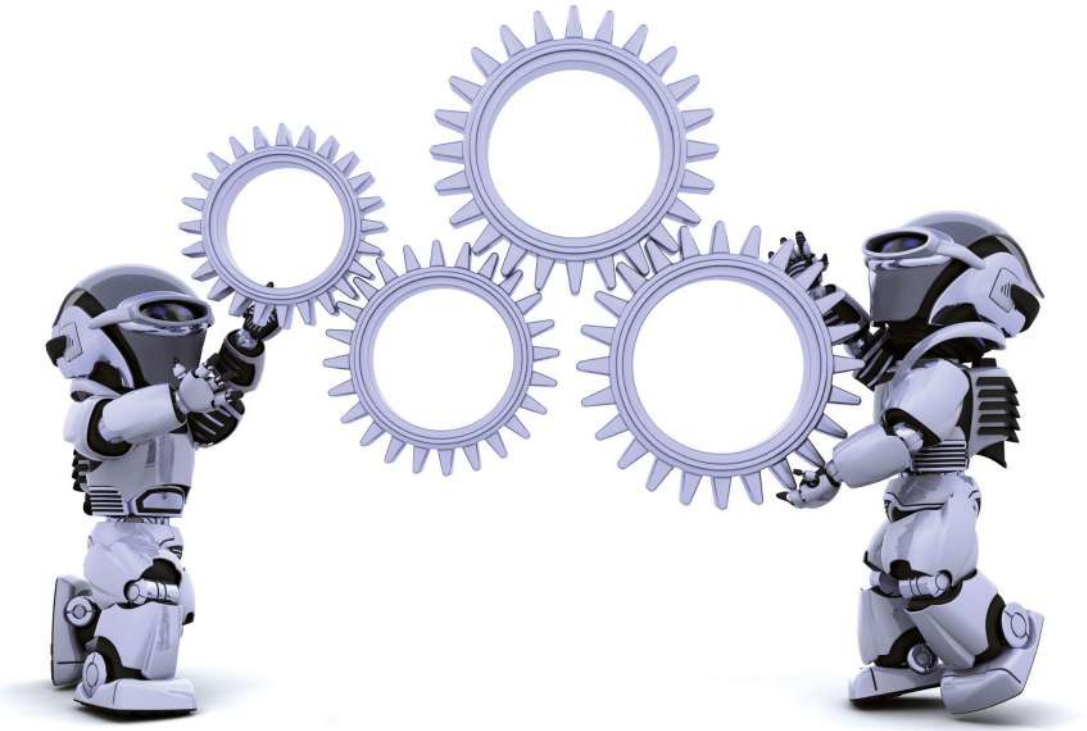
Стивен Ф. Барретт

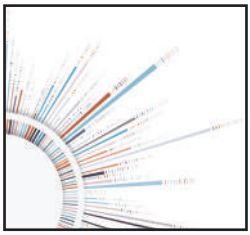
ARDUINO:

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МАШИННОЕ ОБУЧЕНИЕ



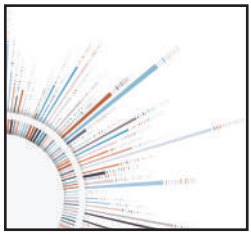
Стивен Ф. Барретт

Arduino: искусственный интеллект и машинное обучение



Arduino V: Machine Learning

Steven F. Barrett



Arduino: искусственный интеллект и машинное обучение

Стивен Ф. Барретт

Барретт С. Ф.
Arduino: искусственный интеллект и машинное обучение /
– 2024. – 242 с.: ил.

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.

Содержание

От издательства 9
Предисловие 11
Благодарности 16
Об авторе 17

Глава 1 Начало работы 18
1.1. Обзор 18
1.2. Общая картина 19
1.3. Быстрый старт Arduino 20
1.3.1. Краткое руководство по быстрому старту 21
1.3.2. Обзор среды Arduino IDE 24
1.3.3. Концепция альбома для эскизов 24
1.3.4. Программное обеспечение Arduino, библиотеки и ссылки на языки 24
1.3.5. Написание скетча Arduino 26
1.4. Приложение: светодиодная лента 28
1.5. Выводы 33
1.6. Задания 33
Источники 33

Глава 2 Arduino Nano 33 BLE Sense 34
2.1. Обзор 34

2.2. Плата Arduino Nano 33 BLE Sense....35
2.3. Возможности Arduino Nano 33 BLE Sense....37
2.4. Подсистемы модуля NINA B306....37
2.4.1. Память модуля B306....40
2.4.1.1. Программируемая флеш-память....40
2.4.1.2. Статическая память с произвольным доступом (SRAM) в модуле B306....40
2.5. Периферийные устройства модуля NINA B306....41
2.5.1. Каналы широтно-импульсной модуляции (PWM)....41
2.5.2. Последовательная связь....43
2.5.2.1. USART....43
2.5.2.2. Последовательный периферийный интерфейс (SPI)....47
2.5.2.3. Интерфейс I2C (TWI)....52
2.5.2.4. Аналого-цифровой преобразователь ADC....53
2.5.3. Bluetooth с низким энергопотреблением (BLE)....56
2.5.3.1. Библиотека ArduinoBLE....59
2.6. Периферийные устройства Nano 33 BLE Sense....65
2.6.1. Девятиосевой IMU LSM9DS1....65
2.6.2. Барометр и датчик температуры LPS22HB....67
2.6.3. Датчик относительной влажности и температуры HTS221....69
2.6.4. Цифровой датчик расстояния, окружающего освещения, RGB-цвета и распознавания жестов APDS-9960....71
2.6.4.1. Распознавание жестов....71
2.6.4.2. Датчик цвета....74
2.6.4.3. Датчик расстояния....76
2.6.5. Цифровой микрофон MP34DT05....77
2.7. Приложение: Bluetooth BLE GreenhouseMonitor....80
2.8. Выводы....87
2.9. Задания....87
Источники....88

Глава 3 Arduino Nano 33 BLE Sense: питание и сопряжение с внешними устройствами....90

3.1. Обзор....90
3.2. Требования к питанию Arduino....91
3.3. Стабилизаторы напряжения....91
3.3.1. Питание Nano 33 от батарей....93

3.4. Концепции сопряжения с внешними устройствами ...93
3.5. Устройства ввода ...94
3.5.1. Переключатели и кнопки ...94
3.5.1.1. Устранение дребезга контактов ...96
3.6. Выходные устройства ...98
3.6.1. Светоизлучающие диоды (LED) ...98
3.6.2. Жидкокристаллический дисплей (ЖК-дисплей, LCD) ...99
3.7. Принципы управления двигателем ...99
3.7.1. Двигатель постоянного тока ...102
3.7.1.1. Характеристики двигателей постоянного тока ...102
3.7.1.2. Однонаправленное управление двигателем постоянного тока ...103
3.7.1.3. Управление скоростью двигателя постоянного тока – широтно-импульсная модуляция (PWM) ...106
3.8. Приложение: Dagu Magician робот ...107
3.8.1. Требования ...111
3.8.2. Принципиальная схема ...112
3.8.3. Алгоритм управления роботом DaguMagician ...113
3.8.4. Тестирование алгоритма управления ...121
3.9. Выводы ...121
3.10. Задания ...122
Источники ...122

Глава 4 Искусственный интеллект и машинное обучение ...124
4.1. Обзор ...125
4.2. Краткая история развития искусственного интеллекта и машинного обучения ...127
4.3. Метод K ближайших соседей ...129
4.4. Дерево решений ...134
4.5. Приложение: классификатор KNN ...150
4.6. Приложение: дерево решений ...150
4.7. Выводы ...152
4.8. Задания ...152
Источники ...153

Глава 5 Нечеткая логика ...155
5.1. Обзор концепций ...155
5.2. Теория ...157

5.2.1. Установить цель, входы и выходы системы нечеткого управления....159
5.2.2. Размыть четкий сигнал датчика....159
5.2.3. Применение правил....162
5.2.4. Объединение активных правил и восстановление четкости выхода....162
5.3. Arduino-библиотека eFLL....163
5.3.1. Простой пример....163
5.3.2. Расширенный пример....169
5.4. Применение....172
5.5. Выводы....180
5.6. Задания....180
Источники....181

Глава 6 Нейронные сети....183
6.1. Обзор....184
6.2. Биологический нейрон....184
6.3. Персептрон....185
6.3.1. Обучение модели персептрона....187
6.3.2. Режим выполнения одиночного персептрона....195
6.3.3. Сортировка помидоров....197
6.4. Модель группы персептронов....201
6.4.1. Режим выполнения трех персептронов....210
6.5. Проблемы персептрона....211
6.6. Искусственная нейронная сеть (ANN)....211
6.6.1. Модель одиночного нейрона....211
6.6.2. Режим выполнения одиночного нейрона....216
6.6.3. Искусственные нейронные сети ANN....216
6.6.4. Сходимость ANN....231
6.7. Глубокие нейронные сети и глубокое обучение. Введение в программные инструменты....232
6.8. Приложение: управление роботом с помощью ANN....235
6.9. Выводы....236
6.10. Задания....236
Источники....238

Предметный указатель....239

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.

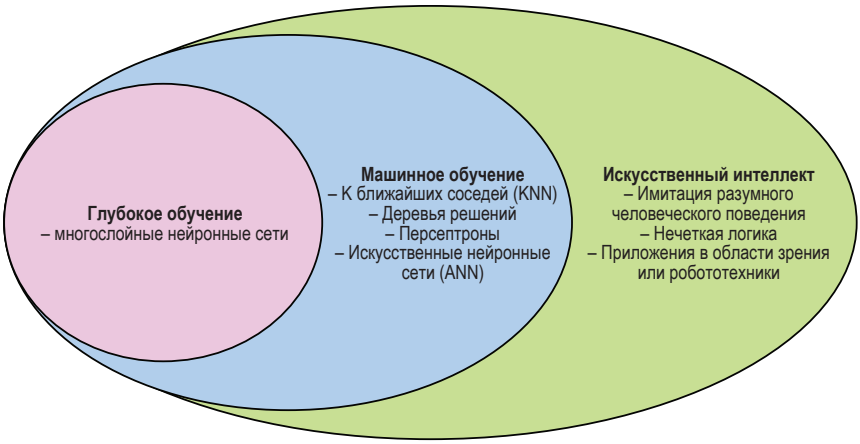


Рис. 1. Искусственный интеллект и машинное обучение [5]

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 разбираем 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 рассматриваем 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.

Благодарности

Эта книга стала возможной благодаря нескольким людям. Я хотел бы поблагодарить Массимо Банзи и команду разработчиков Arduino за поддержку и поощрение при написании первого издания данной книги.

Я также хотел бы выразить признательность Джоэлу Клейпулу (Joel Claypool) за его поддержку ряда писательских проектов. Его видение и опыт в издательском мире сделали эту книгу возможной. Джоэл «ушел на пенсию» в сентябре 2022 года после 40 с лишним лет на службе ВМС США в издательский мир. От имени множества писателей мы благодарим вас за предоставленную возможность стать публикуемыми авторами! Следующая глава в жизни Джоэла начинается с предстоящей поездки по оказанию помощи пострадавшим от урагана, и я посвящаю эту книгу тебе, мой друг.

Я также хотел бы поблагодарить Дхаранисварана Сундарамурти (Dharaneeswaran Sundaramurthy) из Total Service Books Production за его опыт в преобразовании законченного проекта в готовый продукт.

Наконец, что самое важное, я хотел бы поблагодарить моего лучшего друга в течение многих (почти 50) лет, мою жену Синди.

Стивен Ф. Барретт,
Ларами, Вайоминг, США,
январь, 2023

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 выступает 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.*

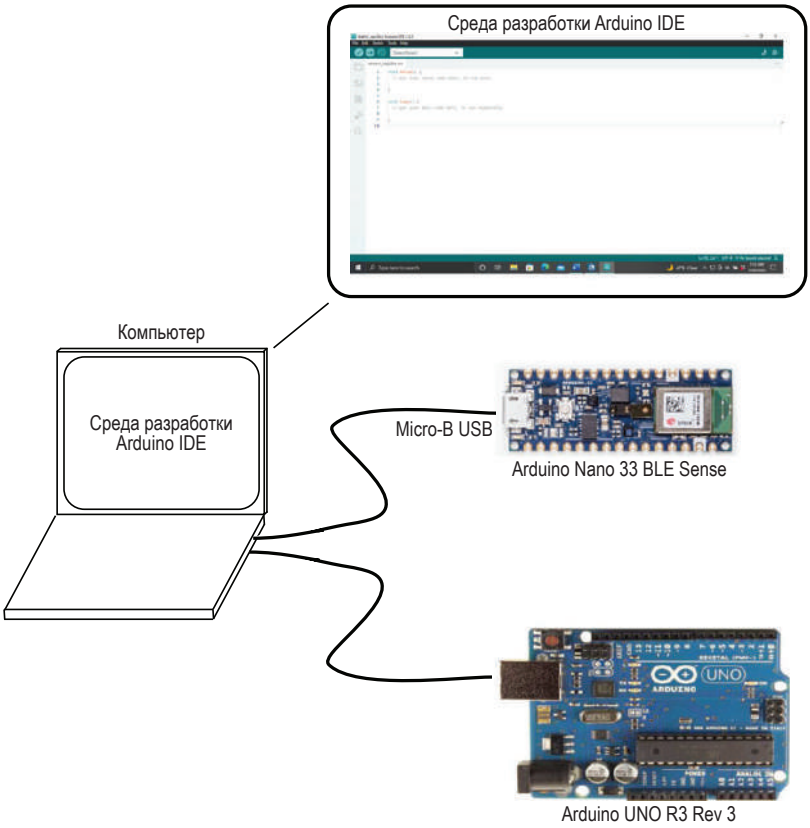


Рис. 1.1. Программирование платы Arduino. С разрешения команды Arduino (лицензия CC BY–NC–SA, www.arduino.cc)

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 расположен 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 контроллеры 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.

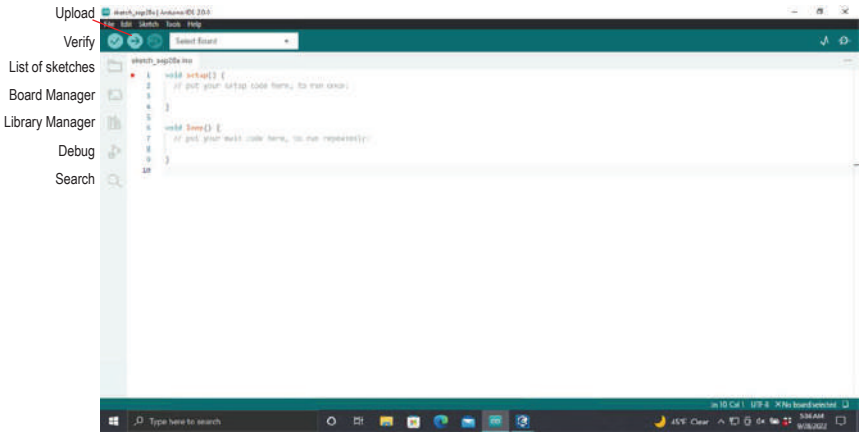


Рис. 1.2. Среда разработки Arduino IDE (www.arduino.cc)

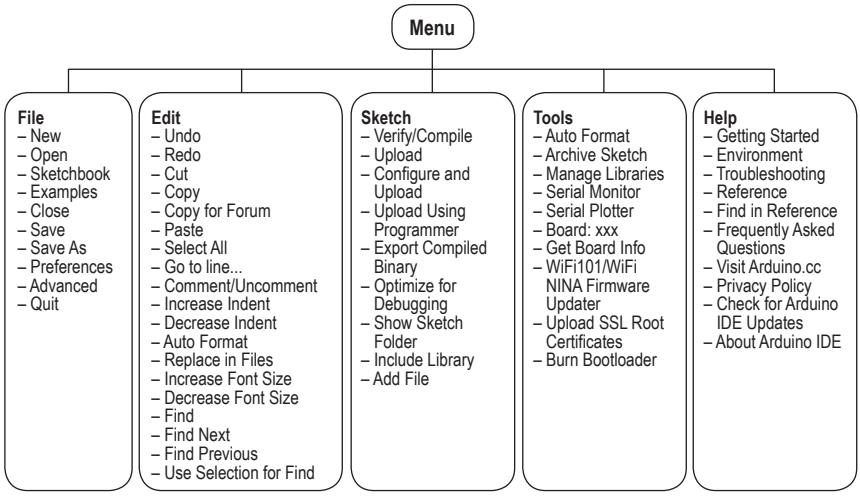


Рис. 1.3. Меню среды разработки Arduino IDE [2]

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. Далее 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.

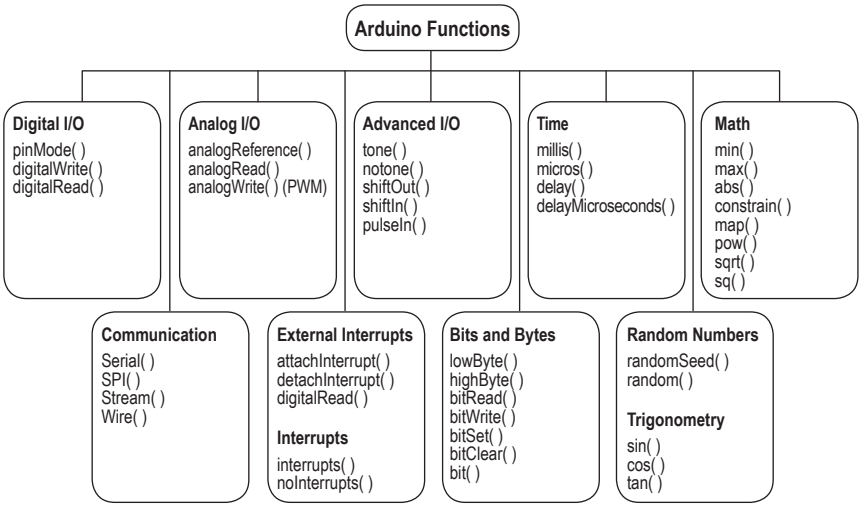


Рис. 1.4. Встроенные функции среды Arduino IDE (www.arduino.cc)

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.

