Б25

Барретт С. Ф.

Б25 Arduino: искусственный интеллект и машинное обучение / пер. с англ. Ю. В. Ревича. – М.: ДМК Пресс, 2024. – 242 с.: ил.

ISBN 978-5-93700-276-1

Автор книги концентрируется на приложениях искусственного интеллекта и машинного обучения для систем на базе микроконтроллеров в среде Arduino на примере платы Arduino Nano 33 BLE Sense. Описаны примеры, пригодные для выполнения в том числе на простейшем 8-разрядном контроллере Arduino Uno.

Издание будет полезно студентам, практикующим инженерам и широкому кругу любителей современной электроники.

ISBN 978-3-031-21876-7 (англ.)
ISBN 978-5-93700-276-1 (рус.)

Содержание

Предисловие

Эта книга посвящена микроконтроллеру Arduino и концепции Arduino. Визионерская команда Arduino в составе Массимо Банзи (Massimo Banzi), Дэвида Куартиелеса (David Cuartielles), Тома Иго (Tom Igoe), Джанлуки Мартино (Gianluca Martino) и Дэвида Мелиса (David Mellis) представила инновацию в области микроконтроллерного аппаратного обеспечения в 2005 году – концепцию оборудования с открытым исходным кодом. Их подход заключался в том, чтобы открыто делиться подробностями построения микроконтроллерных систем и платформами проектирования аппаратного обеспечения для стимулирования обмена идеями и продвижения инноваций. Эта концепция уже много лет популярна в мире программного обеспечения. В июне 2019 года мы с Джоэлом Клейпулом встретились, чтобы спланировать четвертое издание книги «Arduino Microcontroller Processing for Everyone!». Нашей целью было предоставить доступную книгу о быстро развивающемся мире Arduino для широкой аудитории, включая студентов, изучающих изящные искусства, учащихся средних и старших классов, студентов инженерно-проектных специальностей и практикующих ученых и инженеров. Чтобы сделать книгу еще более доступной и лучше служить нашим читателям, мы решили изменить наш подход и предоставить серию меньших томов. Каждый том написан по конкретной теме и для конкретной аудитории.

Книга «Arduino: искусственный интеллект и машинное обучение» исследует приложения Arduino в увлекательном и быстро развивающемся мире небольших локальных приложений искус-

ственного интеллекта и машинного обучения на базе микроконтроллеров. Первые три главы посвящены изучению среды Arduino IDE, микроконтроллера Arduino Nano 33 BLE Sense, а также методам обращения с интерфейсом датчиков и периферийных устройств. В оставшихся трех главах мы обсуждаем обучающий подход к искусственному интеллекту (Artificial Intelligence, AI) и различные концепции машинного обучения (Machine Learning, ML), подходящие для реализации на микроконтроллере, включая метод K ближайших соседей (K Nearest Neighbors, KNN), деревья решений, нечеткую логику, персептроны и искусственные нейронные сети (Artificial Neural Nets, ANN).

Концепция книги

В книге «Arduino: искусственный интеллект и машинное обучение» мы сосредоточимся на искусственном интеллекте (AI) и машинном обучении (ML) для систем на базе микроконтроллеров. Несколько лет назад команда Arduino заявила: «Arduino ставит перед собой задачу сделать машинное обучение достаточно простым, чтобы каждый мог его использовать» [1]. Те, кто знаком с концепциями искусственного интеллекта и машинного обучения, могут удивиться такому подходу: AI и ML наиболее подходят для более мощных вычислительных платформ. Однако недавние разработки позволили некоторым приложениям искусственного интеллекта после их обучения выполняться на микроконтроллерах. Есть приложения, которые подходят для удаленных приложений искусственного интеллекта на базе микроконтроллеров с батарейным питанием [3]. В этой книге мы ограничиваем обсуждение исключительно методами искусственного интеллекта и машинного обучения для микроконтроллеров. Цель состоит в том, чтобы познакомить вас с этими концепциями и дать вам возможность попрактиковаться на недорогом и доступном аппаратном и программном обеспечении Arduino. Надеемся, вы оцените эту книгу как отправную точку, как введение в эту увлекательную область. Мы предоставляем ряд ссылок на источники для дальнейшего изучения.

Рисунок 1 иллюстрирует взаимосвязь между искусственным интеллектом, машинным обучением и методом глубокого обучения (Deep Learning, DL). Цель искусственного интеллекта состоит в том,

чтобы вычислительная техника могла имитировать разумное человеческое поведение. Некоторые относят происхождение искусственного интеллекта к 1300 году до н. э. [6]. Мы ограничиваем наш исторический обзор развитием AI в XX веке и далее. В этой области мы также исследуем нечеткую логику.

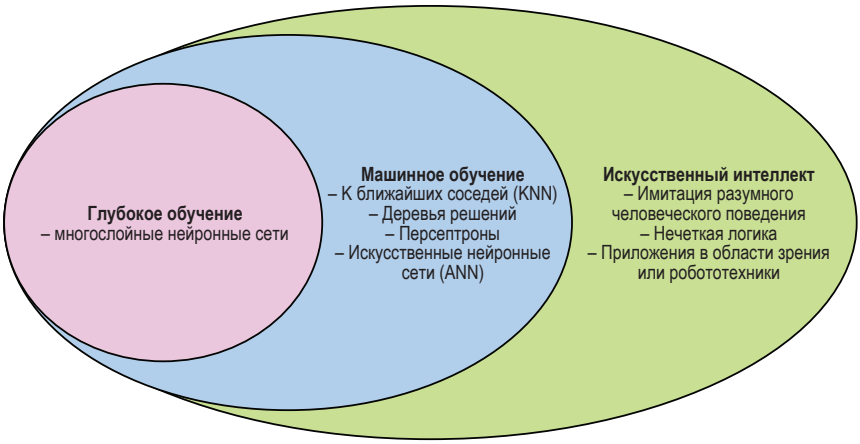

Рис. 1. Искусственный интеллект и машинное обучение [5]

