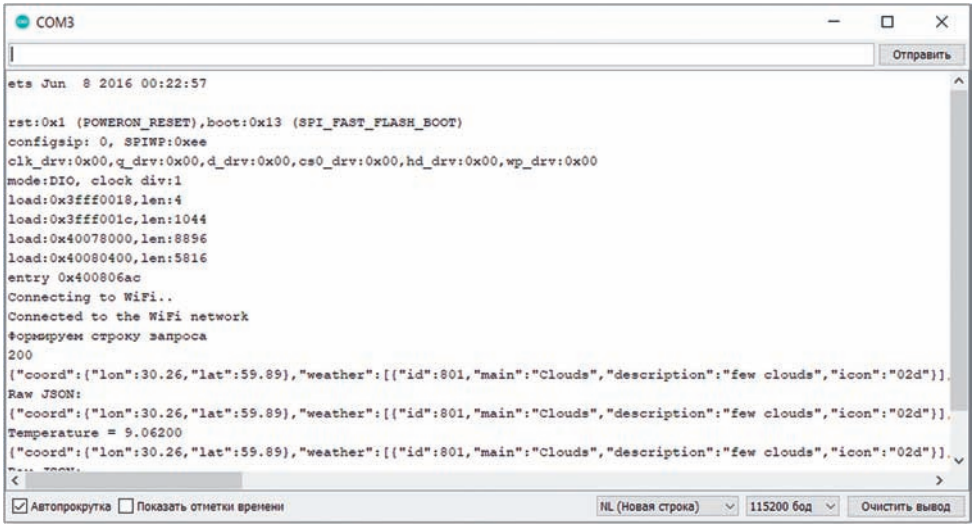


Рис. 3.5. Результат выполнения листинга 3.5 на экране **Монитора порта**

Приложения

П1. Краткое описание плат ESP32

В основе некоторых проектов данного набора лежит плата с микроконтроллером ESP32, разработанная китайской компанией Espressif Systems в 2016 г. Эти микроконтроллеры по своим характеристикам существенно превосходят «стандартные» платы Arduino (Nano/Uno/Mega), имеют низкое энергопотребление, встроенные модули Wi-Fi и Bluetooth и обладают многими другими отличиями.

В настоящее время на рынке представлено большое количество плат на базе ESP32 (рис. П1.1), которые несколько отличаются функциональностью и форм-фактором (DOIT ESP32 DEV KIT, ESP32S NodeMCU, ESP32 Thing и др.).

Рис. П1.1. Некоторые разновидности плат на базе ESP32

В данный набор входит плата ESP32S NodeMCU. Выбор платы обусловлен тем, что ее ширина составляет 26 мм. Это позволяет монтировать ESP32 на стандартную беспаечную макетную плату на 400 контактов таким образом, что остается по одному ряду дорожек для коммутации с обеих сторон платы (рис. П1.2).

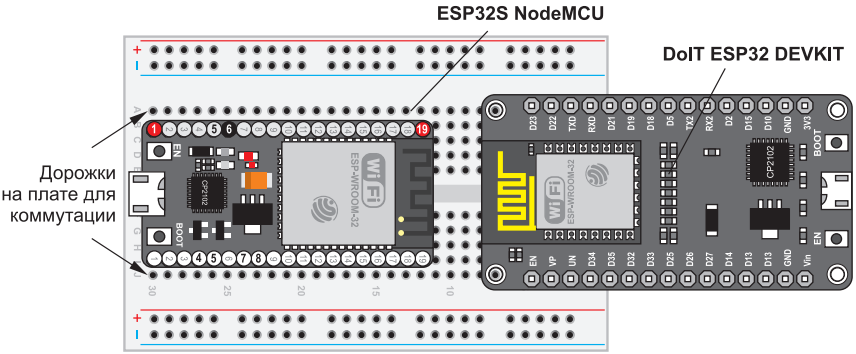


Рис. П1.2 Размещение платформ ESP32 на макетной плате

На рис. П1.3 показана плата ESP-32S NodeMCU. На ней размещены модуль ESP-WROOM-32, стабилизатор напряжения, микросхема CP2102 преобразователя интерфейса USB в интерфейс UART, разъем для подключения кабеля microUSB и кнопки управления.

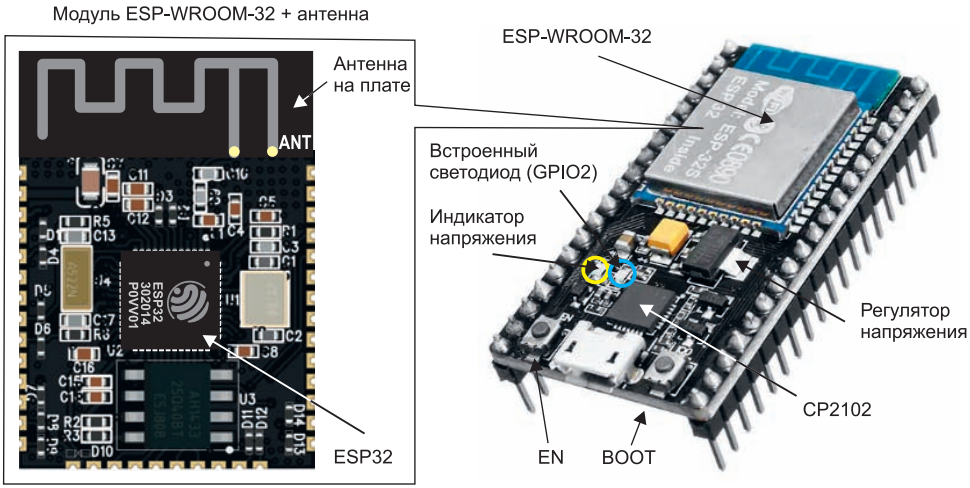


Рис. П1.3. Плата ESP-32S NodeMCU

Назначение выводов ESP-32S NodeMCU показано на рис. П1.4. Для питания платы на ее контакты VIN можно подавать внешнее напряжение 3,3–5 В.