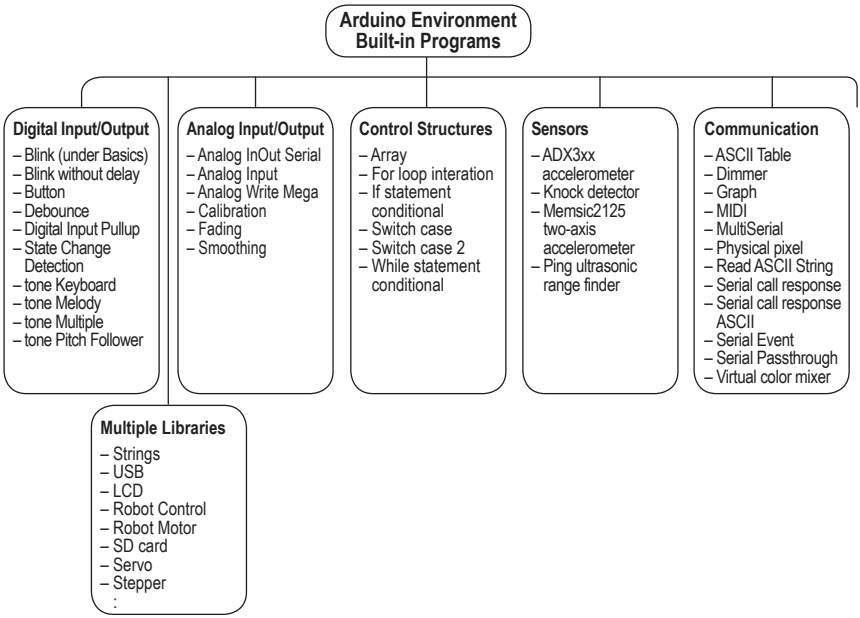


Рис. 1.5. Встроенные примеры программ среды Arduino IDE [2]

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 задается 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.

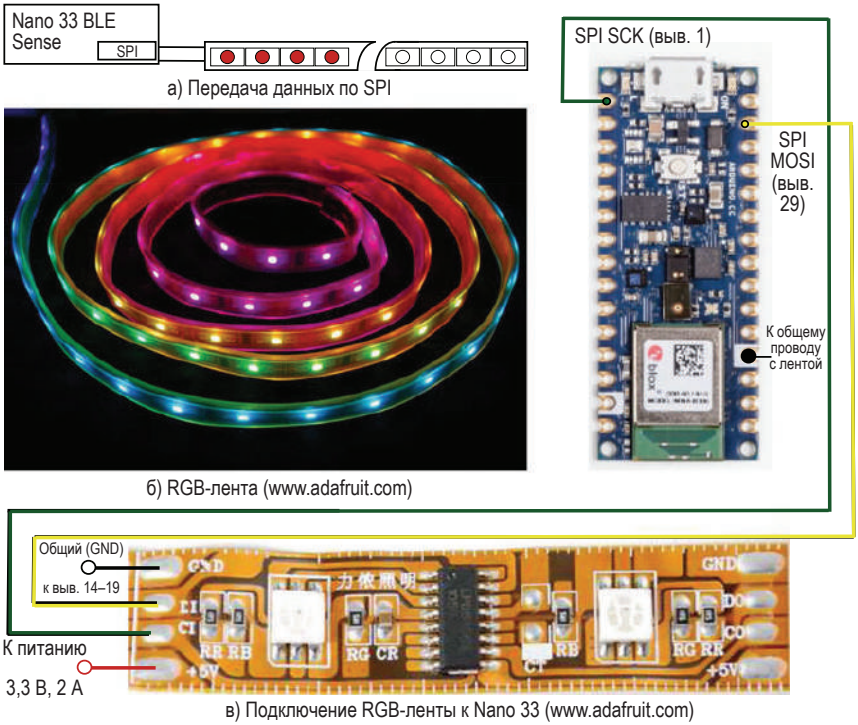


Рис. 1.6. Плата Nano 33 BLE Sense, управляющая светодиодной лентой. С разрешения компании Adafruit (www.adafruit.com) и команды Arduino (CC BY–NC–SA) (www.arduino.cc)

```
//*****************************************************************
//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.

ГЛАВА

2

Arduino Nano 33 BLE Sense

Прочитав эту главу, читатель должен уметь делать следующее:

- назвать и описать различные подсистемы периферийных устройств в составе процессора Nordic Semiconductor nRF52840;
- назвать и описать различные функции платы Arduino Nano 33 BLE Sense;
- описать своими словами основную теорию работы подсистем в составе процессора nRF52840 и платы Arduino Nano 33 BLE Sense;
- использовать Arduino IDE для программирования и выполнения скетчей с участием встроенных подсистем процессора nRF52840 и платы Arduino Nano 33 BLE Sense.

2.1. Обзор

В этой главе мы исследуем плату Arduino Nano 33 BLE Sense и ее процессор nRF52840. Начнем с обзора функций платы Arduino Nano 33 BLE Sense [3]. Затем мы рассмотрим мощный хорошо оснащенный процессор nRF52840 и связанные с ним периферийные подсистемы [11, 12]. Потом исследуем несколько периферийных устройств на плате Nano 33 BLE Sense. Для всех периферийных

устройств предоставляется краткая теория работы, обзор функциональности и примеры. Главу завершает расширенный пример, показывающий приложение Bluetooth BLE, – система мониторинга теплицы.

2.2. Плата Arduino Nano 33 BLE Sense

Плата Arduino Nano 33 BLE Sense [3] показана на рис. 2.1. **Nano 33 – это плата с напряжением питания 3,3 В постоянного тока**. Плата оснащена разъемом Micro-B USB (крайний слева на рис. 2.1), позволяющим программировать процессор с персонального компьютера (ПК) или ноутбука.

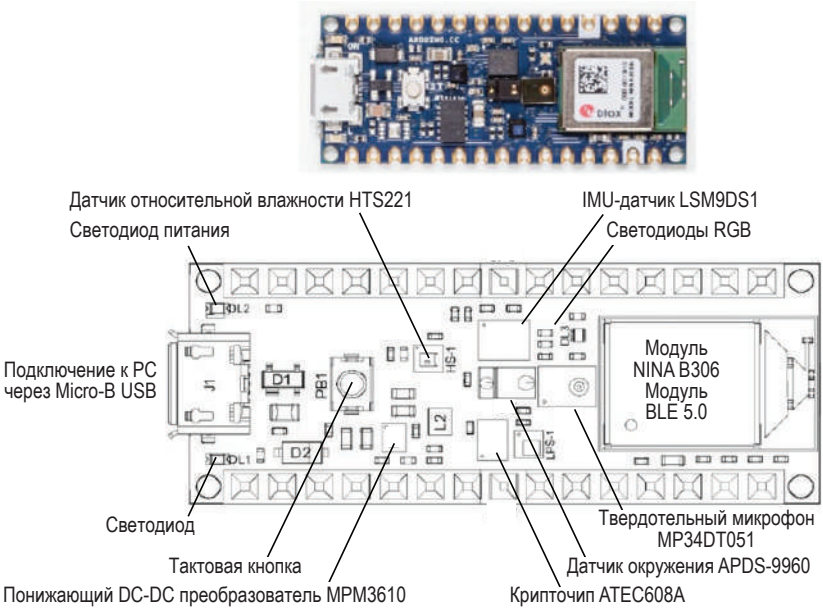


Рис. 2.1. Компоновка платы Arduino Nano 33 BLE Sense. С разрешения команды Arduino (CC BY–NC–SA, www.arduino.cc)

Плата оснащена рядом светодиодов, включая индикатор питания (на плате обозначен как L2), красный–зеленый–синий RGB-светодиоды (DL3) и встроенный светодиод (L1). Из главы 1 мы

знаем, что доступ к этим светодиодам осуществляется через специальные контакты.

Плата Arduino Nano 33 BLE Sense оснащена богатым набором датчиков [3], включающим:

- девятиосевой инерциальный измерительный блок (IMU) LSM9DS1 [9];
- барометр и датчик температуры LPS22HB [8];
- датчик относительной влажности HTS221 [7];
- цифровой датчик приближения, окружающего освещения, RGB-цвета и жестов APDS-9960 [1];
- цифровой твердотельный микрофон MP34DT05 [10];
- криптографический чип ATEC608A;
- понижающий DC-DC преобразователь MPM3610.

На плате Nano 33 находится модуль NINA B306. Он содержит процессор Nordic Semiconductor nRF52840. Это 32-битный процессор, работающий на частоте 64 МГц, с архитектурой ARM Cortex-M4F, оснащенный модулем с плавающей запятой (FPU). Процессор nRF52840 имеет 1 Мбайт флеш-памяти и 256 Кбайт оперативной памяти (ОЗУ, RAM).

Плата Nano 33 обеспечивает достаточную вычислительную мощность при очень небольших размерах. Вычислительная мощность в сочетании с периферийными устройствами и сенсорным блоком превращает этот микроконтроллер в идеальное устройство для приложений искусственного интеллекта и машинного обучения.

Модуль B306 с процессором nRF52840 также оснащен большим набором периферийных устройств, включая (см. [3]):

- подсистемы последовательной связи, в том числе универсальную последовательную шину USB, универсальный асинхронный приемник-передатчик UART, последовательный периферийный интерфейс SPI, интерфейс Quad SPI (QSPI), двухпроводной интерфейс TWI (I2C);
- аналого-цифровой преобразователь (ADC) с разрешением 12 бит, скоростью 200 тыс. выборок в секунду;
- 128-битный сопроцессор расширенного стандарта шифрования (AES);
- радиоинтерфейсы Bluetooth и Zigbee;

- модуль ближней бесконтактной связи (NFC[1]);
- возможность прямого доступа к памяти (DMA);
- криптографический ускоритель ARM CC310 Crytocell;
- каналы широтно-импульсной модуляции (PWM).

Доступ к этим подсистемам осуществляется через контакты платы Arduino Nano 33 BLE Sense, показанные на рис. 2.2.

2.3. Возможности Arduino Nano 33 BLE Sense

Закончив краткий обзор платы Arduino Nano 33 BLE Sense, мы теперь более подробно рассмотрим избранные функции и подсистемы. Как описано в предыдущем разделе, мы разделяем функции, относящиеся к модулю NINA B306, от функций платы Arduino Nano 33 BLE Sense (см. рис. 2.3). Мы начнем с изучения подсистем модуля NINA B306, а затем подсистем платы Arduino Nano 33. Для каждой подсистемы мы предоставляем соответствующую техническую информацию и примеры, где это необходимо.

2.4. Подсистемы модуля NINA B306

В этом разделе мы исследуем периферийные устройства, имеющиеся в модуле NINA B306. Модуль содержит Nordic Semiconductor nRF52840 – 32-битный процессор, работающий на частоте 64 МГц, с архитектурой ARM Cortex-M4F, оснащенный модулем с плавающей запятой (FPU). Кроме того, NINA B306 оснащен специализированной функциональностью, помогающей эффективно выполнять

[1] NFC (Near field communication, «связь в близком поле») – стандарт связи устройств на расстояниях 10 см и менее, представляющий собой расширение стандарта бесконтактных карт. В отличие от Bluetooth, может осуществлять связь только «точка–точка»; в отличие от радиочастотной идентификации (RFID), связь полноценно двухсторонняя. Всем известный пример применения NFC в быту – эмуляция бесконтактных банковских карт с помощью мобильного устройства при оплате покупок или проезда в общественном транспорте. – *Прим. перев.*

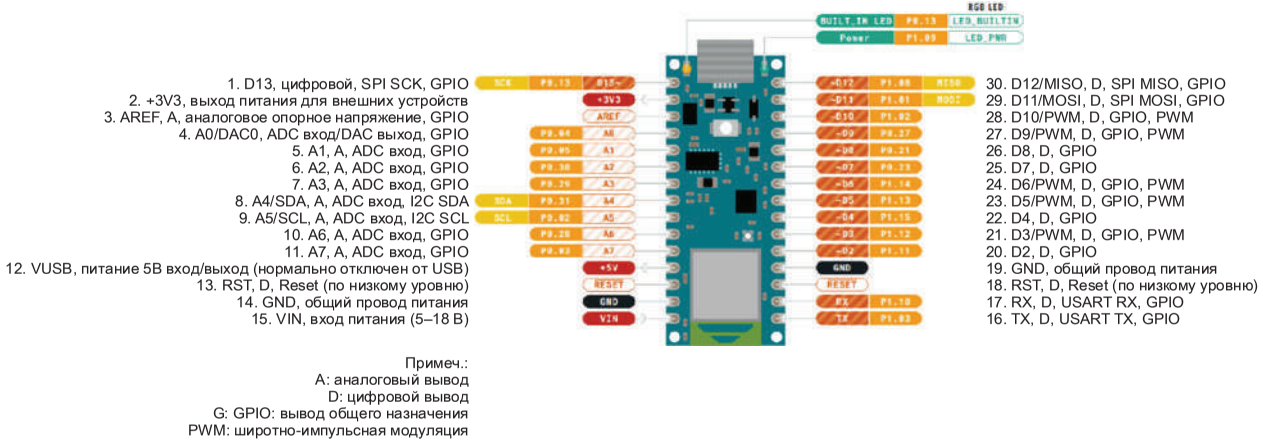


Рис. 2.2. Разводка выводов платы Arduino Nano 33 BLE Sense. С разрешения команды Arduino (CC BY–NC–SA, www.arduino.cc)

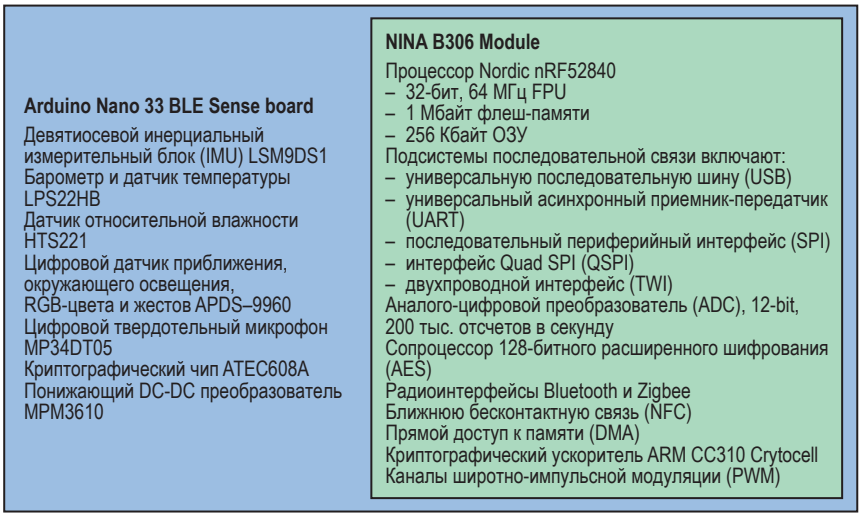


Рис. 2.3. Подсистемы Arduino Nano 33 BLE Sense (процессор nRF52840)

программы искусственного интеллекта и машинного обучения, включающей:

- набор инструкций цифровой обработки сигналов (DSP);
- прямой доступ к памяти (DMA) для эффективного доступа ЦП и периферийных устройств к памяти;
- аппаратный арифметический делитель для ускорения операции деления;
- операции умножения-накопления (multiply-accumulate, MAC), выполняющиеся за один такт.

32-разрядная архитектура процессора обеспечивает широкий диапазон возможностей обработки целых и действительных чисел (чисел с плавающей запятой). Процессор оснащен модулем с плавающей запятой для улучшения их обработки. Тактовая частота процессора 64 МГц обеспечивает расширенные возможности по сравнению с другими продуктами Arduino. Например, процессор классического Arduino UNO R3, на котором установлен контроллер ATmega328, работает на частоте 16 МГц[1].

[1] Главное отличие nRF52840 от классического Arduino-контроллера ATmega328, позволяющее Arduino Nano 33 работать значительно быстрее, – не в повышенной частоте, а в четыре раза большей разрядности: 32 бита у nRF52840 против 8 у контроллеров ATmega. – *Прим. перев.*

2.4.1. Память модуля B306

B306 оснащен двумя разделами основной памяти: электрически стираемой программируемой постоянной памятью (флеш-памятью) и статической оперативной памятью (SRAM). Процессор оснащен 1 мегабайтом флеш-памяти и 256 килобайтами оперативной памяти[1]. Мы обсудим каждый компонент памяти по очереди.

2.4.1.1. Программируемая флеш-память

Большая часть флеш-памяти используется для хранения программ. Ее можно стереть и запрограммировать как единое целое. Кроме того, если программе требуется большая таблица констант, ее можно включить в программу как глобальную переменную и записать во флеш-память вместе с остальной частью программы. Флеш-память является энергонезависимой, т. е. содержимое памяти сохраняется даже при отключении питания микроконтроллера. Модуль B306 оснащен встроенной перепрограммируемой флеш-памятью емкостью 1 Мбайт.

2.4.1.2. Статическая память с произвольным доступом (SRAM)[2] в модуле B306

Статическая оперативная память энергозависима. То есть если микроконтроллер теряет питание, содержимое SRAM теряется. Ее можно записывать и читать во время выполнения программы. Модуль B306 оснащен SRAM объемом 256 Кбайт. Небольшая часть SRAM отведена для регистров общего назначения, используемых процессором, а также для подсистем ввода/вывода и периферийных подсистем микроконтроллера. Во время выполнения программы SRAM используется для хранения программных инструкций, глобальных переменных, поддержки динамического распреде-

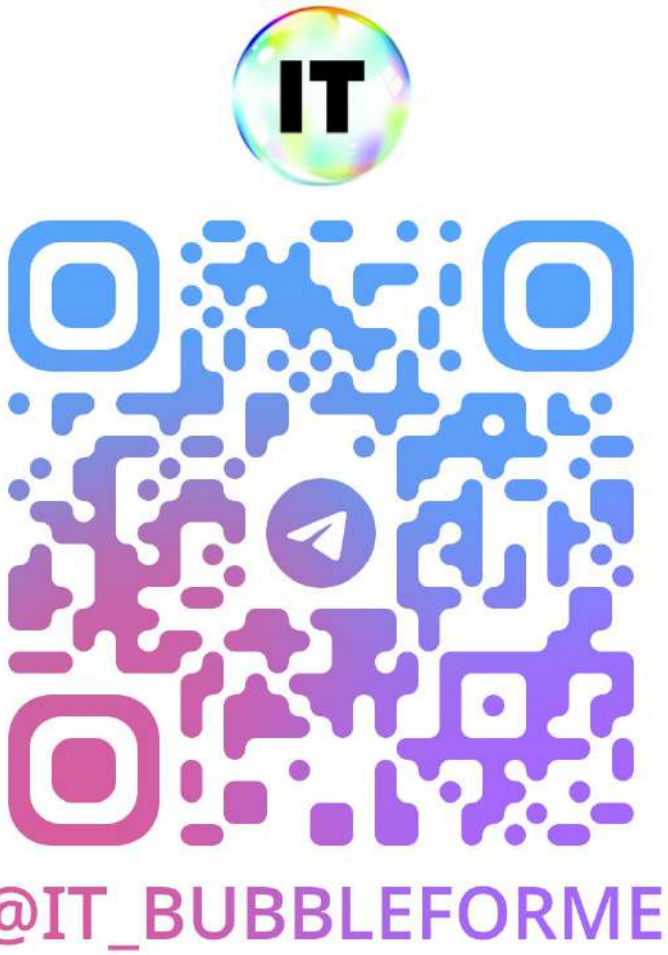
---

[1] В отличие от классических Arduino, Arduino Nano 33 не содержит EEPROM – энергонезависимой памяти с побайтным доступом. Этот недостаток при необходимости преодолевается имитацией EEPROM с помощью выделения части программной флеш-памяти. В разделе 6.3.1 встретится пример с использованием EEPROM, но он будет ориентирован на выполнение на классическом Arduino UNO R3. – *Прим. перев.*

[2] В русскоязычной литературе термину «память с произвольным доступом» (Random Access Memory, RAM) соответствует термин «оперативное запоминающее устройство», ОЗУ. – *Прим. перев.*

Эта книга из Telegram-канала

@IT_BUBBLEFORME



Читай бесплатно в Telegram книги по IT, программированию и ИИ

Сканируй QR или переходи по ссылке

https://t.me/IT_bubbleForMe

ления переменных в памяти и предоставления места для стека. Большой объем SRAM особенно полезен в некоторых приложениях искусственного интеллекта и машинного обучения[1].

2.5. Периферийные устройства модуля NINA B306

В этом разделе мы даем краткий обзор внутренней периферии модуля B306. Следует подчеркнуть, что эти устройства представляют собой внутренние системы, заключенные в пределах чипа микроконтроллера. Встроенные периферийные устройства позволяют микроконтроллеру выполнять сложные и разнообразные задачи.

2.5.1. Каналы широтно-импульсной модуляции (PWM)

Nano 33 BLE Sense оснащена пятью каналами широтно-импульсной модуляции (PWM). Выход PWM в принципе может быть обеспечен на цифровых контактах 1–13 и аналоговых контактах A0–A7. Мы ограничиваем использование контактами 21, 23, 24, 27 и 28, как показано на схеме разводки выводов (рис. 2.2). Базовая частота сигнала PWM для платы Nano 33 BLE Sense установлена на уровне 500 Гц.

Сигнал PWM, т. е. сигнал, модулированный по ширине импульса, характеризуется фиксированной частотой и изменяющимся коэффициентом заполнения (duty cycle). Коэффициент заполнения – это процент времени, в течение которого повторяющийся сигнал имеет высокий логический уровень в течение периода сигнала. Формально это можно выразить так:

коэфф. заполнения (%) = (*время высокого уровня* / *период*) × 100%.

Для генерации PWM-сигнала в Arduino IDE используется функция `analogWrite()`. Функция вызывается с номером нужного вывода PWM (`pin`) и желаемым значением коэффициента заполнения PWM

[1] В процессоре ATmega128 на классических платах Arduino объем SRAM значительно меньше и равен 2 Кбайтам. – *Прим. перев.*

(value). Значение коэффициента заполнения указывается в диапазоне от нуля (0) до 100%-ного значения (равного 255):