Машинное обучение относится к сфере искусственного интеллекта. Его цель – развитие алгоритмов для управления процессом или для классификации (распознавания) объектов. Разработанный алгоритм подвергается этапу обучения, на котором входные данные используются для подтверждения или разработки желаемых выходных данных контроллера. В процессе обучения алгоритм корректирует определенные веса, чтобы улучшить производительность приложения. В рамках ML мы исследуем алгоритмические методы ближайших соседей (KNN), деревья решений, персептроны и искусственные нейронные сети (ANN). Глубокое обучение предполагает разработку алгоритмов с использованием многослойных искусственных нейронных сетей (ANN).

Для полноты картины в главе 1 этой книги мы приводим необходимую предварительную информацию. В этой главе представлено краткое руководство по началу работы с интегрированной средой разработки Arduino (Arduino IDE).

Глава 2 знакомит с отладочной платой микроконтроллера Arduino Nano 33 BLE Sense. Это микроконтроллер с питанием на-

пряжением 3,3 В постоянного тока[1]. В Nano установлен модуль NINA B306, который включает мощный 32-битный процессор Nordic Semiconductor Arm Cortex-M4F nRF52840 с частотой 64 МГц, содержащий 256 Кбайт статической оперативной памяти (SRAM) и 1 Мбайт флеш-памяти. Модуль также содержит средства связи Bluetooth и Zigbee, подсистемы последовательной связи (UART, I2C, SPI), функции прямого доступа к памяти, аналого-цифровые преобразователи (АЦП) и 128-битный сопроцессор Advanced Encryption Standard (AES) [2, 4].

На отладочной плате Nano 33 также находится обширная серия периферийных устройств, включая девятиосевой инерциальный измерительный блок; датчики барометрического давления, температуры и влажности, приближения, света и жестов; цифровой микрофон; а также криптографический сопроцессор [2].

В главе 3 представлена чрезвычайно важная концепция операционного диапазона для микроконтроллера. Электрические параметры напряжения и тока для микроконтроллеров Arduino применяются для правильного сопряжения устройств входа и выхода с 3.3-вольтовой отладочной платой Arduino Nano 33 BLE Sense. Мы предоставляем основы взаимодействия с Nano 33 для приложений, обсуждаемых в книге.

В главе 4, после краткого исторического обзора, мы исследуем концепции машинного обучения: методы классификации ближайших соседей (KNN) и дерева решений. В пределах области искусственного интеллекта мы исследуем нечеткую логику в главе 5.

В главе 6 мы исследуем персептрон и искусственные нейронные сети (ANN). Глубокое обучение предполагает разработку алгоритмов с использованием многослойных искусственных нейронных сетей. Завершает главу 6 краткое введение в передовые инструменты и приложения глубокого обучения AI и ML.

Источники

1. Arduino Team, Get started with machine learning on Arduino, https://blog.arduino.cc, October 15, 2019.

[1] На протяжении всей книги мы подчеркиваем, что это процессор с напряжением 3,3 В постоянного тока. Сигналы на входах и выходах процессора не должны превышать напряжения 3,3 В! – *Прим. авт.*

2. Arduino Nano 33 BLE Sense, ABX00031, January 5, 2022, www.arduino.cc.
3. G. Lawton, Machine Learning on Microcontrollers Enables AI, targettech.com, November 17, 2021.
4. nRF52840 Advanced multi--protocol System--on--Chip, nRF52840 Product Brief Version 1.0, Nordic Semiconductor.
5. J. P. Mueller and L. Massaron, Artificial Intelligence for Dummies, John Wiley and Sons, Inc, 2018.
6. C. Pickover, Artificial Intelligence an Illustrated History, Sterling, New York, 2019.

Об авторе

Стивен Ф. Барретт (**Steven F. Barrett**), доктор философии (Ph. D.), профессиональный инженер (P. E.). Получил степень бакалавра технологий электронной инженерии в университете Небраски в Омахе в 1979 году, стажировался по обмену из Университета Айдахо в Москве в 1986 году и получил степень доктора философии в Техасском университете в Остине в 1993 году. Формально был действующим преподавателем Академии ВВС США в Колорадо, а сейчас является вице-проректором бакалавриата в Университете штата Вайоминг и профессором электротехники и вычислительной техники. Он является членом ассоциации Institute of Electrical and Electronics Engineers (IEEE, пожизненный Senior) и инженерного сообщества Tau Beta Pi (главный советник).

Исследовательские интересы Стивена Ф. Барретта включают цифровую и аналоговую обработку изображений, компьютерную лазерную хирургию и встраиваемые контроллерные системы. Он является зарегистрированным профессиональным инженером в Вайоминге и Колорадо. С. Ф. Барретт является соавтором нескольких учебников по микроконтроллерам и встроенным системам (совместно с доктором Дэниелом Паком). В 2004 году Фонд Карнеги назвал Барретта «профессором года в Вайоминге» за развитие преподавания, а в 2008 году он стал лауреатом премии Национального общества профессиональных инженеров «Профессиональные инженеры в высшем образовании» (NSPE) за выдающиеся достижения в области образования.