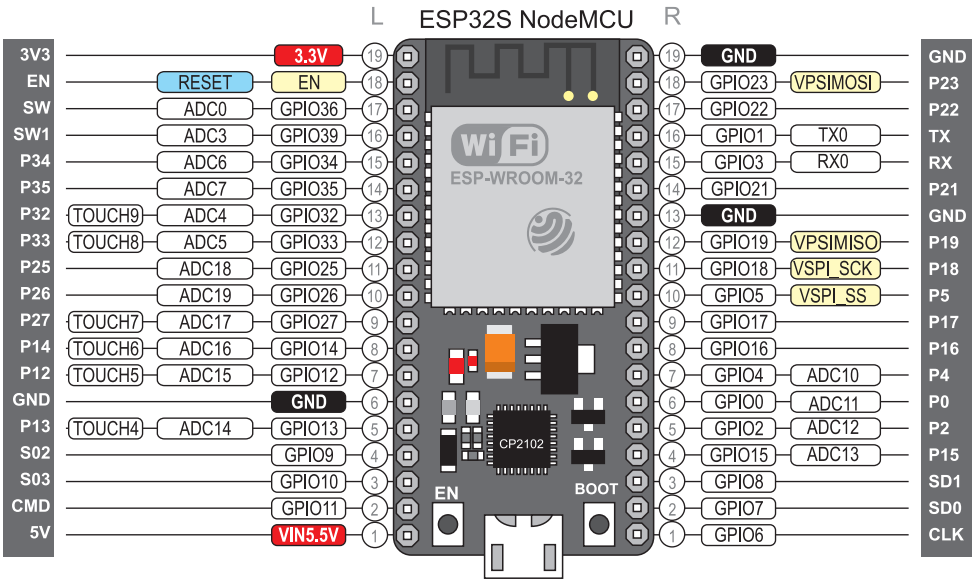


Рис. П1.4. Назначение контактов ESP-32S Node MCU

П2. Установка и настройка Arduino IDE для работы с ESP32

Для того чтобы начать работать с платой ESP32 в среде Arduino IDE, выполните следующие действия:

1. Запустите Arduino IDE и откройте окно **Файл | Настройки**.
2. Введите в поле **Дополнительные ссылки для Менеджера плат** адрес:

https://dl.espressif.com/dl/package_esp32_index.json

Ссылок может быть несколько, они разделяются с помощью запятой следующим образом:

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

3. Откройте окно для установки расширений (**Инструменты | Плата | Менеджер плат**) и установите платформу ESP32, как показано на рис. П2.2.

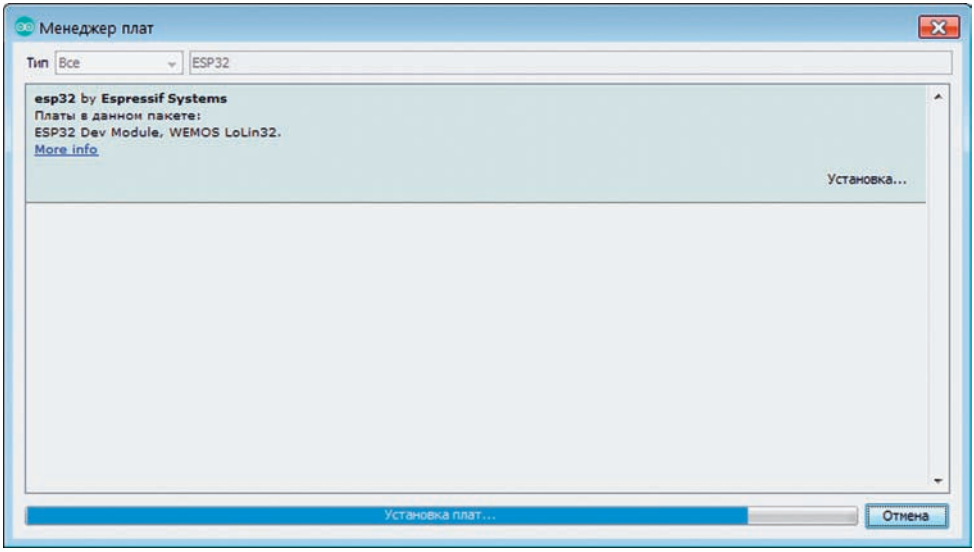


Рис. П2.2. Установка библиотеки **ESP32** в **Менеджере плат** Arduino IDE

Тестируем подключение

Подключите плату ESP32 к USB-порту вашего компьютера и выполните пошагово следующие действия:

1. Откройте среду **Arduino IDE**.
2. Перейдите в меню **Инструменты | Плата** и выберите плату Node32s (рис. П2.3). Если у вас другая модификация платы (например, **DOIT ESP32 DEVKIT V1**, **ESP32 DEV Module** или другая), то укажите ее.