```
analogWrite(pin, value);
```

Пример **В примере далее интенсивность свечения светодиода, подключенного к контакту 21 платы Nano 33 BLE Sense (цифровой вывод D3), медленно увеличивается, а затем уменьшается с использованием метода PWM. Обратите внимание на необходимость применения резистора сопротивлением 220 Ом последовательно со светодиодом. Мы обсудим подробнее метод подключения светодиодов в главе 3.**

```
//*******************************************************
//pwm_LED_test: the intensity of an LED, connected to
//Nano 33 BLE Sense pin 21, is slowly increased and then
//decreased using PWM techniques.
//
//Notes:
//- Provide a 220 Ohm resistor in series with the LED
// to common ground with the Nano 33.
//- Nano 33 BLE Sense pin 21 is specified as digital D3
// in the Arduino IDE sketch.
//*******************************************************
int ext_red_LED_pin = 3; //LED on D3 (physical pin 21)
void setup()
{
  pinMode(ext_red_LED_pin, OUTPUT); //sets the pin as output
}
void loop()
{
  int LED_int = 0;
                            //increase intensity
  for(LED_int=0; LED_int<=255; LED_int=LED_int+5)
    {                       //0% (0) to 100% (255)
      analogWrite(ext_red_LED_pin, LED_int);
      delay(100); //delay 100 ms
    }
  LED_int = 255;
                            //decrease intensity
  for(LED_int=255; LED_int>=0; LED_int=LED_int-5)
    {                       //100% (255) to 0% (0)
      analogWrite(ext_red_LED_pin, LED_int);
      delay(100); //delay 100 ms
    }
  delay(100);               //delay 100 ms
}
//*******************************************************
```

Сигналы PWM используются в самых разных приложениях, включая управление положением серводвигателя и управление

скоростью вращения двигателя постоянного тока. Мы рассмотрим подобные приложения в следующей главе.

2.5.2. Последовательная связь

Arduino Nano 33BLE Sense оснащена множеством различных подсистем последовательной связи, включая универсальный синхронный/асинхронный последовательный приемник и передатчик (USART), последовательный периферийный интерфейс (SPI) и двухпроводный последовательный интерфейс (TWI[1]). Общим для этих систем является последовательная передача данных. При последовательной передаче данные передаются по одному биту от передатчика к приемнику. Подсистемы последовательной связи обычно используются для добавления дополнительных периферийных устройств и связи с ними.

2.5.2.1. USART

Последовательный порт USART может использоваться для полнодуплексной (двусторонней) связи между приемником и передатчиком. Это достигается за счет оснащения Nano 33 независимым оборудованием для передатчика и приемника. USART обычно используется для асинхронной связи (UART), то есть между передатчиком и приемником нет линии общих тактов, которые могли бы синхронизировать их друг с другом[2]. В UART для поддержания синхронизации между передатчиком и приемником при асинхронной связи в начале и конце каждого байта данных в последовательности передачи используются стартовые и стоповые кадровые биты.

USART в Nano 33 достаточно гибок. Он имеет возможность установки различных скоростей передачи данных, измеряемых в бодах (*baud*) или битах в секунду (*bps*)[3]. USART также может быть настро-

---

[1] Для двухпроводного интерфейса TWI более распространено название I2C, которое автор и будет употреблять далее. – *Прим. перев.*

[2] В подавляющем большинстве применений «синхронные» возможности последовательного порта не используются, потому сокращения UART и USART здесь употребляются в случайном порядке и означают одно и то же. – *Прим. перев.*

[3] В Arduino для скорости передачи данных принято употреблять термин «бод» (*baud rate*), имея в виду количество битов в секунду (*bps*). Терминологически это неверно («боды» в теории связи – количество не битов, а неких «посылок» в секунду), но на всякий случай следует иметь в виду, что в данном случае «боды» и «биты в секунду» – синонимы. – *Прим. перев.*

ен на несколько разрядностей данных и разные скорости передачи данных. Кроме того, он оснащен аппаратно генерируемым битом четности (четным или нечетным) и аппаратным обеспечением проверки четности на приемнике. Один бит четности позволяет обнаружить ошибку в один бит в байте данных.

Пример **В этом примере мы оборудуем Nano 33 жидкокристаллическим дисплеем (ЖК-дисплеем). ЖК-дисплей – это устройство вывода для отображения текстовой информации, см. рис. 2.5. ЖК-дисплеи бывают самых разных конфигураций, включая многосимвольный и многострочный формат. Широко распространены ЖК-дисплеи формата 16×2, имеющие возможность отображать две строки по 16 символов каждая.**

Символы передаются на ЖК-дисплей в формате американского стандартного кода обмена информацией (American Standard Code for Information Interchange, ASCII) по одному символу или управляющей команде за раз. ASCII – это стандартизированный семибитный метод кодирования буквенно-цифровых данных. Он применяется уже много десятилетий, поэтому некоторые символы и действия, перечисленные в таблице ASCII, сегодня широко не используются, однако ASCII по-прежнему остается наиболее распространенным методом кодирования буквенно-цифровых данных. Код ASCII показан на рис. 2.4. Использование таблицы проиллюстрируем примером: заглавная буква «G» кодируется в ASCII как 0x47 (символ «0x» указывает на представление шестнадцатеричного числа).

Unicode – международный аналог ASCII. Он обеспечивает стандартизированный 16-битный формат кодирования для всех письменных языков мира. ASCII – это подмножество Unicode. Заинтересованный читатель может получить дополнительную информацию об этом стандартизированном формате кодирования на сайте Unicode по адресу: www.unicode.org.

ЖК-дисплеи выпускаются с последовательным или параллельным интерфейсом для обмена данными с главным микроконтроллером. Для ЖК-дисплея с параллельным интерфейсом требуется восьмибитный тракт данных и две линии управления. Многие ЖК-дисплеи с параллельной конфигурацией также могут быть сконфигурированы для четырехбитного пути передачи данных, что позволяет сэкономить несколько драгоценных выводов микроконтроллера. Небольшой микроконтроллер, установленный на задней панели ЖК-дисплея, преобразует символы данных ASCII

и сигналы управления для правильного отображения символов. Большинство дисплеев питаются от напряжения 5 В, но некоторые производители предлагают дисплеи, совместимые с напряжением 3,3 В постоянного тока.

		Старшие 4 бита (MSD)							
		0x0_	0x1_	0x2_	0x3_	0x4_	0x5_	0x6_	0x7_
Младшие 4 бита (LSD)	0x_0	NUL	DLE	SP	0	@	P	`	p
	0x_1	SOH	DC1	!	1	A	Q	a	q
	0x_2	STX	DC2	“	2	B	R	b	r
	0x_3	ETX	DC3	#	3	C	S	c	s
	0x_4	EOT	DC4	$	4	D	T	d	t
	0x_5	ENQ	NAK	%	5	E	U	e	u
	0x_6	ACK	SYN	&	6	F	V	f	v
	0x_7	BEL	ETB	‘	7	G	W	g	w
	0x_8	BS	CAN	(	8	H	X	h	x
	0x_9	HT	EM	)	9	I	Y	i	y
	0x_A	LF	SUB	*	:	J	Z	j	z
	0x_B	VT	ESC	+	;	K	[	k	{
	0x_C	FF	FS	‘	<	L	\	l	|
	0x_D	CR	GS	-	=	M	]	m	}
	0x_E	SO	RS	.	>	N	^	n	~
	0x_F	SI	US	/	?	O	_	o	DEL

Рис. 2.4. ASCII-код. Код ASCII используется для кодирования буквенно-цифровых символов («0x» означает шестнадцатеричную запись на языке программирования C)

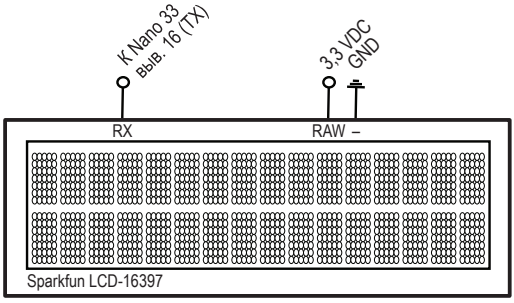


Рис. 2.5. ЖК-дисплей с последовательным интерфейсом

Строка	Позиция символа (*n*)
1	0–15
2	64–79

Примечание: позиция символа указывается как 0x80+*n*.

Код команды	Команда
0x01	Очистка дисплея
0x14	Курсор на одно место вправо
0x10	Курсор на одно место влево
0x80+*n*	Курсор на позиции *n*

Примечание: команда предваряется кодом 0xFE (254_{10}).

Чтобы еще больше сэкономить контакты ввода/вывода микроконтроллера, можно использовать ЖК-дисплей с последовательным интерфейсом. В самом экономном варианте последовательный ЖК-дисплей уменьшает количество необходимых контактов микроконтроллера для интерфейса до одного. Данные и управляющая информация передаются на ЖК-дисплей через асинхронный последовательный канал связи USART по одной линии от микроконтроллера к дисплею (формат 8 бит, 1 стоповый бит, без контроля четности, 9600 бод).

В этом примере ЖК-дисплей Sparkfun LCD–16397[1] (3,3 В постоянного тока, 16 строк на 2 символа) подключен к Nano 33 BLE Sense. Связь между Nano 33 и этим дисплеем осуществляется посредством одного соединения со скоростью 9600 бит в секунду с использованием встроенного USART.

В разделе «Приложения» главы 3 мы настраиваем мини-робота для автономного перемещения по лабиринту. Робот будет отображать состояние стены лабиринта на ЖК-дисплее. В примере скетча ниже мы используем смоделированные данные стены лабиринта для отображения на ЖК-дисплее. Обратите внимание, что встроенный USART в коде скетча обозначается как «Serial1».

```
//*****************************************************************
//LCD_example
//
//Sparkfun LCD-16397, 3.3 VDC, 16x2 character display
//- Nano 33 BLE Sense, USART TX pin 16 is connected to LCD USART RX pin
//- Provide 3.3 VDC power to the LCD
//*****************************************************************
int left_IR_sensor_value = 0;   //variable for left IR sensor
int center_IR_sensor_value = 0; //variable for center IR sensor
int right_IR_sensor_value = 0;  //variable for right IR sensor
void setup()
{
  Serial1.begin(9600);          //Baud rate: 9600 Baud
  delay(500);                   //Delay for display
}
void loop()
{
//read analog output from IR sensors Â- simulated maze wall data
  left_IR_sensor_value = left_IR_sensor_value + 1;
```

[1] Дисплей LCD–16397 можно прибрести в отечественных условиях, например в интернет-магазине «Чип и Дип» (https://www.chipdip.ru/product0/8006766087). Отметим, что кроме одностороннего UART этот дисплей также может управляться через другие последовательные интерфейсы I2C (TWI) или SPI. – *Прим. перев.*