ГЛАВА

1

Начало работы

После изучения этой главы читатель должен уметь делать следующее:

- успешно загрузить и выполнить простую программу с помощью среды разработки Arduino IDE;
- описать ключевые особенности Arduino IDE.

1.1. Обзор

Добро пожаловать в мир Arduino! Как мы говорили, концепция аппаратного обеспечения с открытым исходным кодом Arduino была разработана визионерской командой Arduino в составе Массимо Банзи, Дэвида Куартиллеса, Тома Иго, Джанлука Мартино и Дэвида Меллис в Иврее, Италия. Целью команды было разработать линейку простого в использовании аппаратного и программного обеспечения на основе микроконтроллеров, обеспечивающего достаточную вычислительную мощность, но при этом легко доступного каждому.

В этой главе мы даем краткий обзор процесса написания программ (скетчей) Arduino в среде разработки Arduino IDE. Мы используем нисходящий подход к проектированию: начнем с общей картины предмета главы, затем обсудим среду разработки Arduino IDE и то, как она может быть использована для быстрой разработки скетчей для Arduino Nano 33 BLE Sense.

1.2. Общая картина

Большинство микроконтроллеров программируются на каком-либо варианте языка программирования C[1]. Язык программирования C обеспечивает хороший баланс между контролем аппаратуры микроконтроллера со стороны программиста и экономии времени при написании программы. Как альтернатива[2] среда Arduino (Arduino IDE) обеспечивает удобный интерфейс для быстрой разработки программы (скетча), преобразования ее в машинный код и затем загрузки машинного кода в процессор Arduino за несколько простых шагов, как показано на рис. 1.1.

Первая версия среды разработки Arduino была выпущена в августе 2005 года. Она была разработана в Институте интерактивного дизайна в Иврее, Италия, чтобы дать студентам возможность быстро использовать вычислительную мощность в самых разных проектах. С этого момента обновленные версии, включающие новые функции, выпускаются регулярно (см. [2]).

На самом фундаментальном уровне среда разработки Arduino IDE представляет собой удобный для пользователя интерфейс, позволяющий быстро писать, загружать и выполнять код на микроконтроллере. Минимальный вариант программы может состоять только из функций `setup()` и `loop()`. В среде Arduino IDE могут быть добавлены другие необходимые части, такие как файлы библиотечных заголовков или пользовательские функции. IDE написана на Java и берет свое начало от языка программирования микропроцессоров Wiring Project [2].

---

[1] Последнее время все возрастающая часть разработки программного обеспечения микроконтроллеров осуществляется на языке Python. На него, в частности, ориентирован программный пакет средств AI и ML для микроконтроллеров TensorFlow (см. краткий обзор в разделе 6.7). – *Прим. перев.*

[2] Строго говоря, Arduino IDE не является альтернативой языку программирования C – она допускает использование «чистого C»; однако расширенная версия «языка Arduino» основана на модифицированном языке C++ (т. н. AVRGCC) со специальными добавлениями, ориентированными на конкретные платы контроллеров. В оригинале это платы компании Arduino (например, Arduino Uno или Arduino Nano 33, см. рис. 1.1) и некоторые им родственные, но к среде легко добавляется поддержка многих других контроллеров. – *Прим. перев.*

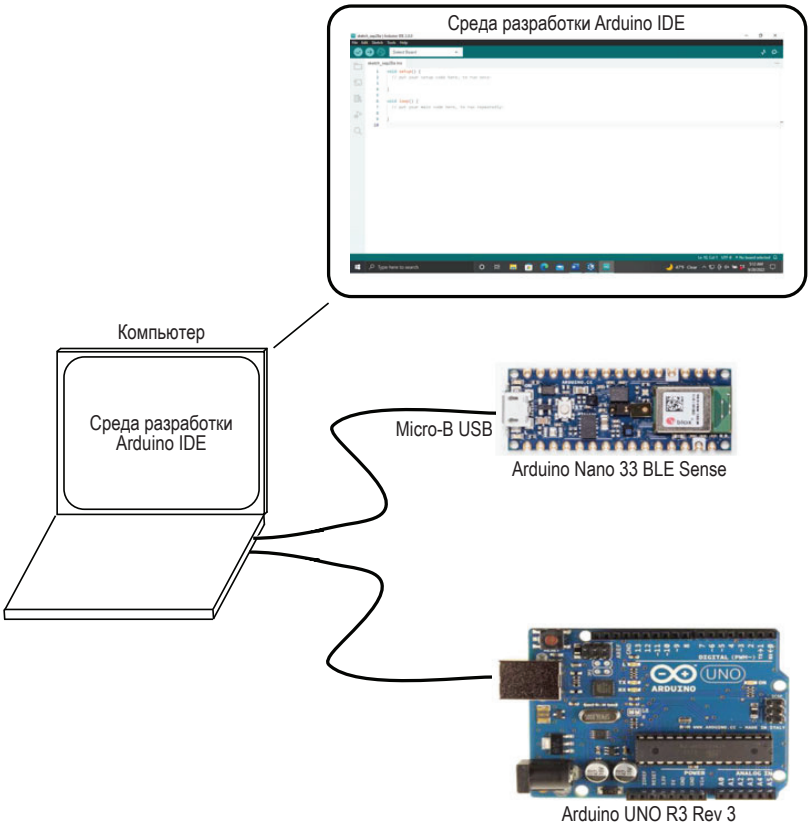

Рис. 1.1. Программирование платы Arduino. С разрешения команды Arduino (лицензия CC BY–NC–SA, www.arduino.cc)

Arduino IDE размещается на ноутбуке или персональном компьютере (PC). После того как программа Arduino, называемая скетчем, написана, она проверяется (компилируется) и загружается на плату Arduino.