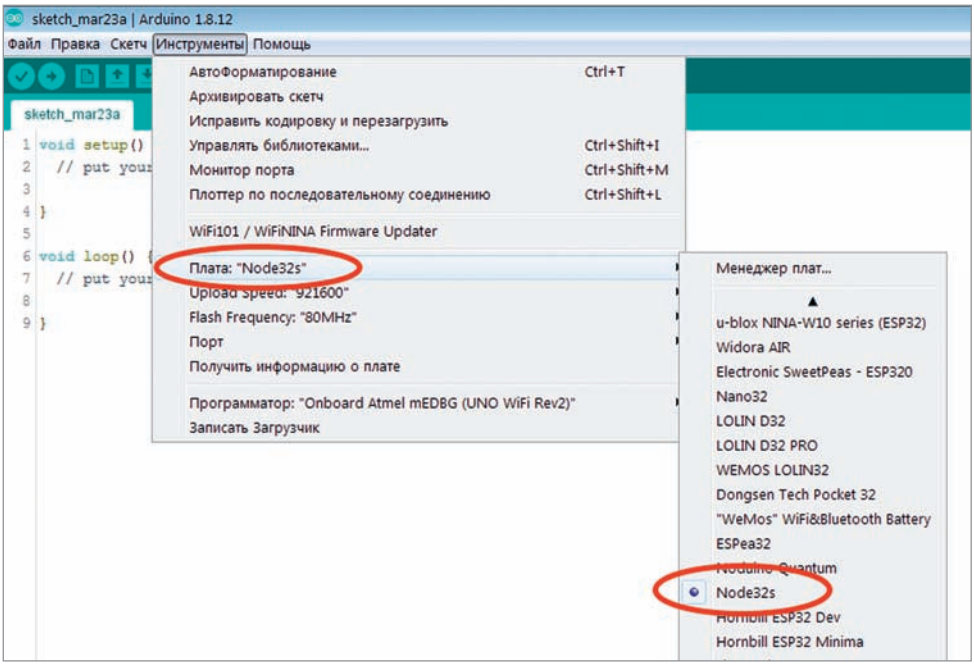


Рис. П2.3. Выбор платы **Node32s**

Если у вас не установлен на ПК драйвер CP2102, то установите его. В Диспетчере устройств появится запись о порте **Selicon Labs CP210x USB to UART Bridge** (в нашем случае это порт COM13, рис. П2.4).

Драйвер CP2102 имеется в электронном архиве, сопровождающем руководство. Последние версии можно скачать с сайта производителя https://www.silabs.com/products/development-tools/software/usb-to-uart-bridge-vcp-drivers

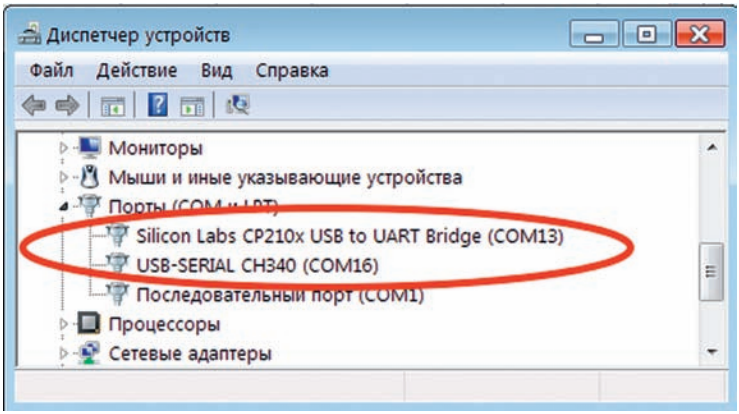


Рис. П2.4. Записи о состоянии портов в Диспетчере устройств

3. В меню **Инструменты | Последовательный порт** выберите порт, который у вас определился в Диспетчере устройств (рис. П2.5). В нижней части окна должна появиться надпись с названием выбранных контроллера и COM-порта. Установите скорость загрузки программ (**Upload Speed**) равной 115 200 бод.

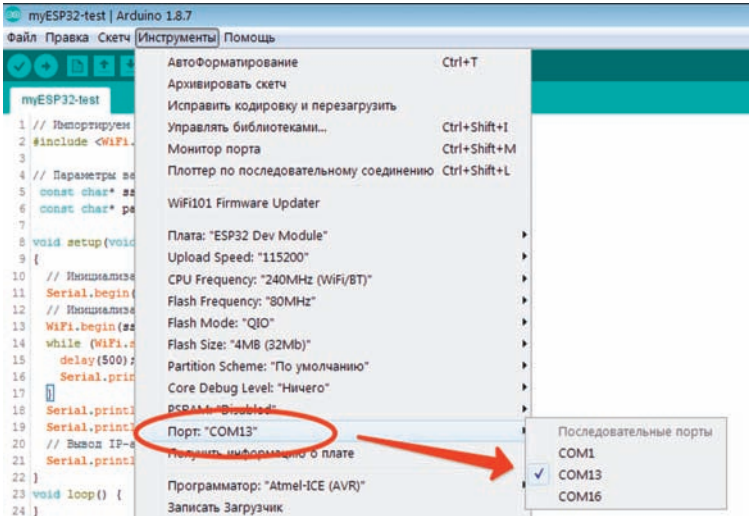


Рис. П2.5. Контроллер и COM-порт выбраны

4. Откройте в примерах скетч WiFiScan (**Файл | Примеры | WiFi | WiFiScan**) (рис. П2.6). Этот скетч сканирует сети Wi-Fi и выводит их на экран **Монитора порта** (рис. П2.7).