```
  center_IR_sensor_value = center_IR_sensor_value + 2;
  right_IR_sensor_value = right_IR_sensor_value + 3;
//Clear LCD
//Cursor to line one, character one
  Serial1.write(254);           //Command prefix
  Serial1.write(128);           //Command
//clear display
  Serial1.write(" ");
  Serial1.write(" ");
//Cursor to line one, character one
  Serial1.write(254);           //Command prefix
  Serial1.write(128);           //Command
  Serial1.write("Left Ctr Right");
  delay(50);
  Serial1.write(254);            //Command to LCD
  delay(5);
  Serial1.write(192);            //Cursor line 2, position 1
  delay(5);
  Serial1.print(left_IR_sensor_value);
  delay(5);
  Serial1.write(254);            //Command to LCD
  delay(5);
  Serial1.write(198);            //Cursor line 2, position 8
  delay(5);
  Serial1.print(center_IR_sensor_value);
  delay(5);
  Serial1.write(254);            //Command to LCD
  delay(5);
  Serial1.write(203);            //Cursor line 2, position 13
  delay(5);
  Serial1.print(right_IR_sensor_value);
  delay(5);
  delay(500);
}
//*****************************************************************
```

2.5.2.2. Последовательный периферийный интерфейс (SPI)

Последовательный периферийный интерфейс (Serial Peripheral Interface, SPI) обеспечивает в Nano 33 BLE Sense двустороннюю последовательную связь между передатчиком и приемником. В системе SPI передатчик и приемник используют общий источник синхронизации. Это требует дополнительной линии синхронизации между передатчиком и приемником, но дает более высокие скорости передачи данных по сравнению с USART.

Система SPI обеспечивает быстрый и эффективный обмен данными между микроконтроллерами и периферийными устройства-

ми. Существует множество SPI-совместимых внешних устройств, расширяющих возможности микроконтроллера. Например, с помощью системы SPI к микроконтроллеру можно добавить жидкокристаллический дисплей или цифроаналоговый преобразователь.

Функционирование SPI. SPI можно рассматривать как синхронный 16-битный сдвиговый регистр, одна 8-битная половина которого находится в передатчике, а другая в приемнике (см. рис. 2.6). Передатчик назначается ведущим (master), поскольку он обеспечивает источник синхронизации между передатчиком и приемником. Приемник обозначен как ведомый (slave). Ведомое устройство выбирается для приема с помощью перевода его линии выбора ведомого устройства ($\overline{SS}$) в низкий уровень. Когда линия $\overline{SS}$ устанавливается в низкий уровень, активируется доступ к сдвиговому регистру ведомого устройства.

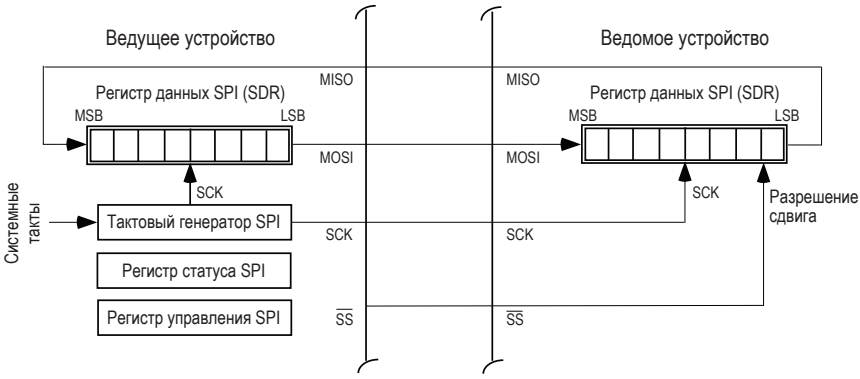


Рис. 2.6. Устройство SPI

Передача SPI инициируется с помощью загрузки байта данных в регистр данных SPI на стороне ведущего устройства. В это время генератор тактовых импульсов SPI на стороне ведущего подает тактовые импульсы и ведущему, и ведомому устройствам через вывод SCK. Один бит сдвигается из назначенного ведущим регистра сдвига на выводе «выход ведущего – вход ведомого» (Master Out Slave In, MOSI) микроконтроллера при каждом импульсе SCK.

Данные принимаются на одноименный вывод (MOSI) устройства, назначенного ведомым. Одновременно один бит сдвигается с вывода «вход ведущего – выход ведомого» (Master In Slave Out, MISO) ведомого устройства на (одноименный!) вывод MISO ве-

дущего устройства[1]. После восьми тактовых импульсов по линии SCK между ведущим и ведомым устройствами происходит обмен одним байтом данных.

Связанные с SPI контакты на Arduino Nano 33 BLE Sense включают:

- контакт 29, SPI MOSI, «выход ведущего – вход ведомого» (Master Out Slave In);
- контакт 30, SPI MISO, «вход ведущего – выход ведомого» (Master In Slave Out);
- контакт 1, SPI SCK.

Чтобы настроить Arduino Nano 33 BLE Sense для работы SPI, используются следующие команды Arduino IDE:

- `SPI.begin()` – вызывается внутри функции `setup()`;
- `SPI.beginTransaction (SPISettings (SPIspeed, SPIbitdirection, SPImode));`
- `SPI.transfer(data);`
- `SPI.endTransaction()`.

В главе 1 мы использовали интерфейс SPI для отправки данных RGB на отдельные светодиоды в светодиодной ленте, совместимой с SPI.

Пример **В этом примере мы настраиваем ЖК-дисплей Sparkfun LCD–16397. Необходимые соединения между Nano 33 и ЖК-дисплеем, показанные на рис. 2.7, включают в себя:**

- **Nano 33 контакт 1 SPI SCK к SCK на ЖК-дисплее;**
- **Nano 33 контакт 29 SPI MOSI ко входу последовательных данных SDI (Serial Data In) на ЖК-дисплее;**
- **вывод «выбор кристалла» ЖК-дисплея $\overline{\text{CS}}$ к общему проводу GND.**

Ниже представлен скетч Arduino для связи с LCD–16397 с использованием системы SPI. Несколько интересных моментов относительно скетча:

---

[1] Следует подчеркнуть, что в последовательных интерфейсах SPI и I2C/TWI (см. далее) при подключении между собой объединяются *одноименные* линии связи, в то время как в описанном ранее UART/USART соединение *перекрестное*: RX одного устройства соединяется с TX второго и наоборот. – *Прим. перев.*

- в документации SparkFun для ЖК-дисплея указаны следующие настройки SPI: скорость (speed) = 100 000, MSBFIRST, SPI_MODE0;
- обратите внимание на использование функции `send_int_via_SPI()` для отправки трехзначного целочисленного значения на ЖК-дисплей. Функция выделяет каждую цифру и отправляет на ЖК-дисплей ее ASCII-эквивалент.

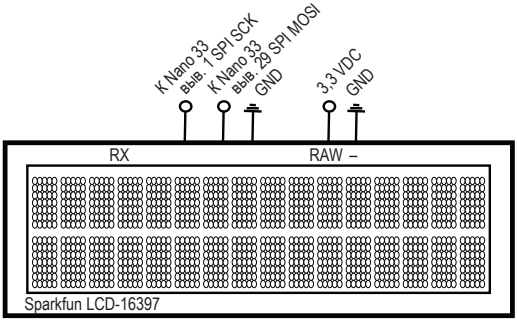


Рис. 2.7. Подключение ЖК-дисплея через SPI

```
//*********************************************************************
//LCD_SPI_example
//
//The robot is equipped with a Sparkfun LCD-16397
// - Nano 33 SCK pin 1 to SCK on LCD
// - Nano 33 SPI MOSI pin 29 to SDI (Serial Data In) on LCD
// - Ground LCD chip select (\CS)
//- Provide 3.3 VDC power to the LCD
//*********************************************************************
#include <SPI.h>
int left_IR_sensor_value = 0; //variable for left IR sensor
int center_IR_sensor_value = 0; //variable for center IR sensor
int right_IR_sensor_value = 0; //variable for right IR sensor
void setup()
{
  SPI.begin();
  delay(500); //Delay for display
}
void loop()
{
  SPI.beginTransaction(SPISettings(100000, MSBFIRST, SPI_MODE0));
//read analog output from IR sensors
  left_IR_sensor_value = left_IR_sensor_value + 1;
  center_IR_sensor_value = center_IR_sensor_value + 2;
```

```
  right_IR_sensor_value = right_IR_sensor_value + 3;
//Clear LCD
//Cursor to line one, character one
  SPI.transfer(254);                //Command prefix
  SPI.transfer(128);                //Command
//clear display
  spiSendString(" ");
  spiSendString(" ");
//Cursor to line one, character one
  SPI.transfer(254);                //Command prefix
  SPI.transfer(128);                //Command
  spiSendString("Left Ctr Right");
  delay(50);
  SPI.transfer(254);                //Command to LCD
  delay(5);
  SPI.transfer(192);                //Cursor line 2, position 1
  delay(5);
  transmit_int_via_SPI(left_IR_sensor_value);
  delay(5);
  SPI.transfer(254);                //Command to LCD
  delay(5);
  SPI.transfer(198);                //Cursor line 2, position 8
  delay(5);
  transmit_int_via_SPI(center_IR_sensor_value);
  delay(5);
  SPI.transfer(254);                //Command to LCD
  delay(5);
  SPI.transfer(203);                //Cursor line 2, position 13
  delay(5);
  transmit_int_via_SPI(right_IR_sensor_value);
  SPI.endTransaction();
  delay(500);
}
//*********************************************************************
void spiSendString(char* data)
{
  for(byte x = 0; data[x] != '\0'; x++)//send chars to end of string
  {
    SPI.transfer(data[x]);
  }
}
//*********************************************************************
void transmit_int_via_SPI(unsigned int num_to_convert)
{
  unsigned int hundreds_place, tens_place, ones_place;
  char hundreds_place_char, tens_place_char, ones_place_char;
  hundreds_place = (unsigned int)(num_to_convert/100);
  hundreds_place_char = (char)(hundreds_place + 48);
  SPI.transfer(hundreds_place_char);
  delay(5);
  tens_place = (unsigned int)((num_to_convert-(hundreds_place * 100))/10);
```

```
  tens_place_char = (char)(tens_place + 48);
  SPI.transfer(tens_place_char);
  delay(5);
  ones_place=(unsigned int)(num_to_convert-(hundreds_place*100)-(tens_place*10));
  ones_place_char = (char)(ones_place + 48);
  SPI.transfer(ones_place_char);
  delay(5);
}
//*****************************************************************
```

2.5.2.3. Интерфейс I2C (TWI)

Интерфейс I2C (Inter–Integrated Circuit – «соединение интегральных схем») позволяет разработчику связать различные устройства (микроконтроллеры, преобразователи, дисплеи, запоминающие устройства и т. д.) в единую систему, используя двухпроводную схему соединения. I2C позволяет соединить между собой максимум 127 устройств. Каждое устройство имеет свой уникальный адрес и может как передавать, так и принимать по двухпроводной шине на частотах до 400 кГц. Это позволяет устройству свободно обмениваться информацией с другими устройствами в сети на небольшой территории. Устройства в этой сети соединяются всего двумя проводниками: для обмена данными (SDA) и общими тактовыми импульсами (SCL).

Пример **Подключение ЖК-дисплея Sparkfun LCD-16397 по I2C. На плате Nano 33 вывод I2C SDA (контакт 8) подключается к контакту DA LCD-дисплея. Вывод Nano 33 I2C CLK (контакт 9) подключается к контакту CL LCD-дисплея. Вывод «выбор кристалла» LCD-дисплея (\CS) подключается в общему проводу GND. Вывод питания 3,3 В на ЖК-дисплее подключается к питанию 3,3 В постоянного тока на выводе 3.3VDC. I2C-адрес ЖК-дисплея по умолчанию 0x72.**

```
//*****************************************************************
//LCD_I2C_example
//
//The robot is equipped with a Sparkfun LCD-16397
// - Nano 33 I2C SDA pin 8 to LCD DA pin
// - Nano 33 I2C CLK pin 9 to LCD CL
// - Ground LCD chip select (\CS)
//- Provide 3.3 VDC power to the LCD
//- LCD default I2C address: 0x72
//*****************************************************************
#include <Wire.h>
#define LCD_I2C_addr 0x72        //default address of LCD
int cycles = 0;
int left_IR_sensor_value = 0;    //variable for left IR sensor
```

```
int center_IR_sensor_value = 0;  //variable for center IR sensor
int right_IR_sensor_value = 0;   //variable for right IR sensor
void setup()
{
  Wire.begin();                   //Join I2C bus - master mode
  Wire.beginTransmission(LCD_I2C_addr);
  Wire.write('|');               //LCD setting mode
  Wire.write('-');               //clear display command
  Wire.endTransmission();
}
void loop()
{
//read analog output from IR sensors
  left_IR_sensor_value = left_IR_sensor_value + 1;
  center_IR_sensor_value = center_IR_sensor_value + 2;
  right_IR_sensor_value = right_IR_sensor_value + 3;
  I2CSendValue(left_IR_sensor_value, center_IR_sensor_value, right_IR_sensor_value);
  delay(500);                          //delay
}
//*****************************************************************
void I2CSendValue(int value1, int value2, int value3)
{
  Wire.beginTransmission(LCD_I2C_addr); //transmit to LCD
  Wire.write('|');                      //LCD setting mode
  Wire.write('-');                      //clear display command
  Wire.print("Left Ctr Right");
  Wire.print(value1);
  Wire.print(" ");
  Wire.print(value2);
  Wire.print(" ");
  Wire.print(value3);
  Wire.print(" ");
  Wire.endTransmission();               //Stop I2C transmission
}
//*****************************************************************
```

2.5.2.4. Аналого-цифровой преобразователь ADC

Целью работы аналого-цифрового преобразователя (Analog to Digital Converter, ADC) является точное представление аналоговых сигналов в цифровом виде. С этой целью необходимо объединить три процедуры обработки сигналов: дискретизацию, квантование и кодирование.

Прежде чем преобразование АЦП начнется, необходимо преобразовать физический сигнал в электрический с помощью преобразователя. Преобразователь – это электрическая и/или механическая система, которая преобразует физические сигналы в электрические или электрические сигналы в физические.

В зависимости от цели мы подразделяем преобразователь на входной преобразователь или выходной преобразователь. Если преобразование происходит из физического состояния в электрическое, мы называем его входным преобразователем. Например, датчик температуры считается входным преобразователем. Выходной преобразователь преобразует электрические сигналы в физические явления. Например, ЖК-дисплей или двигатель будет считаться выходным преобразователем.

Важно тщательно спроектировать интерфейс между датчиками и микроконтроллером, чтобы обеспечить правильную работу. Плохо спроектированный интерфейс может привести к неправильной работе или сбою встроенной системы. Конкретные методы интерфейса подключения входных и выходных преобразователей обсуждаются в главе 3.

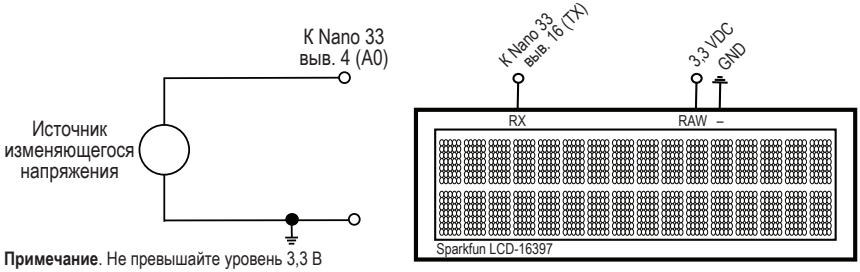
Плата Arduino Nano 33 BLE Sense оснащена восьмиканальным аналого-цифровым преобразователем с разрешением 12 бит и скоростью 200 тыс. выборок в секунду (ksps). ADC преобразует аналоговый сигнал из внешнего мира в двоичное представление, подходящее для использования микроконтроллером. Разрешение 12 бит означает, что аналоговое напряжение от 0 до 3,3 В будет закодировано в одно из 4096 двоичных представлений между $(000)_{16}$ и $(FFF)_{16}$. Это обеспечивает разрешение по напряжению примерно 0,81 мВ.

Аналоговый канал считывается с помощью функции `analogRead()`. Эта функция использует по умолчанию 10-битное разрешение ADC. Разрешение преобразования ADC можно настроить с помощью функции `analogReadResolution(num_bits)`. Желаемый уровень разрешения здесь указывается с помощью переменной `num_bits`.

Плата Nano 33 имеет восемь каналов аналого-цифрового преобразования:

- канал A0, контакт 4;
- канал A1, контакт 5;
- канал A2, контакт 6;
- канал A3, контакт 7;
- канал A4, контакт 8 (также контакт I2C SDA);
- канал A5, контакт 9 (также контакт I2C SCL);
- канал A6, контакт 10;
- канал A7, контакт 11.

Пример **В этом примере мы измеряем напряжение от источника переменного тока. Обратите внимание на строку в коде для преобразования показаний АЦП от 0 до 1023 (т. е. с разрешением 10 бит) в значение аналогового напряжения: `A0_voltage = (analog_reading_A0 * 3.3)/1024`. Конфигурация схемы показана на рис. 2.8.**



Примечание. Не превышайте уровень 3,3 В в источнике изменяющегося напряжения.

Рис. 2.8. Тест ADC с ЖК-дисплеем

```
//*********************************************************************
//LCD_USART_ADC_example
//
//The robot is equipped with a Sparkfun LCD-16397
//- Nano 33 BLE Sense, USART TX pin 16 is connected to LCD USART RX pin
//- Provide 3.3 VDC power to the LCD
//- Nano 33 BLE Sense Analog A0 at pin 4
//*********************************************************************
int analog_reading_A0;
float A0_voltage;
void setup()
{
  Serial1.begin(9600);               //Baud rate: 9600 Baud
  delay(500);                        //Delay for display
}
void loop()
{
  analog_reading_A0 = analogRead(A0); //read A0
//convert to voltage
  A0_voltage = (analog_reading_A0 * 3.3)/1024;
//Clear LCD
//Cursor to line one, character one
  Serial1.write(254);                 //Command prefix
  Serial1.write(128);                 //Command
//clear display
  Serial1.write(" ");
  Serial1.write(" ");
//Cursor to line one, character one
  Serial1.write(254);                 //Command prefix
```