1.3. Быстрый старт Arduino

Чтобы начать использовать платформу на базе Arduino Nano 33, вам понадобится следующее оборудование и программное обеспечение:

- плата Arduino Nano 33 BLE Sense;
- интерфейсный кабель (типа USB A – USB Micro-B) от PC или ноутбука к плате Arduino;
- программное обеспечение Arduino IDE.

1.3.1. Краткое руководство по быстрому старту

Среду разработки Arduino можно загрузить с веб-сайта Arduino arduino.cc. Доступны версии для Windows, Mac OS X и Linux. Ниже по шагам представлен быстрый подход к созданию скетча для мигания встроенного светодиода.

- Загрузите среду разработки Arduino IDE с сайта www.arduino.cc.
- Подключите плату Arduino к главному компьютеру с помощью кабеля Micro-B USB.
- Запустите среду разработки Arduino IDE.
- На вкладке **Tools** (*Инструменты*) выберите тип используемой платы и порт, к которому она подключена. Если платы Arduino Nano 33 BLE Sense нет в списке, используйте **Library Manager** (*Менеджер библиотек*)[1]. В этом разделе найдите и установите библиотеку для поддержки нужной платы. Для платы Arduino Nano 33 BLE Sense используется библиотека «Arduino Mbed OS Nano Boards».
- Введите в Arduino IDE следующую программу:

```
//*****************************************************************
#define LED_PIN 13
void setup()
{
  pinMode(LED_PIN, OUTPUT); //set digital pin to output
}
void loop()
```

[1] Автор использует одну из последних версий Arduino IDE (с номерами, начинающимися с двойки, Arduino IDE2), в которой часть меню вынесена на панель инструментов слева (см. далее рис. 1.2). В традиционных версиях с номером, начинающимся с единицы (Arduino IDE1), пункт управления библиотеками расположена по адресу **Sketch** (*Скетч*) -> **Include Library** (*Подключить библиотеку*) -> **Manage libraries** (*Управление библиотеками*). В дальнейшем этих оговорок не делается, в переводе, как и у автора, будет использоваться версия Arduino IDE2. – *Прим. перев.*

```
{
  digitalWrite(LED_PIN, HIGH);
  delay(500); //delay specified in ms
  digitalWrite(LED_PIN, LOW);
  delay(500);
}
//****************************************************************
```

- Загрузите и запустите программу через кнопку с обозначением стрелки вправо **Upload** (*Загрузка*).
- Встроенный светодиод должен мигать с интервалом в одну секунду.

Плата Arduino Nano 33 BLE Sense оснащена красным, зеленым и синим (RGB) светодиодами. Следующий скетч демонстрирует, как управлять каждым RGB и светодиодом питания. Примечание: светодиоды R, G, B имеют активный низкий уровень.

```
//****************************************************************
//RGB_test
//
//Adapted from Controlling_RGB_and_Power_LED by the Arduino Team
// arduino.cc
//Demonstrates control of the RGB and Power LEDs on the NANO 33 BLE boards
//Note: The R, G, B LEDs are asserted active low.
//****************************************************************
#define RED 22 //provide pin locations of LEDs
#define GREEN 23
#define BLUE 24
#define LED_PWR 25
void setup()
{
  pinMode(RED, OUTPUT); //initialize digital pins as output
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
  pinMode(LED_PWR, OUTPUT);
}
void loop()
{
  digitalWrite(RED, HIGH); //turn LEDs off
  digitalWrite(GREEN, HIGH);
  digitalWrite(BLUE, HIGH);
  digitalWrite(LED_PWR, LOW);
  delay(1000); //delay 1s
  digitalWrite(RED, LOW); //turn RGB LEDs on in sequence
  delay(1000);
  digitalWrite(RED, HIGH);
  delay(1000);
  digitalWrite(GREEN, LOW);
  delay(1000);
  digitalWrite(GREEN, HIGH);
```

```
  delay(1000);
  digitalWrite(BLUE, LOW);
  delay(1000);
  digitalWrite(BLUE, HIGH);
  delay(1000);
  digitalWrite(LED_PWR, HIGH);
  delay(1000);
}
//*****************************************************************
```

В следующем скетче используется функция настройки светодиодов R, G и B.

```
//*****************************************************************
//RGB_test2
//
//Adapted from Controlling_RGB_and_Power_LED by the Arduino Team
// arduino.cc
//Demnstrates control of the RGB and Power LEDs on the NANO 33 BLE boards
//Note: The R, G, B LEDs are asserted active low.
//
//This sketch uses a function call to set the LED colors.
//*****************************************************************
#define RED 22 //provide pin locations of LEDs
#define GREEN 23
#define BLUE 24
#define LED_PWR 25
void setup()
{
  pinMode(RED, OUTPUT); //intitialize digital pins as output
  pinMode(GREEN, OUTPUT);
  pinMode(BLUE, OUTPUT);
  pinMode(LED_PWR, OUTPUT);
}
void loop()
{
  RGB_set(LOW, HIGH,HIGH); //red
  RGB_set(HIGH,LOW, HIGH); //green
  RGB_set(HIGH,HIGH, LOW); //blue
  RGB_set(LOW, LOW, HIGH); //yellow
  RGB_set(HIGH,LOW, LOW); //cyan
  RGB_set(LOW, HIGH,LOW); //magenta
  RGB_set(LOW, LOW, LOW); //white
}
//*****************************************************************
void RGB_set(bool R, bool G, bool B)
{
  digitalWrite(RED, HIGH); //turn LEDs off
  digitalWrite(GREEN, HIGH);
  digitalWrite(BLUE, HIGH);
  digitalWrite(LED_PWR, LOW);
  delay(1000); //delay 1s
```

```
  digitalWrite(RED, R); //set LEDs
  digitalWrite(GREEN,G);
  digitalWrite(BLUE, B);
  digitalWrite(LED_PWR, HIGH);
  delay(1000); //delay 1s
  digitalWrite(RED, HIGH); //turn LEDs off
  digitalWrite(GREEN, HIGH);
  digitalWrite(BLUE, HIGH);
  digitalWrite(LED_PWR, LOW);
  delay(1000); //delay 1s
}
//*****************************************************************
```

Загрузив и протестировав среду Arduino IDE, давайте поближе познакомимся с ее особенностями.