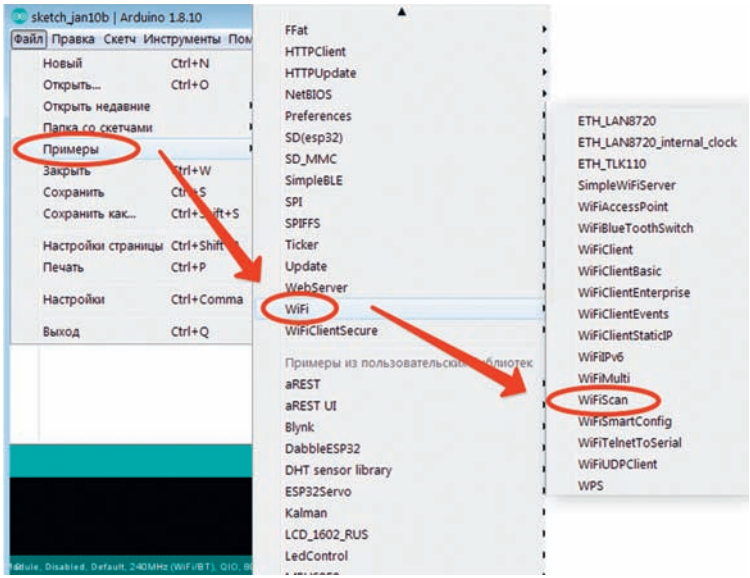


Рис. П2.6. Откройте в примерах скетч WiFiScan (**Файл | Примеры | WiFi | WiFiScan**)

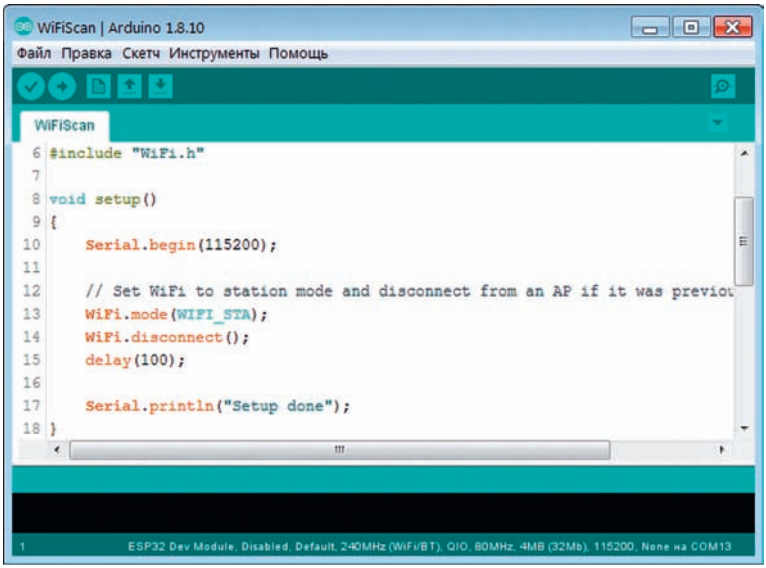


Рис. П2.7. Окно скетча WiFiScan

5. Нажмите в меню Arduino IDE кнопку загрузки прошивки в плату . Начнется компиляция программы с последующей ее загрузкой на плату. Если все прошло хорошо, то вы увидите на экране следующее сообщение: Leaving... Hard resetting via RTS pin... (рис. П2.8).

Если программа не загружается в плату, читайте далее раздел «Советы по устранению неполадок».

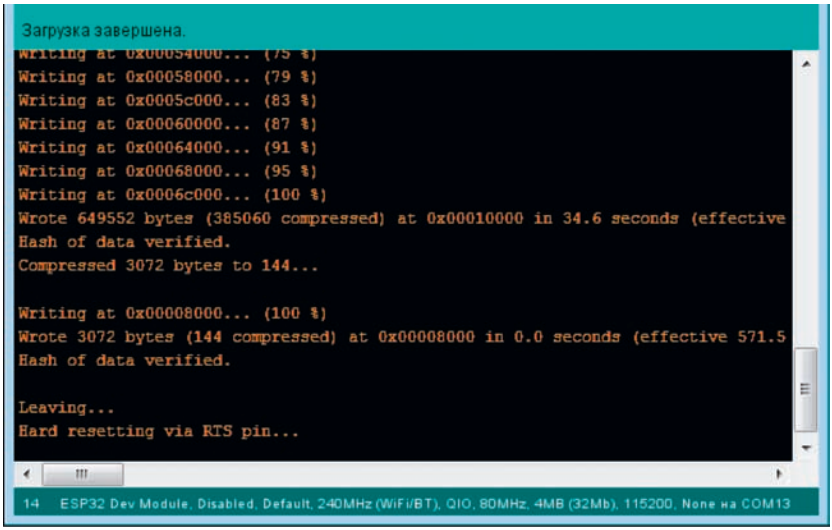


Рис. П2.8. Сообщения об успешном завершении загрузки программы

6. Откройте окно терминала (**Инструменты | Монитор порта**) и установите скорость 115 200 бод. Нажмите кнопку **EN** на плате. Когда соединение будет установлено и плата получит IP-адрес, вы увидите сообщение со списком обнаруженных платой сетей Wi-Fi (наподобие приведенного на рис. П2.9).