```
  Serial1.write(128);                //Command
  Serial1.write("Voltage: ");
  delay(50);
  Serial1.write(254);                //Command to LCD
  delay(5);
  Serial1.write(192);                //Cursor line 2, position 1
  delay(5);
  Serial1.print(analog_reading_A0);
  Serial1.write(" ");
  Serial1.print(A0_voltage);
  Serial1.write("V");
  delay(500);
}
//*********************************************************************
```

2.5.3. Bluetooth с низким энергопотреблением (BLE)

Классическая версия Bluetooth была разработана для беспроводной замены обычного стандарта последовательного соединения RS-232. Arduino Nano 33 BLE sense оснащена функциями более новой версии с низким энергопотреблением Bluetooth Low Energy (BLE) [4]. Важно отметить, что функции классического Bluetooth Classic и BLE несовместимы друг с другом. Здесь мы концентрируемся на функциях именно BLE.

Bluetooth версии BLE обеспечивает радиочастотное соединение с низкой мощностью передачи (10 мВт) и малым радиусом действия (максимум 100 м) для замены проводов. Он использует перегруженный диапазон частот промышленной, научной и медицинской сферы (ISM) от 2,40 до примерно 2,50 ГГц. Полоса BLE разделена на 40 различных каналов по 2 МГц, как показано на рис. 2.9. BLE использует для связи интересную технику скачкообразной перестройки частоты. Данные для передачи разбиваются на пакеты со скоростью передачи данных до 2 Мбит/с. Устройство передает пакет данных на первой несущей частоте. Затем он переходит на другую несущую частоту для следующего пакета и так далее, пока все сообщение не будет передано, как показано на рис. 2.9, *б*. Формально метод модуляции BLE называется «метод прямой последовательности для расширения спектра» (Direct Sequence Spread Spectrum, DSSS) (www.bluetooth.com)[1].

[1] Метод повышения помехозащищенности и расширения спектра путем скачкообразной перестройки несущей, описанный автором в преды-



а) Полоса частот для промышленных, научных и медицинских целей (ISM)

б) Метод расширения спектра скачкообразной перестройкой частоты (FHSS)

Рис. 2.9. Концепции связи Bluetooth BLE

BLE использует профиль общего атрибута (Generic Attribute, GATT) для установления двух разных основных ролей при соединении:

- роль периферийного устройства или сервера обеспечивает функции доски объявлений, на которой публикуются данные для чтения;
- центральное или клиентское устройство может читать опубликованные данные и взаимодействовать с ними.

На рис. 2.10 мы используем Arduino Nano 33 BLE Sense в роли периферийного сервера для сбора важной информации о теплице, такой как внешняя температура, внутренняя температура, влажность и содержание влаги в почве. Данные о теплицах собираются и организуются в службу BLE. Данные, относящиеся к услуге, предоставляются как заданные характеристики BLE. Чтобы облегчить доступ к информации с внешнего центрального клиентского устройства, каждой службе и характеристике BLE присвоен универсальный уникальный идентификатор (UUID) [5]. Если бы мы

дущих предложениях, называется «методом псевдослучайной перестройки рабочей частоты» (Frequency-Hopping Spread Spectrum, FHSS, см. рис. 2.9, *б*). Метод прямого расширения спектра DSSS был добавлен к основному методу в последних версиях Bluetooth с целью повышения скорости передачи при сохранении низкого энергопотребления. – *Прим. перев.*

расширили возможности проекта дополнительными услугами, мы могли бы сгруппировать их в профиль.



Рис. 2.10. Теплица, оснащенная Bluetooth BLE

Существует несколько предварительно назначенных 16-битных идентификаторов UUID. UUID предоставляют различные производители и технологические компании, использующие технологии на основе Bluetooth. Кроме того, UUID предварительно назначены общим функциям Bluetooth и типам данных (например, температуре, давлению и т. д.) [4]:

- члены группы Bluetooth: 0xFxxx;
- характеристика GATT и тип объекта: 0x2xxx;

- декларации GATT: 0x28xx и 0x29xx;
- служба GATT: 0x18xx;
- единица измерения GATT: 0x27xx;
- идентификатор протокола: 0x00xx;
- классы и профили сервисов: 0x10xx и 0x11xx.

Для сервисов и характеристик BLE без заранее назначенного 16-битного UUID используется уникальный 128-битный код UUID. Уникальный UUID Bluetooth можно получить с помощью различных онлайн-генераторов UUID.

В примере с теплицей сотовый телефон настроен как центральный сервер или клиент BLE. Через беспроводное радиосоединение BLE мобильный телефон может считывать данные теплицы и взаимодействовать с ее управляющими функциями.

2.5.3.1. Библиотека ArduinoBLE

Библиотека ArduinoBLE предоставляет широкий спектр конфигураций BLE. Ее можно загрузить в Arduino IDE с помощью Library Manager. Библиотека включает много различных классов, в том числе [2]:

- класс BLE, используемый для включения BLE-модуля;
- класс BLE-устройств для получения информации о подключенных устройствах;
- класс сервисов BLE для подключения услуг и взаимодействия с ними;
- класс характеристик BLE для доступа к характеристикам и взаимодействия с ними;
- класс BLE-дескрипторов для описания характеристик.

Знакомство с библиотекой продолжим серией примеров. Первые два примера взяты из библиотеки Arduino BLE. В третьем примере мы настраиваем Arduino Nano 33 BLE Sense в качестве сервера для сбора и публикации данных из теплицы. Сотовый телефон настроен как клиент для опроса и взаимодействия с данными теплицы. Он оснащен BLE-совместимым приложением для взаимодействия с Nano 33. Этот пример приведен в разделе «Приложения» в конце главы.

Пример **В первом примере из библиотеки Arduino BLE под названием «LED» сотовый телефон служит центральным клиентом для управления светодиодом на плате Nano 33 BLE Sense, настроенной как сервер. Чтобы лучше познакомиться со скетчем, мы изучим этапы кода, связанные с настройкой Bluetooth. На рис. 2.11 подробно показаны эти шаги на UML-диаграмме действий[1].**

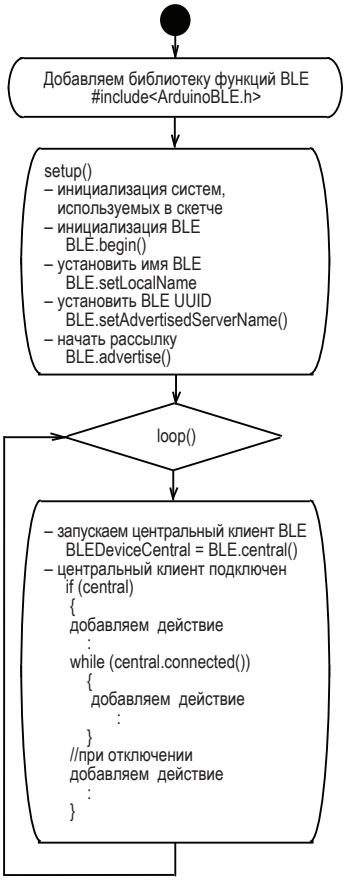


Рис. 2.11. Настройка Bluetooth BLE

[1] UML (Unified Modeling Language, унифицированный язык моделирования) – язык графического описания для объектного моделирования в области разработки программного обеспечения. Диаграмма действий (activity diagram) – последовательность операций, представленная на языке UML. Подробнее см. статьи «Диаграмма деятельности» и «UML» в Википедии (https://ru.wikipedia.org). – *Прим. перев.*

```
//************************************************************
//LED: This example creates a BLE peripheral with service
//that contains a characteristic to control an LED.
//
//A generic BLE central phone app, like LightBlue or
//nRF Connect is used to interact with the Nano 33
//hosted BLE services and characteristics created in this
//sketch.
//
//This example code is in the public domain.
//************************************************************
#include <ArduinoBLE.h>
//Declare BLE LED Service
//Link to 128 bit UUID
BLEService ledService("19B10000-E8F2-537E-4F6C-D104768A1214");
//BLE LED Switch Characteristic - custom 128-bit UUID, read and
//writable by central client device (cell phone)
BLEByteCharacteristic switchCharacteristic
   ("19B10001-E8F2-537E-4F6C-D104768A1214", BLERead | BLEWrite);
const int ledPin = LED_BUILTIN;        //Use builtin LED
void setup()
{
  Serial.begin(9600);                  //status to serial monitor
  while (!Serial);
  pinMode(ledPin, OUTPUT);             //set LED pin to output mode
  if(!BLE.begin())                     //BLE initialization
  {
    Serial.println("starting BLE failed!");
    while (1);
  }
//set advertised local name and service UUID:
  BLE.setLocalName("LED");
  BLE.setAdvertisedService(ledService);
//add the characteristic to the service
  ledService.addCharacteristic(switchCharacteristic);
//add service
  BLE.addService(ledService);
//set the initial value for the characeristic:
  switchCharacteristic.writeValue(0);
//start advertising
  BLE.advertise();
  Serial.println("BLE LED Peripheral");
}
void loop()
{
//listen for BLE clients (central) to connect:
  BLEDevice central = BLE.central();
//if a client (central) is connected to peripheral:
  if(central)
  {
    Serial.print("Connected to client: ");
```

```
//print the client's MAC address:
    Serial.println(central.address());
//while the client (central) is still connected to
//the Nano 33 based server (peripheral):
    while (central.connected())
    {
//if the remote client device wrote to the
//Nano 33 server characteristic, use the
//value to control the LED:
      if(switchCharacteristic.written())
      {
       if(switchCharacteristic.value())
       {
         Serial.println("LED on");      //any value other than 0
         digitalWrite(ledPin, HIGH);    //will turn the LED on
       }
       else
       {
         Serial.println(F("LED off"));  //a 0 value
         digitalWrite(ledPin, LOW);     //will turn the LED off
       }
      }
    }//end while
//when the central disconnects, print it out:
    Serial.print(F("Disconnected from central: "));
    Serial.println(central.address());
  }//end if(central)
}
//************************************************************
```

Скетч можно скомпилировать и загрузить в Nano 33 BLE. После загрузки можно протестировать:

- откройте монитор последовательного порта в Arduino IDE, чтобы отслеживать состояние скетча;
- используя мобильный телефон в качестве клиента, откройте приложение «nRF Connect», чтобы установить соединение BLE с сервером на базе Nano 33[1];
- найдите «LED» в списке сканированных устройств nRF;
- нажмите «Connect» (*Подключиться*), чтобы подключить клиента (сотовый телефон) к серверу (Nano 33 BLE);
- через пункт «Client» и стрелку вверх значения для управления светодиодом можно отправлять с клиента на сервер;

[1] Приложения Bluetooth BLE, такие как «nRF Connect» или «LightBlue», доступны в магазине приложений вашего мобильного телефона. – *Прим. авт.*

- выберите «Write Value» (*Записать значение*) и «Unsigned» (*Беззнаковое*);
- отправка ненулевого значения включает светодиод, а отправка нуля выключает светодиод.

Пример **Пример «Battery monitor» («Монитор батареи») адаптирован из библиотеки Arduino BLE. Здесь Nano 33 BLE Sense настроен как сервер. Nano 33 BLE Sense контролирует аналоговый сигнал на входе A0 и отправляет это значение как характеристику на «доску объявлений» на сервере. Для опроса публикуемых данных используется клиент на базе мобильного телефона, оснащенный приложением, совместимым с BLE.**

Для имитации батареи к выводу A0 подключается потенциометр сопротивлением 100 кОм. Потенциометр подключается между напряжением 3,3 В постоянного тока и общим проводом (крайние выводы). Движок потенциометра (центральный вывод) подключается к контакту A0.

Соединение клиент/сервер тестируется с использованием методов, аналогичных приведенным в предыдущем примере.

```
//************************************************************
//Battery Monitor - This example creates a BLE server
//(peripheral) with the standard battery service and level
//characteristic. The A0 pin is used to monitor the battery
//level.
//
//A generic BLE central phone app, like LightBlue or
//nRF Connect is used to interact with the Nano 33
//hosted BLE services and characteristics created in this
//sketch.
//
//This example code is in the public domain.
//************************************************************
#include <ArduinoBLE.h>
BLEService batteryService("180F");   //BLE Battery Service
//BLE Battery Level Characteristic
// - standard 16-bit characteristic UUID
// - remote clients get notifications if characteristic changes
BLEUnsignedCharCharacteristic batteryLevelChar("2A19",
                                    BLERead | BLENotify);
int oldBatteryLevel = 0;      //last battery level reading from A0
long previousMillis = 0;      //last time battery level checked (ms)
void setup()
{
  Serial.begin(9600);         //initialize serial communication
  while (!Serial);
                              //initialize built-in LED pin
```

```
    pinMode(LED_BUILTIN,OUTPUT);   //indicates when central is connected
    if(!BLE.begin())               //initialize Bluetooth BLE device
    {
      Serial.println("starting BLE failed!");
      while (1);
     }
//Set a local name for the BLE device. This name appears
//in advertising packets. Name used by remote devices to
//identify this BLE device.
  BLE.setLocalName("BatteryMonitor");
  BLE.setAdvertisedService(batteryService);  //add the service UUID
//add the battery level characteristic
  batteryService.addCharacteristic(batteryLevelChar);
  BLE.addService(batteryService);               //add battery service
  batteryLevelChar.writeValue(oldBatteryLevel); //set initial value
//Start advertising BLE. Continuously transmits BLE advertising
//packets. Advertising will be visible to remote BLE central devices.
  BLE.advertise();
  Serial.println("Bluetooth device active, waiting for connections...");
}
void loop()
{
  BLEDevice central = BLE.central();    //wait for a BLE central
 //if a central client
//is connected to peripheral
  if(central)
  {
    Serial.print("Connected to central: ");
    Serial.println(central.address()); //print the central's BT address
    digitalWrite(LED_BUILTIN, HIGH);   //LED on when client connected
//while client connected
//check battery level every 200ms
  while(central.connected())
  {
    long currentMillis = millis();
//if 200ms have passed,
//check the battery level
    if(currentMillis - previousMillis >= 200)
    {
      previousMillis = currentMillis;
      updateBatteryLevel();
    }
  }
//when the central client disconnects, turn off the LED
    digitalWrite(LED_BUILTIN, LOW);
    Serial.print("Disconnected from central: ");
    Serial.println(central.address());
  }
}
//************************************************************
//void updateBatteryLevel() - Read the current voltage level
```

```
//on the A0 analog input pin. This is used here to simulate
//the battery level.
//*************************************************************
void updateBatteryLevel()
{
  int battery = analogRead(A0);
  int batteryLevel = map(battery, 0, 1023, 0, 100);
  if(batteryLevel != oldBatteryLevel)
  {                           //if the battery level has changed
   Serial.print("Battery Level %: ");         // print it
   Serial.println(batteryLevel);
   batteryLevelChar.writeValue(batteryLevel); //update battery level
//characteristic
   oldBatteryLevel = batteryLevel;       //save level for comparison
  }
}
//*************************************************************
```

2.6. Периферийные устройства Nano 33 BLE Sense

Как обсуждалось ранее в этой главе, плата Arduino Nano 33 BLE Sense оснащена представительным набором датчиков. В этом разделе мы подробно рассмотрим следующие датчики:

- девятиосевой инерциальный измерительный блок (inertial measurement unit, IMU) LSM9DS1 [9];
- барометр и датчик температуры LPS22HB [8];
- датчик относительной влажности HTS221 [7];
- цифровой датчик приближения, окружающего освещения, RGB-цвета и жестов APDS-9960 [1];
- цифровой микрофон MP34DT05 [10].

2.6.1. Девятиосевой IMU LSM9DS1

Инерциальный измерительный блок (IMU) реализует важную функцию для локальной ориентации в координатах X, Y, Z. IMU состоит из акселерометра, гироскопа и магнитометра для измерения ускорения, вращения и напряженности магнитного поля Земли в системе координат X, Y, Z (см. рис. 2.12).

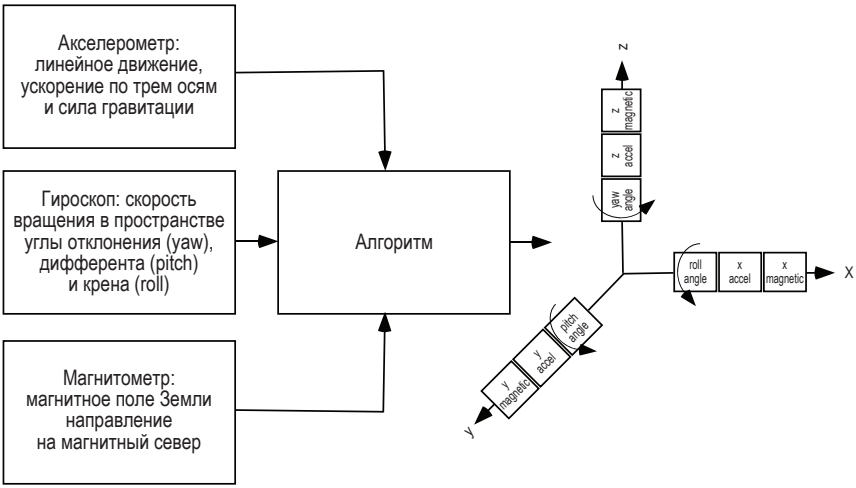


Рис. 2.12. Инерционный измерительный блок (IMU)

Nano 33 BLE sense оснащен инерциальным модулем LSM9DS1 iNemo. Модуль состоит из 3D-акселерометра, 3D-гироскопа и 3D-магнитометра. Модуль имеет следующие характеристики:

- диапазон измерения линейного ускорения: ±2, 4, 8, 16 g;
- диапазон измерения магнитной индукции ±4, 8, 12, 16 Гаусса (Гс, G);
- диапазон измерения углового ускорения ±245, ±500, ±2000 градусов в секунду (°/c, dps).

Пример **Библиотека «Arduino_LSM9DS1», созданная Риккардо Риццо (Riccardo Rizzo), предоставляет примеры для проверки встроенного акселерометра, гироскопа и магнитометра. Здесь представлен пример «Simple_Accelerometer».**

```
//*****************************************************
//Arduino LSM9DS1 - Simple Accelerometer
//This example reads the acceleration values from
//the LSM9DS1 sensor and continuously prints them
//to the Serial Monitor or Serial Plotter.
//
//The circuit: Arduino Nano 33 BLE Sense
//
//created 10 Jul 2019 by Riccardo Rizzo
//
//This example code is in the public domain.
```