1.3.2. Обзор среды Arduino IDE

Среда разработки Arduino IDE показана на рис. 1.2. Arduino IDE содержит текстовый редактор, область сообщений для отображения статуса, текстовую консоль, панель инструментов общих функций и обширную систему меню. Arduino IDE также предоставляет удобный интерфейс для плат Arduino, позволяющий быстро загружать код. Это возможно, потому что контролеры в платах Arduino оснащены программой-загрузчиком.

1.3.3. Концепция альбома для эскизов

В соответствии с аппаратной и программной платформой для студентов, изучающих искусство, среда Arduino использует концепцию альбома для эскизов. Художник сохраняет свои незавершенные работы в подобном альбоме, аналогичным образом программы сохраняются в среде Arduino. Кроме того, мы называем отдельные программы скетчами (набросками, эскизами). Доступ к отдельному скетчу в альбоме можно получить через перечень скетчей на панели инструментов или через меню **File** (Файл).

1.3.4. Программное обеспечение Arduino, библиотеки и ссылки на языки

Среда Arduino IDE имеет ряд встроенных функций. Доступ к некоторым функциям можно получить напрямую через раскрыва-

ющуюся панель инструментов среды Arduino IDE, показанную на рис. 1.2. На рис. 1.3 представлена удобная справочная информация о доступных функциях. Панель инструментов предоставляет широкий спектр функций для создания, компиляции, загрузки и выполнения скетчей.

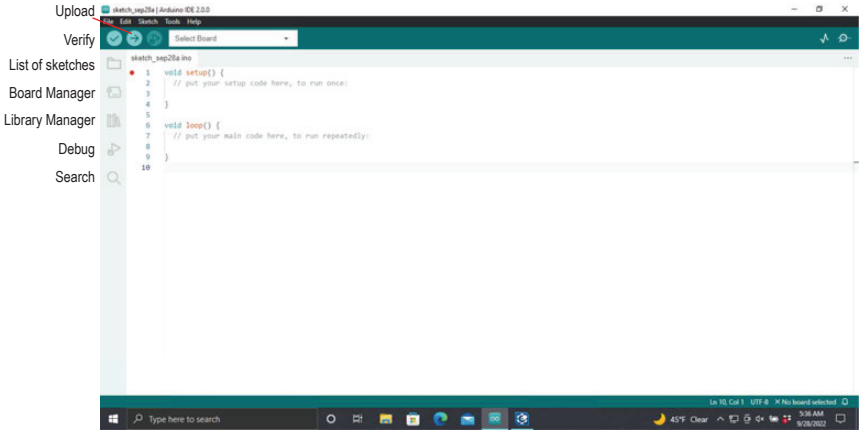

Рис. 1.2. Среда разработки Arduino IDE (www.arduino.cc)

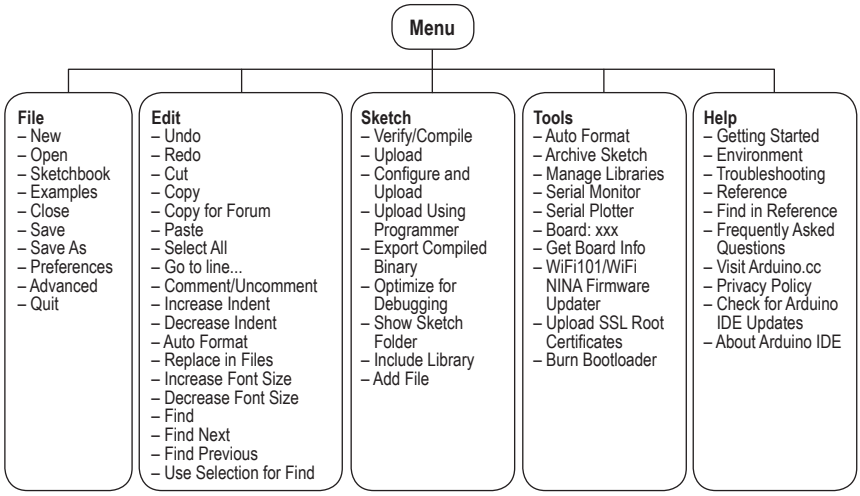

Рис. 1.3. Меню среды разработки Arduino IDE [2]

1.3.5. Написание скетча Arduino

Базовый формат скетча Arduino состоит из функций `setup()` («настройка») и `loop()` («цикл»). Функция настройки выполняется один раз в начале программы. Она используется для настройки выводов, объявления переменных и констант и т. д. Функция цикла будет выполняться последовательно, шаг за шагом.