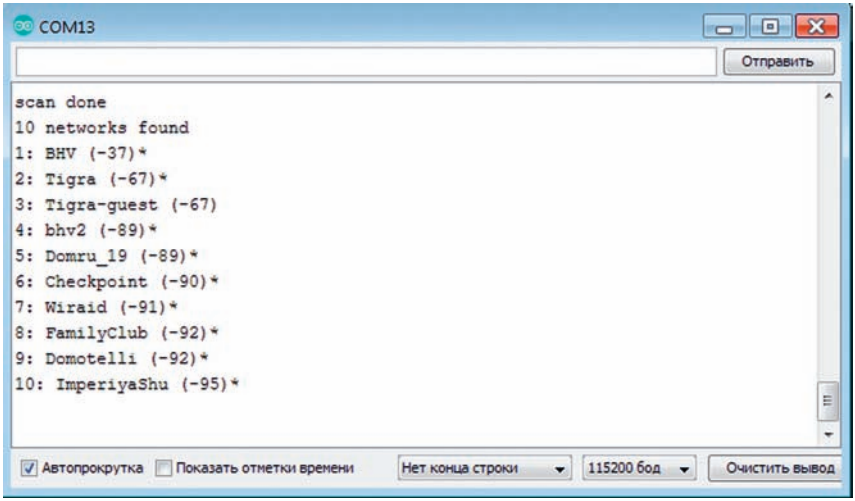


Рис. П2.9. Результаты сканирования сетей Wi-Fi

Это сообщение означает, что ваша плата подключена к сети Wi-Fi.

Советы по устранению неполадок

Ошибка «Failed to connect to ESP32...»

При загрузке программы на плату ESP32 может появиться сообщение

Failed to connect to ESP32: Timed out... Connecting...

которое означает, что ваш модуль ESP32 не находится в режиме перепрошивки/загрузки (рис. П2.10).

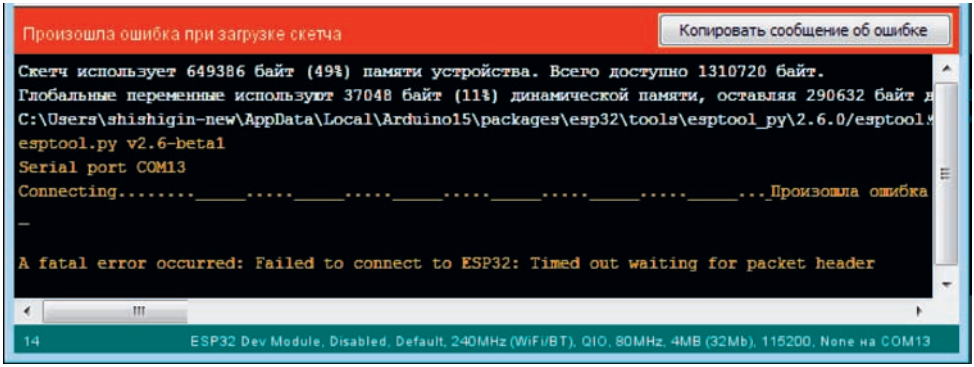


Рис. П2.10. Ошибка загрузки программы

Для устранения этой ошибки при загрузке программы на плату нажмите и удерживайте на плате кнопку **BOOT** (рис. П2.11).

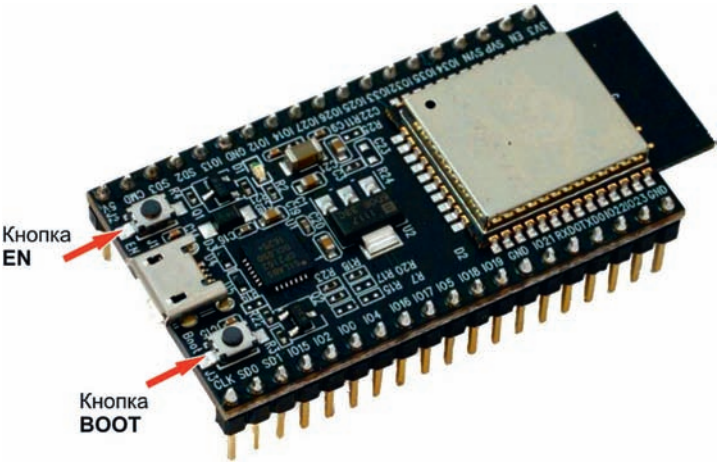


Рис. П2.11. Кнопки **BOOT** и **EN** на плате ESP32

При успешном начале загрузки после слова Connecting... появится соответствующее сообщение в зависимости от типа платы (рис. П2.12). После этого можете отпустить палец с кнопки, и загрузка продолжится.