```
//*******************************************************
#include <Arduino_LSM9DS1.h>
void setup()
{
  Serial.begin(9600);
  while (!Serial);
  Serial.println("Started");
  if(!IMU.begin())
  {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
  Serial.print("Accelerometer sample rate = ");
  Serial.print(IMU.accelerationSampleRate());
  Serial.println(" Hz");
  Serial.println();
  Serial.println("Acceleration in G's");
  Serial.println("X\tY\tZ");
}
void loop()
{
  float x, y, z;
  if(IMU.accelerationAvailable())
  {
    IMU.readAcceleration(x, y, z);
    Serial.print(x);
    Serial.print('\t');
    Serial.print(y);
    Serial.print('\t');
    Serial.println(z);
  }
}
//*******************************************************
```

2.6.2. Барометр и датчик температуры LPS22HB

Сенсорный модуль LPS22HB обеспечивает измерение атмосферного давления и температуры[1]. Давление (в паскалях) – это величина

[1] Следует представлять, что величина температуры в барометрическом датчике имеет вспомогательный характер (для точного пересчета показаний чувствительного элемента – тензомоста – в значения давления) и, как правило, значительно, на единицы градусов и более отличается от температуры окружающей среды. По этой причине величина температуры в данном случае носит справочный характер и использоваться в качестве параметра окружающей среды может только условно, в процессе тестирования алгоритмов. Отметим, что на показаниях атмо-

силы (в ньютонах) на единицу площади (в кв. метрах). Барометрическое давление – это давление на Земле, вызванное весом воздуха над ним по сравнению с вакуумом. Интуитивно понятно, что барометрическое давление уменьшается с увеличением высоты, на которой проводятся измерения. Кроме того, атмосферное давление меняется в зависимости от погодных условий. Например, барометрическое давление снижается при дождливой погоде [6].

LPS22HB обеспечивает показания давления от 260 до 1260 гПа (гектопаскалей). Стандартная атмосфера на уровне моря равна 1013 гПа. Это эквивалентно 101,325 кПа, 1013,25 мбар и 14,696 фунта на квадратный дюйм[1].

Пример **Пример «ReadPressure» («Читаем давление») из библиотеки «Arduino_LPS22HB» обеспечивает считывание давления и температуры. Интересно, что когда я запускаю эту программу в своей домашней лаборатории, то получаю значение 77,62 кПа[2]. Напомним, стандартная атмосфера на уровне моря равна 101,325 кПа. Когда показания 77,62 кПа передаются в «Калькулятор давления воздуха на высоте», моя высота определяется как 7194,33 фута (www.mide.com). Футбольная и баскетбольная команды Университета Вайоминга регулярно напоминают соперникам, что они соревнуются на высоте 7220 футов над уровнем моря[3]!**

```
//*******************************************************
//LPS22HB - Read Pressure
//This example reads data from the on-board LPS22HB
//sensor of the Nano 33 BLE Sense and prints the
//temperature and pressure sensor value to the Serial
//Monitor once a second.
//
//The circuit: Arduino Nano 33 BLE Sense
//
//This example code is in the public domain.
//*******************************************************
#include <Arduino_LPS22HB.h>
void setup()
{
  Serial.begin(9600);
```

сферного давления эти факторы сказываться не будут, но для контроля температуры и влажности (см. далее) окружающего воздуха следует применять отдельный датчик, вынесенный за пределы платы и всего корпуса устройства. – *Прим. перев.*

[1] 760 мм ртутного столба. – *Прим. перев.*

[2] 582,2 мм ртутного столба. – *Прим. перев.*

[3] 2200 метров. – *Прим. перев.*

```
  while (!Serial);
  if(!BARO.begin())
  {
    Serial.println("Failed to initialize pressure sensor!");
    while (1);
  }
}
void loop()
{
//read the sensor value
  float pressure = BARO.readPressure();
//print the sensor value
  Serial.print("Pressure = ");
  Serial.print(pressure);
  Serial.println(" kPa");
  float temperature = BARO.readTemperature();
//print the sensor value
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" C");
// print an empty line
  Serial.println();
// wait 1 second to print again
  delay(1000);
}
//***************************************************
```

2.6.3. Датчик относительной влажности и температуры HTS221

Датчик HTS221 обеспечивает измерение относительной влажности (rH). Относительная влажность является мерой содержания водяного пара в воздухе. Абсолютная влажность – это масса водяного пара в объеме воздуха, измеряемая в граммах на куб. метр. Относительная влажность характеризует количество водяного пара относительно максимального количества воды, которое может содержаться в объеме воздуха при данной температуре (точка насыщения). Относительная влажность указывается в процентах от 0 до 100 [6].

Я жил во многих местах с широким диапазоном влажности. Например, я учился в неполной средней школе на Гуаме. Гуам – тропический остров в Тихом океане, где относительная влажность регулярно превышает 80 %. Сейчас я живу в Вайоминге, где относительная влажность летом составляет в среднем 25 %. Рисунок 2.13 иллюстрирует диапазон относительной влажности.

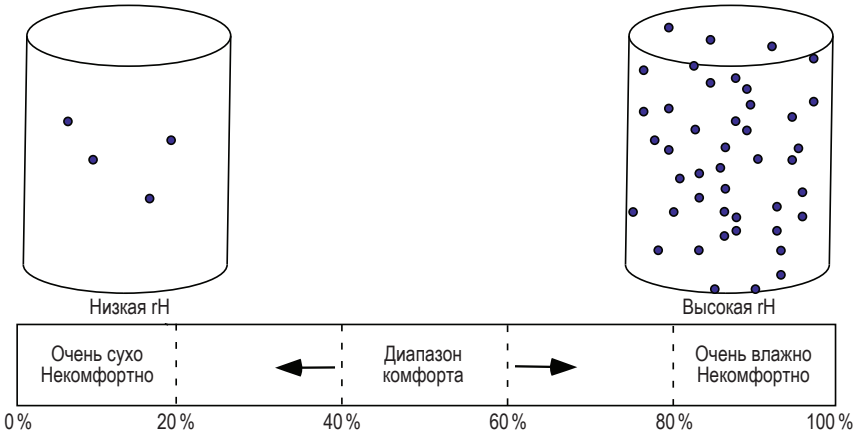


Рис. 2.13. Относительная влажность (rH) [6]

Датчик HTS221 обеспечивает показания относительной влажности от 0 до 100 %. Он обеспечивает точность ±3,5 % в диапазоне от 20 до 80 % относительной влажности.

Пример **Пример «ReadSensors» («Чтение датчиков») из библиотеки «Arduino_HTS221» обеспечивает считывание относительной влажности и температуры**[1]**.**

```
//*************************************************
//HTS221 - Read Sensors
//This example reads data from the on-board HTS221
//sensor of the Nano 33 BLE Sense and prints the
//temperature and humidity sensor values to the Serial
//Monitor once a second.
//
//The circuit: Arduino Nano 33 BLE Sense
//
//This example code is in the public domain.
//*************************************************
#include <Arduino_HTS221.h>
```

[1] К показаниям датчика влажности-температуры, установленного на плате Nano 33, следует также относиться с осторожностью (см. сноску на стр. 67). Датчик окружен компонентами, нагревающимися при работе, и, кроме того, в готовом изделии он вместе с платой помещен в корпус, изолирующий его от окружающего воздуха. Повторим, что для контроля температуры и влажности окружающего воздуха следует применять отдельный датчик, вынесенный за пределы платы и всего корпуса устройства. – *Прим. перев.*

```
void setup()
{
  Serial.begin(9600);
  while (!Serial);
  if(!HTS.begin())
  {
    Serial.println("Failed to initialize humidity temperature sensor!");
    while (1);
  }
}
void loop()
{
//read all the sensor values
  float temperature = HTS.readTemperature();
  float humidity = HTS.readHumidity();
//print each of the sensor values
  Serial.print("Temperature = ");
  Serial.print(temperature);
  Serial.println(" Â°C");
  Serial.print("Humidity = ");
  Serial.print(humidity);
  Serial.println(" %");
//print an empty line
  Serial.println();
//wait 1 second to print again
  delay(1000);
}
//*************************************************
```

2.6.4. Цифровой датчик расстояния, окружающего освещения, RGB-цвета и распознавания жестов APDS-9960

Arduino Nano 33 BLE Sense оснащен датчиком APDS-9960. Он оснащен функциями распознавания жестов, цветовых компонентов RGB и измерения расстояния до объектов поблизости. Мы обсудим каждый из них по очереди в соответствии с прилагаемыми программными примерами.

2.6.4.1. Распознавание жестов

Функция распознавания жестов использует четыре направленных фотодиода, показанных на рис. 2.14, *а*. По рисунку видно, как определяется направление жеста. Светодиод, расположенный под фотодиодами, обеспечивает освещение источника. Когда проис-

ходит, например, движение слева направо перед фотодиодами, распознается жест вправо.

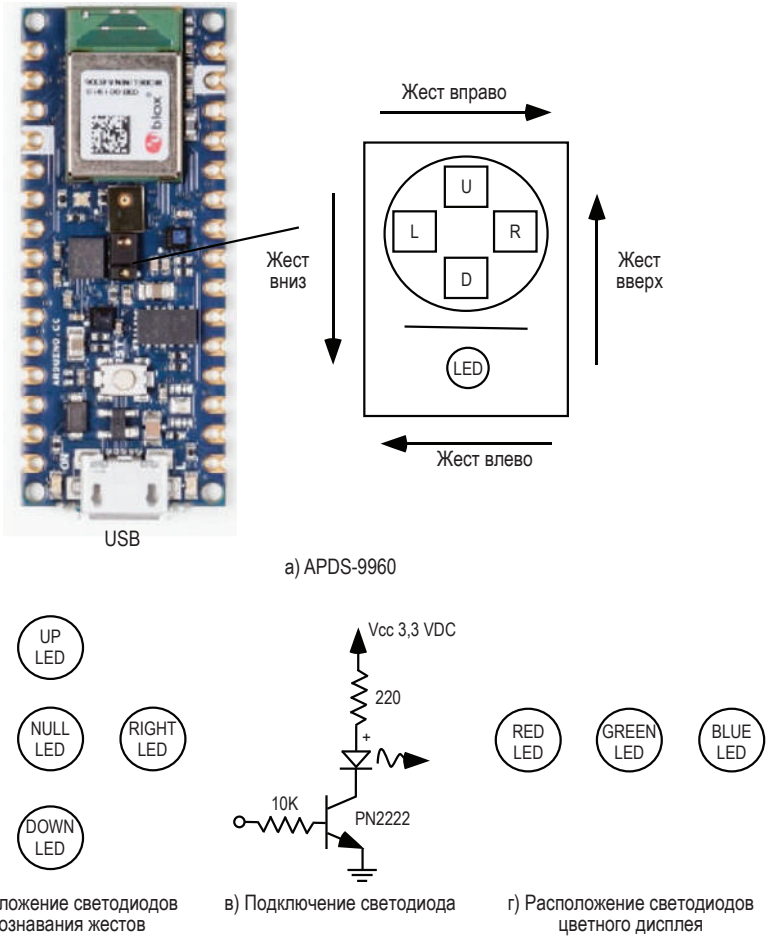


Рис. 2.14. Датчик APDS-9960 [1]

Arduino предоставляет библиотеку ADPS-9960, которая включает в себя примеры скетчей для тестирования различных функций.

Пример **В этом примере мы немного изменили скетч датчика жестов, представленный в библиотеке APDS-9960. При обнаружении жеста загорается соответствующий светодиод. Расположение светодиодов дисплея показано на рис. 2.14, *б*.**

```
//*********************************************************************
//APDS-9960 - Gesture Sensor - This example reads gesture data from
//the onboard APDS-9960 sensor of the Nano 33 BLE Sense and prints
//any detected gestures to the Serial Monitor and illuminates a
//corresponding LED to indicate the gesture direction.
//
//Gesture directions are defined as:
//- UP: from USB connector towards antenna
//- DOWN: from antenna towards USB connector
//- LEFT: from analog pins side towards digital pins side
//- RIGHT: from digital pins side towards analog pins side
//
//Five LEDs are used to indicate gesture direction:
//- UP LED: Digital pin D2 - physical pin 20
//- DOWN LED: Digital pin D3 - physical pin 21
//- LEFT LED: Digital pin D4 - physical pin 22
//- Right LED: Digital pin D5 - physical pin 23
//- NULL LED: Digital pin D6 - physical pin 24
//
//The example code "Gesture Sensor" from the Arduino APDS-9960
//Library was adapted to include the five LED indicators. The code
//is in the public domain.
//*********************************************************************
#include <Arduino_APDS9960.h>
#define UP_LED 2           //physical pin
#define DOWN_LED 3         //physical pin
#define LEFT_LED 4         //physical pin
#define RIGHT_LED 5        //physical pin
#define NULL_LED 6         //physical pin
void setup()
{
  pinMode(UP_LED, OUTPUT);
  pinMode(DOWN_LED, OUTPUT);
  pinMode(LEFT_LED, OUTPUT);
  pinMode(RIGHT_LED,OUTPUT);
  pinMode(NULL_LED, OUTPUT);
  Serial.begin(9600);
  while (!Serial);
  if(!APDS.begin())
  {
    Serial.println("Error initializing APDS-9960 sensor!");
  }
//Note: for setGestureSensitivity(..) a value between 1 and 100
//is required. Higher values make the gesture recognition more
//sensitive but less accurate (i.e. a wrong gesture may be
//detected). Lower values makes the gesture recognition more accurate
//but less sensitive (i.e. some gestures may be missed). A default
//of 80 is used. To change this value, uncomment the next line and insert a desired threshold
value.
//APDS.setGestureSensitivity(80);
  Serial.println("Detecting gestures ...");
}
```

```
void loop()
{
//LED setting to default values
  digitalWrite(UP_LED, LOW);
  digitalWrite(DOWN_LED, LOW);
  digitalWrite(LEFT_LED, LOW);
  digitalWrite(RIGHT_LED,LOW);
  digitalWrite(NULL_LED, HIGH);
  if(APDS.gestureAvailable())      //a gesture was detected,
  {                                //print to Serial Monitor
                                   //illuminate LEDs
    int gesture = APDS.readGesture();
    switch(gesture)
    {
      case GESTURE_UP:
      Serial.println("Detected UP gesture");
      digitalWrite(UP_LED, HIGH);
      digitalWrite(NULL_LED, LOW);
      break;
      case GESTURE_DOWN:
      digitalWrite(DOWN_LED, HIGH);
      digitalWrite(NULL_LED, LOW);
      Serial.println("Detected DOWN gesture");
      break;
      case GESTURE_LEFT:
      Serial.println("Detected LEFT gesture");
      digitalWrite(LEFT_LED, HIGH);
      digitalWrite(NULL_LED, LOW);
      break;
      case GESTURE_RIGHT:
      Serial.println("Detected RIGHT gesture");
      digitalWrite(RIGHT_LED, HIGH);
      digitalWrite(NULL_LED, LOW);
      break;
      default:                     //ignore
      break;
    }
  delay(1000);                     //1 second delay
  }
}
//*********************************************************************
```

2.6.4.2. Датчик цвета

APDS-9960 также оснащен датчиком цвета, который сообщает о хроматическом составе (RGB-компоненты: красный, зеленый, синий) обнаруженного света. Различное цветовое содержание воспринимается с помощью трех разных фотодиодов, детектирующих красный (с центральной длиной волны примерно 610 нм), зеленый (примерно 540 нм) и синий (примерно 450 нм).

Следующий скетч «Color Sensor» («Датчик цвета»), представленный в библиотеке Arduino ADPS-9960, адаптирован для зажигания трех отдельных цветных светодиодов с интенсивностью обнаруженных хроматических составляющих. Схема размещения светодиодов R, G и B показана на рис. 2.14, *в*. Скетч проверяется при размещении различных цветных фильтров перед источником света (например, набора полноцветных линзовых фильтров Neewer 58 мм[1]).

```
//*****************************************************************
//APDS-9960 - Color Sensor1 - This example reads color data from
//the onboard APDS-9960 sensor of the Nano 33 BLE Sense and prints
//the color RGB (red, green, blue) values to the Serial Monitor once
//a second. Also, external R, G, and B LEDs are illuminated with the
//corresponding intensity of the R,G, B component reading.
//
//Three LEDs are used to indicate R,G,B component presence.
//- RED LED: Digital PWM pin D3
//- BLUE LED: Digital PWM pin D5
//- GREEN LED: Digital PWM pin D6
//
//The example code "Color Sensor" from the Arduino APDS-9960 Library
//was adapted to include the three LED indicators. The code is in
//the public domain.
//*****************************************************************
#include <Arduino_APDS9960.h>
#define RED_LED 3     //physical pin 21
#define BLUE_LED 5    //physical pin 23
#define GREEN_LED 6   //physical pin 24
void setup()
{
  pinMode(RED_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  Serial.begin(9600);
  while (!Serial);
  if(!APDS.begin())
  {
    Serial.println("Error initializing APDS-9960 sensor.");
  }
}
void loop()
{
```

[1] На практике освещение цветового датчика через столь экзотичные и дорогие фотографические фильтры можно использовать разве что для его точной калибровки. Для проверки скетча достаточно света, отраженного от образцов цветной бумаги или пластика; оптимальные результаты получаются на расстоянии 1–2 см. – *Прим. перев.*

```
//LED setting to default values
  digitalWrite(RED_LED, LOW);
  digitalWrite(BLUE_LED, LOW);
  digitalWrite(GREEN_LED, LOW);
//check if a color reading is available
  while (! APDS.colorAvailable())
  {
    delay(5);
  }
  int r, g, b;
//read the color
  APDS.readColor(r, g, b);
//print the values
  Serial.print("r = ");
  Serial.println(r);
  Serial.print("g = ");
  Serial.println(g);
  Serial.print("b = ");
  Serial.println(b);
  Serial.println();
//illuminate the LEDs to intensity of component value via PWM
  analogWrite(RED_LED, r);
  analogWrite(GREEN_LED, g);
  analogWrite(BLUE_LED, b);
  delay(100);        //wait a bit before reading again
}
//*********************************************************************
```

2.6.4.3. Датчик расстояния

APDS-9960 также оснащен датчиком расстояния до объекта поблизости. В качестве источника освещения датчик использует встроенный инфракрасный диод. Свет от источника отражается от объекта, отраженный свет собирается теми же четырьмя фотодиодами, которые используются для распознавания жестов. ADPS-9960 сообщает числовое значение, соответствующее дальности объекта. Числовое значение может быть соотнесено с конкретным диапазоном для данного применения.

Следующий пример скетча «Proximity Sensor» («Датчик расстояния»), представленный в библиотеке Arduino ADPS-9960, адаптирован для сообщения о расстоянии до объекта, а также для засветки светодиода с интенсивностью, соответствующей расстоянию (т. е. чем ближе объект, тем ярче светодиод).