Когда будет достигнут конец функции `loop()`, программа автоматически вернется к первому шагу этой функции и выполнится снова. Это продолжается непрерывно, пока программа не будет остановлена.

```
//*****************************************************************
void setup()
{
//place setup code here
}
void loop()
{
//main code steps are provided here
 :
 :
}
//*****************************************************************
```

Пример **Давайте вернемся к скетчу, представленному ранее в этой главе:**

```
//*****************************************************************
#define LED_PIN 13 //name pin 13 LED_PIN
void setup()
{
  pinMode(LED_PIN, OUTPUT); //set pin to output
}
void loop()
{
  digitalWrite(LED_PIN, HIGH); //write pin to logic high
  delay(500); //delay specified in ms
  digitalWrite(LED_PIN, LOW); //write to logic low
  delay(500); //delay specified in ms
}
//*****************************************************************
```

В первой строке оператор `#define` связывает обозначение `LED_PIN` с выводом 13 платы Arduino. В функции `setup()` `LED_PIN` назначается выходным контактом (`OUTPUT`). Напомним, что функция `setup()` выполняется только один раз. Затем программа входит в функцию `loop()`, которая выполняется последовательно шаг за шагом и по-

стоянно повторяется. В этом примере для вывода `LED_PIN` сначала устанавливается высокий логический уровень (`HIGH`), чтобы включить светодиод на плате Arduino. Затем возникает задержка (`delay`) в 500 мс. Потом вывод `LED_PIN` устанавливается на низкий уровень (`LOW`). Затем опять возникает задержка в 500 мс. Затем последовательность повторяется.

Даже самые сложные скетчи следуют этому базовому формату: функция настроек, за которой следует функция цикла. Чтобы помочь в разработке более сложных скетчей, среда Arduino IDE имеет множество встроенных элементов, которые можно разделить на структуры (*structure*), переменные (*variables*) и функции (*functions*). Свойства структур и переменных подчиняются правилам, аналогичным правилам языка программирования C[1]. Встроенные функции представляют набор заранее определенных действий, полезных для программиста. Эти встроенные функции обобщены на рис. 1.4.

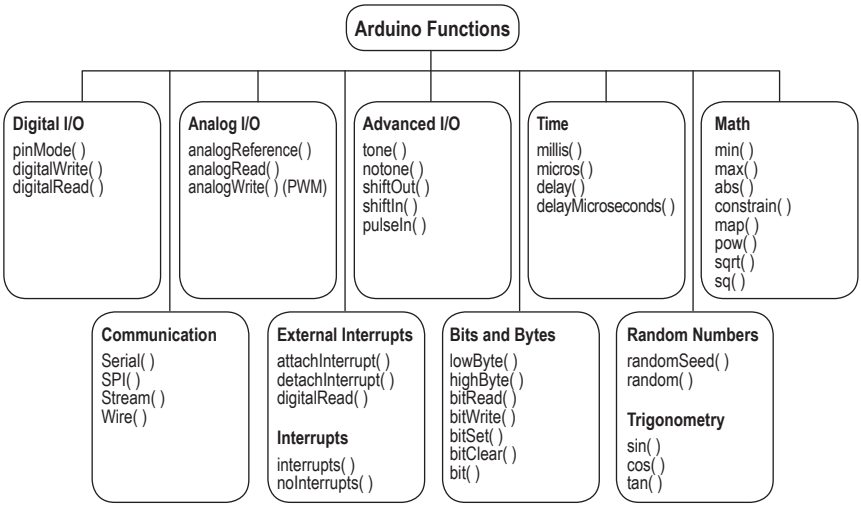

Рис. 1.4. Встроенные функции среды Arduino IDE (www.arduino.cc)

Доступно также множество программных образцовых примеров, позволяющих пользователю быстро построить скетч. Эти примеры обобщены на рис. 1.5. Полную документацию по примерам можно найти на домашней странице Arduino [2]. Эта документация легко доступна также через вкладку **Help** (*Справка*) на панели инстру-

[1] Точнее C++. – *Прим. перев.*

ментов среды разработки Arduino. Здесь эта документация повторяться не будет.

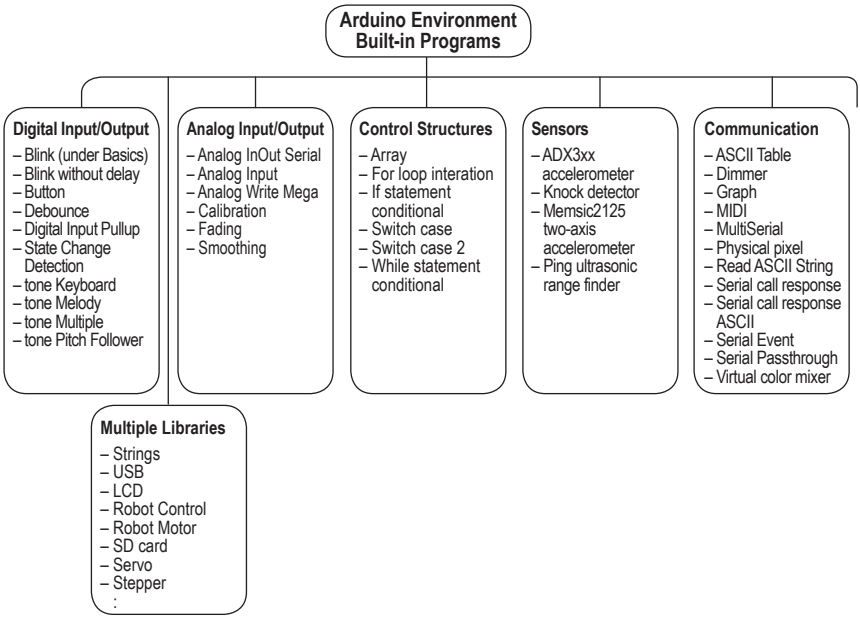

Рис. 1.5. Встроенные примеры программ среды Arduino IDE [2]

Благодаря концепции открытого исходного кода Arduino пользователи по всему миру постоянно создают новые встроенные функции. По мере появления новых функций они добавляются в новых версиях среды Arduino IDE. Как пользователь Arduino вы тоже можете пополнять эту коллекцию полезных инструментов.