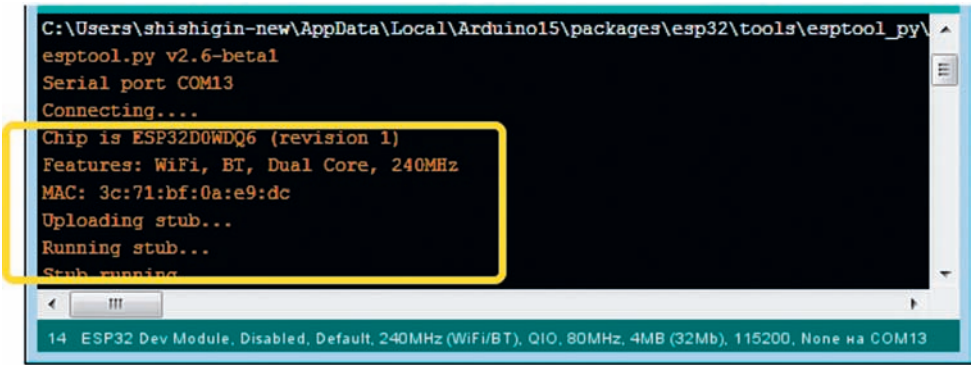


Рис. П2.12. Загрузка программы началась

Ошибка «Brownout detector was triggered...»

После открытия **Монитора порта** по окончании загрузки программы циклически появляется сообщение следующего вида:

12:16:20.295 -> Brownout detector was triggered

...

12:16:20.329 -> load:0x3fff0018,len:4

```
12:16:20.329 -> load:0x3fff001c,len:1100
12:16:20.329 -> load:0x40078000,len:10088
12:16:20.329 -> load:0x40080400,len:6380
12:16:20.329 -> entry 0x400806a4
```

Это означает, что имеется какая-то проблема при взаимодействии платы с ПК. Как правило, причины могут быть следующими:

- некачественный или короткий USB-кабель;
- на кабеле USB отсутствует линия данных;
- некачественный USB-порт компьютера;
- плата с дефектами (плохие паяные соединения);
- загружена прошивка, выполняющая действия, которые приводят к некорректной работе ESP32, например деление на ноль, обращение к несуществующему блоку памяти.

Чтобы локализовать причину ошибки попробуйте использовать другой более короткий USB-кабель (с проводами данных), подключитесь к другому выходу USB вашего ПК или другому ПК, используйте USB-концентратор с внешним источником питания, внимательно проверьте программный код.

Монитор порта Arduino IDE «не работает»

При открытии **Монитора порта** на экране может выводиться всякая «абракадабра» (или экран пуст).

Проверьте, чтобы была установлена скорость 115 200 бод (рис. П2.13), такую же скорость следует установить в прошивке контроллера (Serial.begin(115200)).

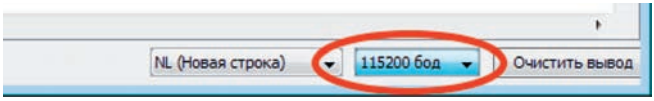


Рис. П2.13. Установите скорость 115 200 бод

П3. Назначение контактов на плате Arduino Uno

В основе набора лежит плата Arduino Uno R3 под управлением микроконтроллера ATmega 328. Для начала работы с устройством достаточно просто подать питание от блока питания или батарейки 9 В либо подключить его к компьютеру посредством USB-кабеля. Назначение контактов на плате Arduino UNO R3 показано на рис. П3.1.