```
//*********************************************************************
//APDS-9960 - Proximity Sensor - This example reads proximity data
//from the onboard APDS-9960 sensor of the Nano 33 BLE Sense and
//prints the proximity value to the Serial Monitor every 100 ms.
```

```
//Sensed values range from 0 (close) to 255 (far). The sensed value
//is used to set the intensity of an LED using PWM. The LED is
//connected to D3.
//
//The example code "Proximity Sensor" from the Arduino APDS-9960
//Library was adapted to include the LED indicator. The code is in
//the public domain.
//*****************************************************************
#include <Arduino_APDS9960.h>
#define range_LED 3 //physical pin 21
void setup()
{
  pinMode(range_LED, OUTPUT);
  Serial.begin(9600);
  while (!Serial);
  if(!APDS.begin())
  {
    Serial.println("Error initializing APDS-9960 sensor!");
  }
}
void loop()
{
//LED setting to default value
  digitalWrite(range_LED, LOW);
//check if a proximity reading is available
  if(APDS.proximityAvailable())
  {
     // read the proximity
     // - 0 => close
     // - 255 => far
     // - -1 => error
     int proximity = APDS.readProximity();
// print value to the Serial Monitor
     Serial.println(proximity);
//illuminate LED to appropriate intensity
//closer proximity, brighter LED
     proximity = abs(proximity - 255);
     analogWrite(range_LED, proximity);
  }
// wait a bit before reading again
  delay(100);
}
//*****************************************************************
```

2.6.5. Цифровой микрофон MP34DT05

Nano 33 BLE Sense оснащен всенаправленным микрофоном, т. е. он может принимать звук с разных направлений. Микрофон использует метод модуляции плотности импульсов (pulse density

modulation, PDM), при котором звук кодируется в последовательный поток цифровых импульсов. Цифровые импульсы могут быть усреднены (пропущены через фильтр низкой частоты) для восстановления аналогового сигнала на аудиовыходе. Отношение сигнал/шум микрофона (signal/noise ratio, SNR) составляет 64 дБ при чувствительности –26 дБ полной шкалы (dBFS)[1].

Пример **Скетч «PDMSerialPlotter» захватывает отсчеты с цифрового микрофона MP34DT05 и отображает их через монитор последовательного порта. Отсчеты также можно отобразить в виде временно́й последовательности, если выбрать Tools (*Инструменты*) -> Serial Plotter (*Последовательный плоттер*) в Arduino IDE.**

```
//*****************************************************
//PDMSerialPlotter: reads audio data from the on-board
//PDM microphones, and prints out the samples to the
//Serial Monitor. The Serial Plotter built into the
//Arduino IDE can be used to plot the audio data
//(Tools -> Serial Plotter)
//
//Circuit: Arduino Nano 33 BLE board
//
//This example code is in the public domain.
//*****************************************************
#include <PDM.h>
//default number of output channels
static const char channels = 1;
//default PCM output frequency
static const int frequency = 16000;
//Buffer to read samples, each sample is 16-bits
short sampleBuffer[512];
//Number of audio samples read
volatile int samplesRead;
void setup()
{
  Serial.begin(9600);
  while (!Serial);
//Configure the data receive callback
  PDM.onReceive(onPDMdata);
//Optionally set the gain.
//Defaults to 20 on the BLE Sense
```

[1] Англ. dBFS означает уровень в децибелах при опорном сигнале (мощности или напряжения), соответствующем полной шкале аналого-цифрового преобразователя. – *Прим. перев.*

```
//PDM.setGain(30);
//Initialize PDM with:
// - one channel (mono mode)
// - a 16 kHz sample rate for the Arduino Nano 33 BLE Sense
  if(!PDM.begin(channels, frequency))
  {
    Serial.println("Failed to start PDM!");
    while (1);
  }
}
void loop()
{
//Wait for samples to be read
  if(samplesRead)
  {
//Print samples to the serial monitor or plotter
    for(int i = 0; i < samplesRead; i++)
    {
      if(channels == 2)
      {
        Serial.print("L:");
        Serial.print(sampleBuffer[i]);
        Serial.print(" R:");
        i++;
      }
      Serial.println(sampleBuffer[i]);
    }
//Clear the read count
    samplesRead = 0;
  }
}
//*******************************************************
// void onPDMdata(): callback function to process the
//data from the PDM microphone.
//Note: This callback is executed as part of an ISR.
//Therefore using 'Serial' to print messages inside
//this function isn't supported.
//*******************************************************
void onPDMdata()
{
//Query the number of available bytes
  int bytesAvailable = PDM.available();
//Read into the sample buffer
  PDM.read(sampleBuffer, bytesAvailable);
//16-bit, 2 bytes per sample
  samplesRead = bytesAvailable / 2;
}
//*******************************************************
```

2.7. Приложение: Bluetooth BLE GreenhouseMonitor

При разработке программного обеспечения для системы не всегда возможно или желательно иметь близкий доступ к системе. Например, при разработке программного обеспечения для управления теплицей сама теплица не всегда доступна. Кроме того, меняющиеся погодные условия не позволяют протестировать алгоритм управления в различных условиях. В таких ситуациях вместо системы можно использовать ее симулятор. Симулятор предоставляет необходимые входные данные и сигналы вместо системы, чтобы упростить разработку программного обеспечения.

В следующем примере мы разрабатываем приложение Bluetooth BLE для сбора данных с теплицы и для просмотра их на клиентском мобильном телефоне. При разработке в качестве замены теплицы используется небольшой аппаратный симулятор, состоящий из нескольких потенциометров и светодиодных индикаторов.

Аппаратный симулятор показан на рис. 2.15. Набор из семи переменных резисторов по 10 кОм обеспечивает моделируемые данные о погоде. Один крайний вывод каждого переменного резистора подключен к питанию (Vcc), другой – к общему проводу (GND). Движок переменного резистора подключается к ряду на макетной плате для подключения перемычкой к выводу микроконтроллера. Для имитации двигателей, насосов и т. п. используется ряд из семи светодиодов. Каждый светодиод оснащен интерфейсной схемой, показанной на рис. 2.15[1]. Вход каждой интерфейсной цепи снабжен перемычкой. Готовый симулятор показан на рис. 2.16.

В примере далее мы используем Arduino Nano 33 BLE Sense в качестве сервера для параметров теплицы и делаем их доступными для клиентов BLE. В качестве клиента выступает сотовый телефон. Через приложение BLE (например, «nRF Connect» или «LightBlue») телефон используется для считывания параметров тепли-

[1] В отличие от классических плат Arduino с контроллером ATmega328, позволяющим довольно существенно нагружать каждый вывод (до 40 мА, при условии непревышения общего тока через вывод питания 200 мА), процессор nRF52840 имеет относительно слабые выходы GPIO (не более нескольких мА, см. п. 3.4 далее), поэтому напрямую светодиоды к ним подключать не рекомендуется и промежуточный транзисторный ключ здесь обязателен. – *Прим. перев.*

цы и управления имитациями вентилятора или водяного насоса. Фрагменты примеров BLE, представленных ранее в этой главе, используются в качестве строительных блоков для этого более сложного алгоритма управления[1].

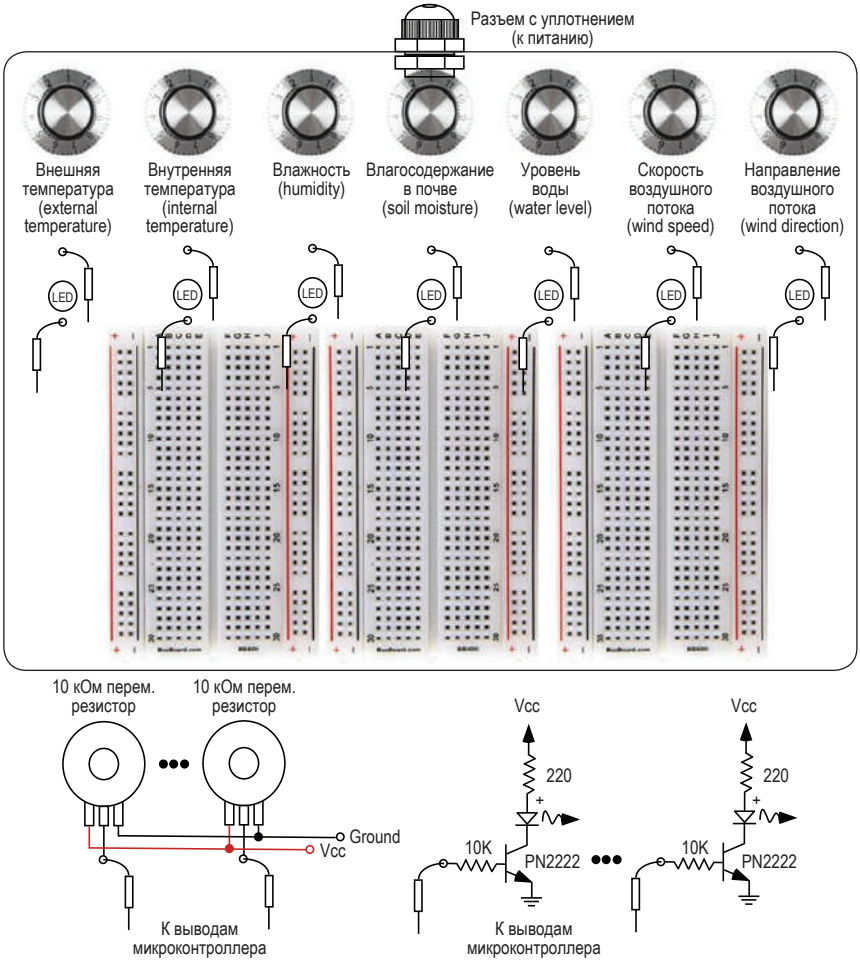


Рис. 2.15. Схема симулятора

[1] Перевод названий параметров среды и некоторых устройств, встречающихся в комментариях к коду, см. рис. 2.15 и 2.16. – *Прим. перев.*

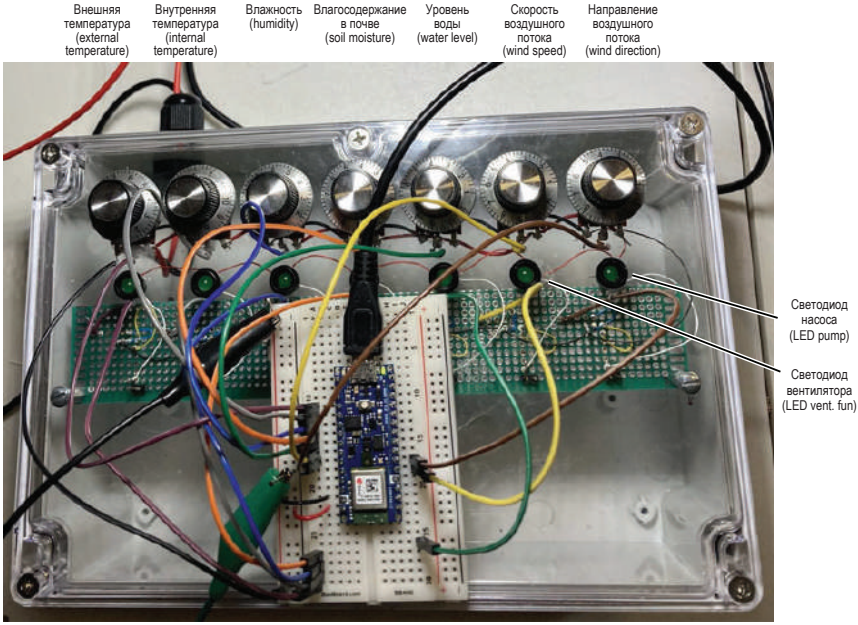


Рис. 2.16. Симулятор теплицы

```
//*********************************************************************
//GreenHouse Monitor: This example provides a BLE peripheral to
//monitor parameters and control features including:
//
//Greenhouse (GH) Service (0x1800)
//Greenhouse Characteristics
//- External temperature (0x2A1C) - A0 (pin 4)
//- Internal temperature (0x2A6E) - A1 (pin 5)
//- Humidity (0x2A6F) - A2 (pin 6)
//- Soil moisture (0x2ACA) - A3 (pin 7)
//- Water level in rain barrel (0x2A78) - A4 (pin 8)
//- Wind speed (0x2A70) - A5 (pin 9)
//- Wind direction (0x2A71) - A6 (pin 10)
//- Battery level (0x2A19) - A7 (pin 11)
//- Vent fan ("19B10001-E8F2-537E-4F6C-D104768A121F") - D2 (pin 20)
//- Water pump ("19B10001-E8F2-537E-4F6C-D104768A121A") - D3 (pin 21)
//Standard BLE characteristic and service codes are used where
//available.
//
//You can use a generic BLE central app (e.g. LightBlue, nRF Connect),
//to interact with the services and characteristics created in this
//sketch.
//
//This example code is in the public domain.
```

```
//*****************************************************************
#include <ArduinoBLE.h>
//Declare BLE service
//BLE Greenhouse service
BLEService greenhouseService("19B10001-E8F2-537E-4F6C-D104768A1214");  //BLE GH Characteristic
//External temperature (0x2A1C) - A0 (pin 4)
BLEUnsignedCharCharacteristic exttempChar("2A1C", BLERead | BLENotify);
int oldexttemp = 0;      //last ext temp reading from analog input A0
//Internal temperature (0x2A6E) - A1 (pin 5)
BLEUnsignedCharCharacteristic inttempChar("2A6E", BLERead | BLENotify);
int oldinttemp = 0;      //last int temp reading from analog input A1
//Humidity (0x2A6F) - A2 (pin 6)
BLEUnsignedCharCharacteristic humidityChar("2A6F", BLERead | BLENotify);
int oldhumidity = 0;     //last humidity reading from analog input A2
//Soil moisture (0x2ACA) - A3 (pin 7)
BLEUnsignedCharCharacteristic soilmoistureChar("2ACA",BLERead |
BLENotify);
int oldsoilmoisture = 0; //last soil moisture reading analog input A3
//Water level in rain barrel (0x2A78) - A4 (pin 8)
BLEUnsignedCharCharacteristic rainlevelChar("2A78", BLERead | BLENotify);
int oldrainlevel = 0;    //last rain level reading from analog input A4
//Wind speed (0x2A70) - A5 (pin 9)
BLEUnsignedCharCharacteristic windspeedChar("2A70", BLERead | BLENotify);
int oldwindspeed = 0;    //last windspeed reading from analog input A5
//Wind direction (0x2A71) - A6 (pin 10)
BLEUnsignedCharCharacteristic winddirectionChar("2A71",BLERead |
BLENotify);
int oldwinddirection = 0; //last wind direction reading analog input A6
//Battery level (0x2A19) - A7 (pin 11)
BLEUnsignedCharCharacteristic batteryLevelChar("2A19", BLERead |
BLENotify);
int oldBatteryLevel = 0; //last battery level reading analog input A7
long previousMillis = 0;  //last time the parameters were checked, ms
//Vent fan - client can remotely operate vent fan
BLEByteCharacteristic switchCharacteristic1
("19B10001-E8F2-537E-4F6C-D104768A121F", BLERead | BLEWrite);
const int fan_ledPin = 2;      //pin to use for the fan LED
//Water pump - client can remotely operate water pump for mister
BLEByteCharacteristic switchCharacteristic2
("19B10001-E8F2-537E-4F6C-D104768A121A", BLERead | BLEWrite);
const int pump_ledPin = 3;     //pin to use for the water pump LED
void setup()
{
  Serial.begin(9600);             //initialize serial communication
  while (!Serial);
  pinMode(fan_ledPin, OUTPUT);    //initialize LED - fan
//indicates when central connected
  if(!BLE.begin())                //begin initialization
  {
    Serial.println("starting BLE failed!");
    while (1);
```

```
  }
//Set local name for BLE device. Name will appear in advertising packets.
//Used by remote devices to identify BLE device.
  BLE.setLocalName("Greenhouse Monitor");
  BLE.setAdvertisedService(greenhouseService);       //add the service UUID
  greenhouseService.addCharacteristic(exttempChar);  //add exttemp char
  greenhouseService.addCharacteristic(inttempChar);  //add inttemp char
  greenhouseService.addCharacteristic(humidityChar); //add humidity char
  greenhouseService.addCharacteristic(soilmoistureChar); //add soil moisture
  greenhouseService.addCharacteristic(rainlevelChar);  //add rain level char
  greenhouseService.addCharacteristic(windspeedChar);  //add windspeed char
  greenhouseService.addCharacteristic(winddirectionChar); //wind dir char
  greenhouseService.addCharacteristic(batteryLevelChar); //batt level char
  greenhouseService.addCharacteristic(switchCharacteristic1); //fan char
  greenhouseService.addCharacteristic(switchCharacteristic2); //pump char
  BLE.addService(greenhouseService);        //add the GH service
  exttempChar.writeValue(oldexttemp);       //set initial value char
  inttempChar.writeValue(oldinttemp);
  humidityChar.writeValue(oldhumidity);
  soilmoistureChar.writeValue(oldsoilmoisture);
  rainlevelChar.writeValue(oldrainlevel);
  windspeedChar.writeValue(oldwindspeed);
  winddirectionChar.writeValue(oldwinddirection);
  batteryLevelChar.writeValue(oldBatteryLevel);
  switchCharacteristic1.writeValue(0);        //vent fan off
  BLEDescriptor vent_fanDescriptor("19B10001-E8F2-537E-4F6C-D104768A121F", "Vent Fan");
  switchCharacteristic1.addDescriptor(vent_fanDescriptor);
  switchCharacteristic2.writeValue(0);        //water pump off
  BLEDescriptor water_pumpDescriptor
          ("19B10001-E8F2-537E-4F6C-D104768A121A", "Water Pump");
  switchCharacteristic2.addDescriptor(water_pumpDescriptor);
  BLE.advertise();                            //start advertising
  Serial.println("Bluetooth device active, waiting for connections...");
}
void loop()
{
  BLEDevice central = BLE.central();       //wait for a BLE central
  if(central)                        //if central client connected...
  {
    Serial.print("Connected to central: ");
    Serial.println(central.address());  //print clients's BT address
//while client is connected
    while(central.connected())
    {
      long currentMillis = millis();
      if(currentMillis - previousMillis >= 200) //check param every 200 ms
       {
         previousMillis = currentMillis;
         update_sensor_values();
       }
//if the remote device wrote to the characteristic,
```