На протяжении оставшейся части книги мы будем использовать среду Arduino IDE для программирования в основном платы Arduino Nano 33 BLE Sense. С особенностями Nano 33 мы познакомимся в следующей главе.

1.4. Приложение: светодиодная лента

Светодиодные ленты можно использовать для развлекательных световых дисплеев, игр или для приборных приложений. В этом

примере мы управляем светодиодной лентой на базе контроллера LPD8806 с помощью платы Arduino Nano 33 BLE sense. Мы используем трехметровую светодиодную ленту из 96 RGB-светодиодов, которую можно приобрести в компании Adafruit (www.adafruit.com)[1].

Интенсивность красного, синего и зеленого компонентов каждого RGB-светодиода задаются независимо с помощью восьмибитного кода. Самый старший бит (MSB) – это логическая единица, за которой следуют семь битов для установки яркости светодиода (от 0 до 127). Значения компонентов последовательно выводятся из Arduino Nano 33 BLE Sense с использованием функций последовательного периферийного интерфейса (SPI), как показано на рис. 1.6, *а*. Мы обсудим интерфейс SPI подробнее в следующей главе.

Первое выведенное значение компонента соответствует одному цвету светодиода, ближайшего к микроконтроллеру. Каждое последующее значение компонента фиксируется для соответствующего компонента R, G и B очередного светодиода (см. далее). При получении значения следующего компонента значение предыдущего запоминается и сохраняется постоянным. Для запоминания окончательного значения параметров требуется дополнительный байт. Нулевой байт 0x00 используется для завершения последовательности данных и возврата к первому светодиоду (www.adafruit.com).

Как показано на рис. 1.6, *в*, между Nano 33 и светодиодной лентой требуется всего четыре соединения. Соединения имеют цветовую маркировку: красный – питание, черный – земля, желтый – данные и зеленый – тактовые импульсы. Важно отметить, что для светодиодной ленты требуется питание 3,3 В постоянного тока и номинальный ток 2 А на каждый метр светодиодной ленты.

В этом примере каждый компонент RGB отправляется на полосу отдельно. Пример иллюстрирует, как каждая переменная в программе управляет определенным аспектом светодиодной ленты. Вот несколько важных замечаний по реализации:

- SPI необходимо настроить старшим битом (MSB) вперед;
- яркость светодиода выражается семибитным числом. Старший (восьмой) бит (MSB) каждого байта должен быть установлен в логическую единицу;

---

[1] В российских условиях RGB-ленту на основе контроллера LPD8806 можно приобрести в интернет-магазинах, торгующих китайской продукцией. Следует отметить, что у ленты на основе контроллера LPD8806 есть много аналогов, но каждый из них требует, как правило, своей особой библиотеки. – *Прим. перев.*

- для каждого светодиода требуется отдельный компонент интенсивности цветов RGB. Порядок данных при передаче: G–R–B;
- после отправки данных для всех светодиодов для возврата полосы к первому светодиоду должен быть отправлен нулевой байт (0x00);
- таким образом, поток битовых данных для каждого светодиода: 1–G6–G5–G4–G3–G2–G1–G0–1–R6–R5–R4–R3–R2–R1–R0–1–B6–B5–B4–B3–B2–B1–B0.