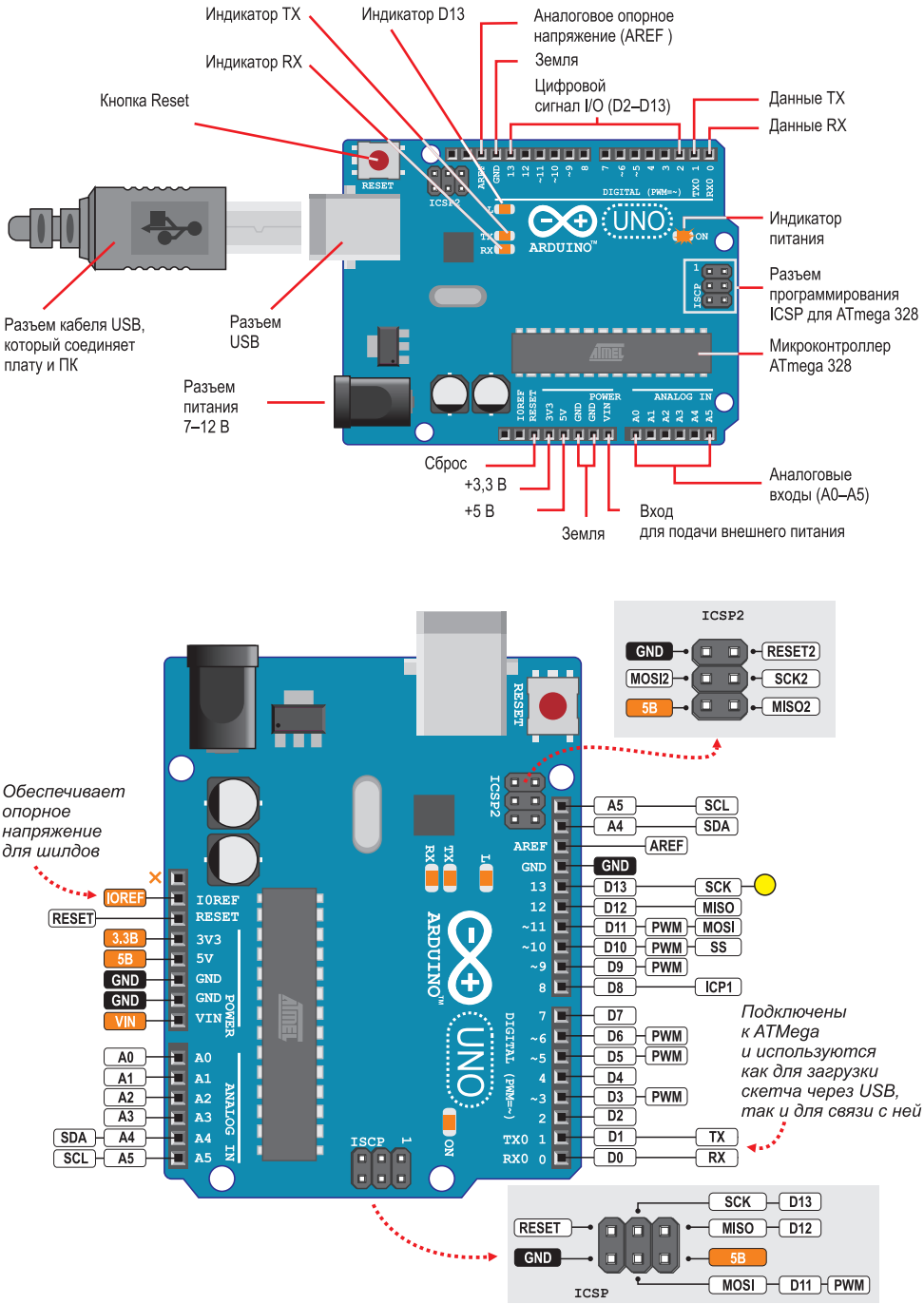


Рис. П3.1. Назначение контактов

П4. Краткая справка по программированию Arduino

Этот раздел ориентирован на тех, кто уже имеет некоторый опыт программирования и нуждается только в пояснении особенностей языка С для программирования Arduino. Для более подробного изучения вопроса рекомендуем посетить сайты **http://arduino.cc/en/Reference/HomePage** (англ.) или **http://arduino.ru/Reference** (рус.).

Основные команды для программирования Arduino приведены в табл. П4.1.

Таблица П4.1. *Описание основных команд для программирования Arduino*

Команда	Описание
Основные операторы	
Основные подпрограммы	void setup() {} void loop() {}
Объявление переменных (тип Integer, значение = 0)	int led = 0;
Объявление констант (тип Integer, значение = A0)	const int analogInPin = A0;
Объявление массивов (тип Integer, 6 значений)	int Arrayname[6] = {2, 4, -8, 3, 2};
Устанавливает режим работы заданного входа/выхода (pin)	pinMode(led, OUTPUT);
Инициализация последовательного интерфейса	Serial.begin(9600);
Подключение библиотеки	#include <libX.h>
Ввод-вывод	
Подает HIGH (или LOW) значение на цифровой вход/выход (pin)	digitalWrite(led,HIGH);
Выдает аналоговую величину (волну ШИМ) на порт вход/выхода	analogWrite(led, 255);
Функция — считывает значение с заданного входа: HIGH или LOW	digitalRead(schalter)
Функция — считывает значение с указанного аналогового входа	analogRead(A0)
Передает данные через последовательный порт как текст ASCII	Serial.print("Hello!");
Передает текст через последовательный порт + перенос строки	Serial.println("Hello!");
Функция — считывает очередной доступный байт из буфера последовательного соединения	Serial.read();
Генерирует сигнал [Pin 8, частота X (Гц), длительность Y (мс)]	tone(8,x) tone(8, x, y)
Останавливает сигнал на порту (8)	noTone(8)
Управляющие операторы	
Конструкция If … Else	if (button == HIGH){ } else{ }

Таблица П4.1 *(продолжение)*

Команда	Описание
Переключатель	switch (x) { case 1: case 2: default:
Период ожидания (500 ms)	delay(500);
Цикл For	for (x = 2; x < 7; x++) {

Цикл While	while (myswitch == HIGH) {
	}
	или: do{
	} while (myswitch ==HIGH);

Преждевременный выход из цикла	break;
Пропускает оставшиеся операторы в текущем шаге цикла	continue;
Математические функции	
Синус, косинус, тангенс	sin(),cos(),tan()
Корень из X	sqrt(x)
X в степени Y	pow(x, y)
Абсолютное значение (значение X)	abs(x)
Случайное число	random()
Сравнение	
Равно	==
Неравно	!=
Больше, меньше	>, <
Больше или равно, меньше или равно	>=, <=
Функция — наименьшее число из двух (X или Y)	min(x, y)
Функция — наибольшее число из двух (X или Y)	max(x, y)
Логические операторы	
И	&&
Или	||
Не	!
Типы данных	**Boolean; char; unsigned char; byte; int; unsigned int; word; long; unsigned long; short; float; double; string (array); String (Object);**
Преобразование типов данных	**char(); byte(); int(); word(); long(); float()**
Прерывания	
Запрет прерываний	noInterrupts()

Таблица П4.1 *(окончание)*

Команда	Описание
Разрешение прерываний	Interrupts()
Включает обработку внешнего прерывания	attachInterrupt(interrupt, function, mode)
Выключает обработку внешнего прерывания	detachInterrupt(interrupt)
Другие	
Комментарии, одна строка	//
Комментарии, несколько строк	/*Комментарий*/
Функция — диапазон преобразований, например: 10-разрядный датчик на 8-битовый ШИМ-сигнал (8-Bit-PWM)	map(значения датчика, 0, 1024, 0, 255);
Функция — проверяет (и, если надо, задает) новое значение (нижняя граница значений/верхняя граница)	constrain(значения датчика, 0, 255);
Для функции — возвращаемое значение	return x;
Переход к метке (label1)	goto label1;
Процедура	void Имя процедуры() { }

Функция, которая возвращает значения Integer и имеет входные параметры Byte	int Имя функции (byte параметры передаваемые в функцию) {
	return 13;
	}

П5. Решение проблем с Arduino

В табл. П5.1 приведены перечень наиболее распространенных ошибок и способы их устранения. Также для поиска возможных ошибок используйте сайт разработчика Arduino **http://arduino.cc/en/guide/troubleshooting#toc18**.

Таблица П5.1. *Наиболее распространенные ошибки и способы их устранения*[1]

Ошибка	Причина	
Загрузка		
Программа не загружается на плату	Неправильно выбран COM-порт	Установите правильный порт в меню **Сервис**	**Последовательный порт**. Если вы не знаете, какой порт правильный, посмотрите его значение в Диспетчере устройств
Сообщение "avrdude: stk500_getsync (): not in sync: resp=0x00"	Неправильно выбран тип платы	Выберите правильно тип платы **Сервис	Плата**

[1] *F. Kainka, Das Franzis Starterpaket Arduino Uno: TURN ON YOUR CREATIVITY, 65 s., Franzis-Verlag.*

Таблица П5.1 *(продолжение)*

Ошибка	Причина	
Невозможно выбрать правильный порт COM	Возможно, плата не была подключена к ПК во время запуска среды разработки Arduino IDE	Перезапустите Arduino IDE
Перестали функционировать монитор порта и среда разработки	Возможно, вы посылаете очень много данных через последовательный интерфейс	Отключите контроллер и закройте программу (в крайнем случае, с помощью Диспетчера задач). Внесите изменения в программу (например, установите паузу с помощью функции delay()). Загрузите программу заново
После кратковременного отключения плата Arduino перестала программироваться	Изменился COM-порт	Проверьте установку COM-порта
Меню **Сервис** открывается очень медленно, работа Arduino IDE завершается аварийно	Устройства Bluetooth могут блокировать последовательный порт, который сканируется программным обеспечением Arduino	Деактивируйте устройство Bluetooth (хотя бы на время)
Меню **Сервис** открывается очень медленно, работа Arduino IDE завершается аварийно	Какое-то приложение на ПК блокирует порты COM в фоновом режиме. Этим приложением могут быть сканеры и брандмауэры	Проверьте установки в брандмауэре
Загрузка программ прерывается	Высокое потребление электроэнергии платой	Подключите дополнительно внешнее электропитание или удалите подключенные к плате периферийные устройства
Ошибка при загрузке	Соединены контакты 0 и 1, которые блокируют процесс загрузки	Удалите шилд или кабели во время процесса загрузки
Среда разработки Arduino IDE «зависает» при загрузке	Возможны проблемы с программой LVPrcSrv.exe (утилитой Logitech для Windows)	Завершите процесс LVPrcSev в Диспетчере задач
Ошибки при компиляции		
error: expected ';' before '}' token	Пропущена точка с запятой	Исправьте программный код
error: expected '}' at of input	Пропущена фигурная скобка	Исправьте программный код
error: 'Имя_переменной' was not declared in this scope	Переменная не декларирована	Исправьте программный код

Таблица П5.1 *(окончание)*

Ошибка	Причина	
error: integer constant is too large for 'long' type	Целое число выходит за границы диапазона целых чисел	Исправьте программный код
java.lang.StackOverflowError [...]	Ошибка переполнения стека. Часто возникает при проблемах с описанием строк	Ищите ошибки в тексте программы, связанные с неправильным использованием двойных (" ") или одинарных (') кавычек, косой черты (\), комментариев и т.д. (например, вместо '\"' вы использовали '"' и т.п.)
Неожиданное поведение программы		
Программа не запускается снова после сброса кнопкой Reset на плате	Вы постоянно посылаете данные на последовательный порт платы, и «новая» программа ожидает окончания этого процесса	Убедитесь, что приложение не отправляет непрерывно последовательные данные плате. При необходимости исправьте программный код
Программа успешно загрузилась, но не функционирует	К плате присоединено слишком много аппаратных средств	Подключите внешний источник питания
Программа успешно загрузилась, но не функционирует	Выбран ошибочный тип платы	Проверьте установку платы (**Сервис** I **Плата**)
Скетч слишком большой	Программа слишком большая для выбранной платы	Оптимизируйте программный код или используйте внешние модули для хранения данных (например, SD-Card)
ШИМ (PWM) не функционирует	На данном контакте ШИМ не используется	Проверьте, имеется ли обозначение «~» на контакте
Ошибки инсталляции		
Java Virtual Machine Launcher: Could not find the main class. Program will exit	Возможно, неправильно распаковался архив Arduino.zip	Распакуйте повторно архив и проверьте наличие файла pde.jar
Неправильно указан COM-порт	Возможно, что неправильно установлен драйвер	Проверьте установку драйвера в Диспетчере устройств и при необходимости переустановите его
Windows не определяет подключенную к USB плату Arduino	Плата подключена к разъему USB со спецификацией 3.0	Подключите плату к разъему USB 2.0
Драйвер не находится	Плата подключена к разъему USB со спецификацией 3.0	Подключите плату к разъему USB 2.0