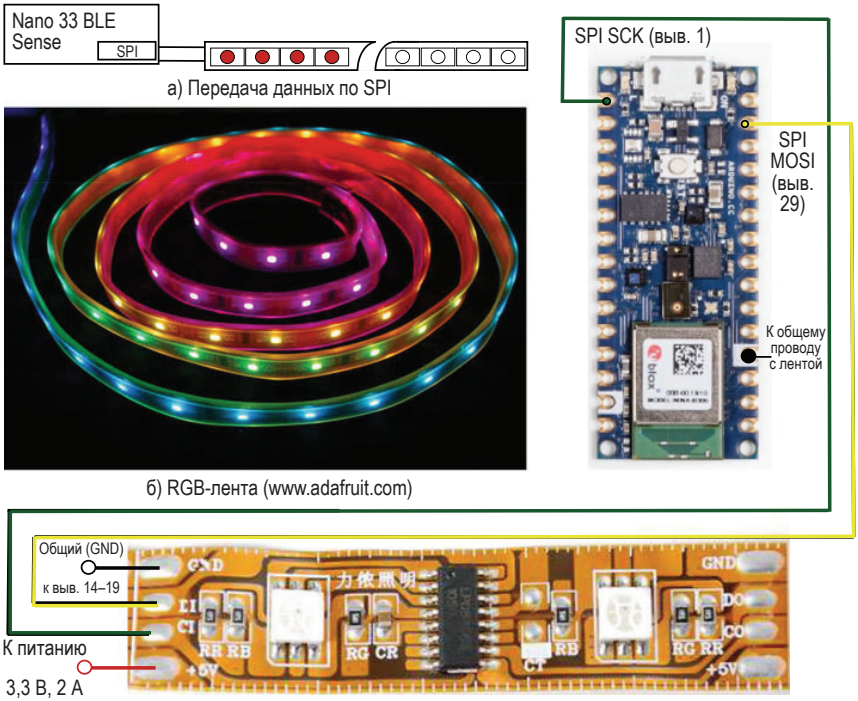

Рис. 1.6. Плата Nano 33 BLE Sense, управляющая светодиодной лентой. С разрешения компании Adafruit (www.adafruit.com) и команды Arduino (CC BY–NC–SA) (www.arduino.cc)

```
//*********************************************************************
//RGB_led_strip_tutorial: illustrates different variables within
//RGB LED strip
//
//LED strip LDP8806 - available from \url{www.adafruit.com} (#306)
```

```
//
//Connections:
// - External 3.3 VDC supply, 2A per LED meter - red
// - Ground - black - include common ground with Nano BLE Sense
// - Serial Data In - Arduino pin 29 (MOSI pin)- yellow
// - CLK Arduino pin 1 (SCK pin)- green
//
//Variables:
// - LED_brightness - set intensity from 0 to 127
// - segment_delay - delay between LED RGB segments
// - strip_delay - delay between LED strip update
//
//Notes:
// - SPI must be configured for Most Significant Bit (MSB) first
// - LED brightness is seven bits. Most Significant Bit (MSB)
// must be set to logic one
// - Each LED requires a separate R-G-B intensity component. The order
// of data is G-R-B.
// - After sending data for all strip LEDs. A byte of (0x00) must
// be sent to return strip to first LED.
// - Data stream for each LED is:
//1-G6-G5-G4-G3-G2-G1-G0-1-R6-R5-R4-R3-R2-R1-R0-1-B6-B5-B4-B3-B2-B1-B0
//
//This example code is in the public domain.
//*****************************************************************
#include <SPI.h>
#define LED_strip_latch 0x00
const byte strip_length = 96;        //number of RGB LEDs in strip
const byte segment_delay = 10;       //delay in milliseconds
const byte strip_delay = 10;         //delay in milliseconds
unsigned char LED_brightness;        //0 to 127
unsigned char position;              //LED position in strip
void setup()
  {
    SPI.begin();                     //SPI support functions
  }
void loop()
 {
  SPI.beginTransaction(SPISettings(200000, MSBFIRST, SPI_MODE3));
  SPI.transfer(LED_strip_latch);    //reset to first segment
  clear_strip();                    //all strip LEDs to black
  delay(50);
//increment the green intensity of the strip LEDs
  for(LED_brightness = 0; LED_brightness <= 60;
      LED_brightness = LED_brightness + 10)
  {
  for(position = 0; position<strip_length; position = position+1)
   {
     SPI.transfer(0x80 | LED_brightness);  //Green - MSB 1
     SPI.transfer(0x80 | 0x00);            //Red - none
     SPI.transfer(0x80 | 0x00);            //Blue - none
     delay(segment_delay);
   }
```

```
   SPI.transfer(LED_strip_latch); //reset to first segment
   delay(strip_delay);
  }
  clear_strip();                        //all strip LEDs to black
  delay(50);
//increment the red intensity of the strip LEDs
  for(LED_brightness = 0; LED_brightness <= 60;
      LED_brightness = LED_brightness + 10)
  {
  for(position = 0; position<strip_length; position = position+1)
   {
     SPI.transfer(0x80 | 0x00);               //Green - none
     SPI.transfer(0x80 | LED_brightness);     //Red - MSB1
     SPI.transfer(0x80 | 0x00);               //Blue - none
     delay(segment_delay);
   }
   SPI.transfer(LED_strip_latch);      //reset to first segment
   delay(strip_delay);
   }
  clear_strip(); //all strip LEDs to black
  delay(50);
//increment the blue intensity of the strip LEDs
  for(LED_brightness = 0; LED_brightness <= 60;
      LED_brightness = LED_brightness + 10)
   {
    for(position = 0; position<strip_length; position = position+1)
     {
       SPI.transfer(0x80 | 0x00);              //Green - none
       SPI.transfer(0x80 | 0x00);              //Red - none
       SPI.transfer(0x80 | LED_brightness);    //Blue - MSB1
       delay(segment_delay);
     }
    SPI.transfer(LED_strip_latch);      //reset to first segment
    delay(strip_delay);
   }
  clear_strip();                        //all strip LEDs to black
  SPI.endTransaction();
  delay(50);
 }
//*****************************************************************
void clear_strip(void)
 {
//clear strip
  for(position = 0; position<strip_length; position = position+1)
    {
      SPI.transfer(0x80 | 0x00);                //Green - none
      SPI.transfer(0x80 | 0x00);                //Red - none
      SPI.transfer(0x80 | 0x00);                //Blue - none
    }
  SPI.transfer(LED_strip_latch);       //Latch with zero
  delay(200);                          //clear delay
 }
//*****************************************************
```

1.5. Выводы

Цель этой главы – предоставить введение в учебное пособие по Arduino IDE. Мы использовали нисходящий подход к проектированию: начали с «общей картины» главы, за которой последовал обзор среды разработки Arduino IDE.

1.6. Задания

1. Опишите этапы написания скетча и его выполнения на плате Arduino.
2. Для чего используется функция монитора последовательного порта в среде Arduino IDE?
3. Опишите, какие переменные используются в следующих встроенных примерах Arduino, а также их основную функциональность: Blink, Analog Input.
4. Что подразумевается под термином «открытый исходный код» (open source)?
5. RGB-светодиоды на плате Nano 33 BLE зажигаются при подаче низкого уровня. Что это значит?
6. Будьте изобретательны! Измените скетч, управляющий светодиодами ленты, чтобы создать другой шаблон.

Наслаждайтесь!

Источники

1. Домашняя страница Arduino, www.arduino.cc.
2. Arduino Nano 33 BLE Sense, ABX00031, 5 января 2022 г. www.arduino.cc.