```
//use the value to control the fan
       if(switchCharacteristic1.written())
      {
        if(switchCharacteristic1.value())
      {                  //any value other than 0
        Serial.println("Vent Fan - on");
        digitalWrite(fan_ledPin, HIGH);//will turn the LED (fan) on
      }
      else
      { //a 0 value
        Serial.println(F("Vent Fan - off"));
        digitalWrite(fan_ledPin, LOW); //will turn the LED (fan) off
      }
    }
//if the remote device wrote to the characteristic,
//use the value to control the pump
    if(switchCharacteristic2.written())
    {
      if(switchCharacteristic2.value())
      {                   //any value other than 0
        Serial.println("Water Pump - on");
        digitalWrite(pump_ledPin, HIGH);  //will turn the LED (pump) on
       }
       else
       { //a 0 value
        Serial.println(F("Water Pump - off"));
        digitalWrite(pump_ledPin, LOW); //turn the LED (pump) off
       }
     }
    }//end while
    Serial.print("Disconnected from central: ");
    Serial.println(central.address());
  }
}
//*****************************************************************
//void update_sensor_values() - Read the current sensor values
//on the analog input pins.
//*****************************************************************
void update_sensor_values()
{
//External greenhouse temperature
  int exttemp = analogRead(A0);
  int exttempLevel = map(exttemp, 0, 1023, 0, 100);
  if (exttempLevel != oldexttemp)
  { //ext temp level changed
    Serial.print("Ext Temp Level % is now: "); //print it
    Serial.println(exttempLevel);
    exttempChar.writeValue(exttempLevel); //update ext temp level char
    oldexttemp = exttempLevel; //save level for next comparison
  }
```

```
//Internal greenhouse temperature
  int inttemp = analogRead(A1);
  int inttempLevel = map(inttemp, 0, 1023, 0, 100);
  if (inttempLevel != oldinttemp)
  { //if int temp level changed
    Serial.print("Int Temp Level % is now: "); //print it
    Serial.println(inttempLevel);
    inttempChar.writeValue(inttempLevel); //update int temp level char
    oldinttemp = inttempLevel; //save level for next comp
  }
//Internal greenhouse humidity
  int inthumidity = analogRead(A2);
  int inthumidityLevel = map(inthumidity, 0, 1023, 0, 100);
  if (inthumidityLevel != oldhumidity)
  { //if int humidity changed
    Serial.print("Int Humidity % is now: "); //print it
    Serial.println(inthumidityLevel);
    humidityChar.writeValue(inthumidityLevel); //update int humidity level char
    oldhumidity = inthumidityLevel; //save level for next comparison
  }
//Internal soil moisture
  int intsoilmoisture = analogRead(A3);
  int intsoilmoistureLevel = map(intsoilmoisture, 0, 1023, 0, 100);
  if (intsoilmoistureLevel != oldsoilmoisture)
  { //if level changed
    Serial.print("Soil Moisture Level % is now: "); //print it
    Serial.println(intsoilmoistureLevel);
    soilmoistureChar.writeValue(intsoilmoistureLevel); //update char
    oldsoilmoisture = intsoilmoistureLevel; //save for next comparison
  }
//Rain level in internal capture barrel
  int intrainlevel = analogRead(A4);
  int intrainlevelLevel = map(intrainlevel, 0, 1023, 0, 100);
  if (intrainlevelLevel != oldrainlevel)
  { //if level changed
    Serial.print("Rain Barrel Level % is now: ");//print it
    Serial.println(intrainlevelLevel);
    rainlevelChar.writeValue(intrainlevelLevel); //update rain level char
    oldrainlevel = intrainlevelLevel; //save for next comp
  }
//Windspeed
   int intwindspeed = analogRead(A5);
   int intwindspeedLevel = map(intwindspeed, 0, 1023, 0, 100);
   if (intwindspeedLevel != oldwindspeed)
   { //if wind speed changed
    Serial.print("Wind Speed % is now: ");      //print it
    Serial.println(intwindspeedLevel);
    windspeedChar.writeValue(intwindspeedLevel); //update wind level char
    oldwindspeed = intwindspeedLevel; //save for next comp
  }
```

```
//Wind direction
  int intwinddirection = analogRead(A6);
  int intwinddirectionLevel = map(intwinddirection, 0, 1023, 0, 100);
  if (intwinddirectionLevel != oldwinddirection)
  { //if direction changed
    Serial.print("Wind Direction % is now: ");     //print it
    Serial.println(intwinddirectionLevel);
    winddirectionChar.writeValue(intwinddirectionLevel);//update char
    oldwinddirection = intwinddirectionLevel; //save for next comp
  }
  int battery = analogRead(A7);
  int batteryLevel = map(battery, 0, 1023, 0, 100);
  if (batteryLevel != oldBatteryLevel)
  {                                 //if level changed
    Serial.print("Battery Level % is now: "); //print it
    Serial.println(batteryLevel);
    batteryLevelChar.writeValue(batteryLevel); //update level char
    oldBatteryLevel = batteryLevel; //save for next comp
  }
}
//*****************************************************************
```

2.8. Выводы

В этой главе мы рассмотрели плату Arduino Nano 33 BLE Sense и ее процессор nRF52840, начиная с обзора функций платы. Затем исследовали оснащение процессора nRF52840 и связанные с ним периферийные подсистемы, а также периферийные устройства на плате Nano 33 BLE Sense. Для некоторых периферийных устройств мы предоставили краткую теорию работы, обзор функций и примеры. Глава завершается расширенным примером приложения Bluetooth BLE – системой мониторинга теплицы.

2.9. Задания

1. Какое рабочее напряжение у Arduino Nano 33 BLE Sense? Какие ограничения накладывает это рабочее напряжение на процессор?
2. Встроенные светодиоды RGB горят слабо. Что это значит?
3. Сколько внешних контактов доступно на Arduino Nano 33 BLE Sense? Какой внешний контакт используется для подачи на-

пряжения питания на Nano 33? Каков приемлемый диапазон этого напряжения[1]?

4. Составьте список встроенных датчиков Nano 33 BLE. Дайте краткое описание каждого датчика.
5. Кратко опишите функции последовательной связи Nano 33 BLE sense. Предоставьте сводную таблицу характеристик.
6. Кратко опишите разницу между EEPROM Flash и SRAM. Как эти элементы памяти используются при обычном выполнении программы?
7. Что такое широтно-импульсная модуляция (PWM, PWM)? Как она используется для управления скоростью двигателя или интенсивностью свечения светодиода?
8. В чем разница между системами аналого-цифрового (ADC) и цифроаналогового преобразования (DAC)? Как используются эти системы в приложениях микроконтроллера?
9. В чем разница между Bluetooth Classic и BLE? Совместимы ли они друг с другом?
10. Кратко опишите роль клиента и сервера в приложении Bluetooth BLE.
11. Что такое UUID Bluetooth?
12. В чем преимущество использования системного симулятора при разработке скетчей Arduino?
13. Каковы функции инерциального измерительного блока IMU?
14. Что такое барометрическое давление? Какова его единица измерения?
15. Что такое относительная влажность? Каков ее диапазон измерения?
16. Опишите датчики и их возможности, имеющиеся в APDS-9960.

Источники

1. APDS-9960 Digital Proximity, Ambient Light, RGB and Gesture Sensor Data Sheet, Avago Technologies, AV02–4191EN, 2015.

[1] Заметьте, что на последние два вопроса ответ в пройденных главах отсутствует и дается лишь в разделе 3.2 главы 3, специально посвященной подобным вопросам. – *Прим. перев.*

2. Arduino homepage, www.arduino.cc.
3. Arduino Nano 33 BLE Sense Product Reference Manual, ABX00031, 2022.
4. Bluetooth, www.bluetooth.com.
5. Bluetooth Document: 16–bit UUID Numbers Document, www.bluetooth.com, 2015.
6. Franklin Miller, Jr., College Physics, 4th edition, Harcourt, Brace, Jovanovich, 1977.
7. HTS221 – Capacitive digital sensor for relative humidity and temperature, DocID026333 Rev 4, STMicroelectronics, 2016.
8. LPS22HB – MEMS nano pressure sensor: 260–1260 hPa absolute digital output barometer, DocID027083 Rev 6 1, STMicroelectronics, 2017.
9. LSM9DS1 – iNEMO inertial module: 3D accelerometer, 3D gyroscope, 3D magnetometer, DocID025715 Rev 3, STMicroelectronics, 2015.
10. MP34DT05 – MEMS audio sensor omnidirectional digital microphone data sheet, STMicroelectronics, 2019.
11. nRF52840 Product Specification, 4413_417 v1.1, Nordic Semiconductor, 2019.
12. nRF52840 Advanced multi–protocol System–on–Chip Supporting: Bluetooth low energy (Bluetooth 5), ANT/ANT+, 802.15.4 and 2.4GHz proprietary, Nordic Semiconductor.

ГЛАВА

3

Arduino Nano 33 BLE Sense: питание и сопряжение с внешними устройствами

Прочитав эту главу, читатель должен уметь:

- указать параметры питания для системы на базе Arduino;
- описать входные/выходные параметры напряжения и тока для Arduino Nano 33 BLE Sense;
- применять параметры входного/выходного напряжения и тока для правильного сопряжения устройств ввода и вывода с платой Arduino Nano 33 BLE Sense;
- осуществлять сопряжение выбранных устройств ввода и вывода с платой Arduino Nano 33 BLE Sense;
- описать, как управлять скоростью и направлением двигателя постоянного тока.

3.1. Обзор

Мы начнем эту главу с изучения требований к источнику питания для Arduino Nano 33 BLE Sense и рассмотрим, как обеспечить питание из нескольких различных источников. Оставшаяся часть главы

содержит информацию о том, как подключить некоторые входные и выходные, в том числе мощные, устройства постоянного тока к плате Nano 33. Как упоминалось ранее, Nano 33 BLE Sense содержит процессор с напряжением 3,3 В постоянного тока. Напоминаем, что входные сигналы платы не должны превышать это значение.

3.2. Требования к питанию Arduino

Платы Arduino могут получать питание от порта USB во время разработки проекта. Настоятельно рекомендуется использовать внешний источник питания каждый раз при подключении периферийного компонента. Это позволит разрабатывать проекты, выходящие за рамки ограниченных возможностей питания порта USB.

Для платы Nano 33 BLE Sense внешние регулируемые напряжения постоянного тока от 5 до 18 В могут быть поданы на вывод питания VIN (контакт 15). Убедитесь, что общий (отрицательный) провод источника питания также подключен к одному из контактов «ground» (GND) платы Nano 33 (контакты 14 или 19) [1].

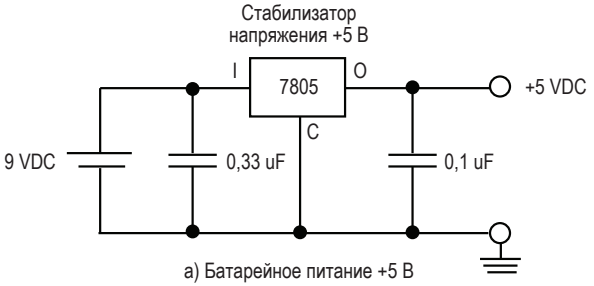
3.3. Стабилизаторы напряжения

Стабилизаторы напряжения обеспечивают стабилизированное фиксированное выходное напряжение при изменении входного напряжения. Распространенным стабилизатором положительного напряжения является серия 78XX. «XX» указывает выходное напряжение регулятора (например, 5, 9, 12 и т. д.). Эта серия имеет номинальный ток до 1,5 А. Входное напряжение стабилизатора обычно должно быть на несколько вольт выше желаемого выходного напряжения[1].

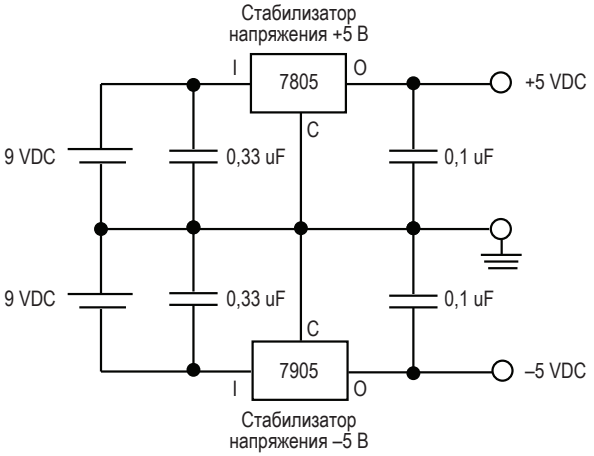
---

[1] В плате Nano 33 установлен импульсный стабилизатор MPM3610. Он формально допускает ток до 1,2 А и входное напряжение до 21 В. Однако, несмотря на достаточно высокий КПД импульсных стабилизаторов (до 80 % при входном напряжении 18–20 В), максимальная рассеиваемая мощность MPM3610 составляет около 2,7 Вт, что приводит к ограничениям по максимальному току и/или входному напряжению. Серия аналоговых стабилизаторов 78xx, о которой упоминает автор, – первая массовая серия интегральных стабилизаторов, была распространена в 1970–1980 гг. и в настоящее время в промышленных изделиях не используется из-за неудовлетворительных характеристик. – *Прим. перев.*

На рис. 3.1 представлены примеры схем для обеспечения напряжений питания +5 В постоянного тока и ±5 В постоянного тока от портативной батареи 9 В.



а) Батарейное питание +5 В



б) Батарейное питание ±5 В

Рис. 3.1. Схемы питания с использованием батарей напряжением 9 В и стабилизаторов напряжения 5 В[1]

[1] Вместо устаревших стабилизаторов 7805 для положительного напряжения +5 В рекомендуется использовать LM1117-5.0 или L4940V5. Для питания 3,3 В автор далее использует LM1084 (см. рис. 3.13). Доступной и недорогой замены для отрицательного стабилизатора 7905, к сожалению, в отечественной рознице не имеется. Отметим, что перед цифровым обозначением всех перечисленных типов может идти различный буквенный префикс, обычно указывающий на производителя (например, AMS1117 вместо LM1117 и т. п.). – *Прим. перев.*

3.3.1. Питание Nano 33 от батарей

Как упоминалось ранее, для Arduino Nano 33 рекомендуется источник питания 5–18 В постоянного тока. Для приложений с низким энергопотреблением можно использовать одну батарею на 9 В постоянного тока (рис. 3.1) и разъем, показанный на рис. 3.2, *а*. Для приложений с более высокой мощностью можно использовать батарейный блок с 6 элементами типа AA (рис. 3.2, *б*, *в*). Питание от батареи подается на стабилизатор (например, 7805), а затем на контакт Nano 33 VIN[1].



а) Разъем для батареи 9 В («Крона»)

б) Батарейный отсек 6 шт. элементов AA (плоский)

в) Батарейный отсек 6 шт. элементов AA (квадратный)

Рис. 3.2. Питание Arduino от батарей 9 В постоянного тока

3.4. Концепции сопряжения с внешними устройствами

В остальной части главы мы обсудим методы правильного подключения некоторых периферийных устройств к Nano 33 BLE Sense. Список правил подключения, которые мы соблюдаем, краток, но важен. Нарушение любого из этих правил может привести к повреждению микроконтроллера, периферии или к нестабильной работе. Вот эти правила:

---

[1] Так как плата Nano 33 допускает питание до 18 В, для нее одной стабилизатор перед VIN не требуется, он уже имеется на плате. Автор, очевидно, имеет в виду распространенный случай, когда от батарейного источника питается не только плата контроллера, но и какая-то внешняя более мощная нагрузка с необходимым напряжением 5 В. – *Прим. перев.*

- Nano 33 BLE Sense, включая установленный процессор, питается напряжением 3,3 В постоянного тока;
- входные сигналы процессора (цифровые и аналоговые) не должны превышать значение 3,3 В постоянного тока;
- аксимальный ток, доступный на выходном контакте, составляет 10 мА;
- настоятельно рекомендуется использовать внешний источник питания при каждом подключении периферийного компонента.

Имея этот список правил, исследуем некоторые устройства ввода и вывода.

3.5. Устройства ввода

В этом разделе мы опишем, как подключить некоторые устройства ввода к микроконтроллеру Arduino Nano 33 BLE Sense.

3.5.1. Переключатели и кнопки

Переключатели бывают разных типов. Как разработчик системы вы должны выбрать подходящий переключатель для конкретного применения. Разновидности переключателей и кнопок, обычно используемые совместно с микроконтроллерами, показаны на рис. 3.3, *а*. Вот краткий обзор различных типов.

- **Ползунковый переключатель**. Ползунковый переключатель имеет два разных положения: включено и выключено. Переключатель вручную перемещается в то или иное положение. Для микроконтроллеров доступны ползунковые переключатели, которые соответствуют размерам стандартного двухрядного корпуса интегральной схемы DIP. Обычно доступен блок из четырех или восьми DIP-переключателей в одном корпусе (рис. 3.3, *а*, крайний слева), встречаются также сдвоенные и одинарные (см. рис. 6.7). Ползунковые переключатели используются для выбора определенных параметров при запуске системы.

- **Кнопка без фиксации.** Кнопка без фиксации поставляется в двух вариантах: нормально замкнутая (normally closed, NC) и нормально разомкнутая (normally open, NO). Нормально разомкнутая кнопка, как следует из названия, изначально не обеспечивает электрического соединения между своими контактами. При нажатии кнопки устанавливается соединение между двумя контактами, удерживаемое до тех пор, пока кнопка нажата. При отпускании контакт разрывается. Обратное верно для нормально замкнутой кнопки. Для приложений с микроконтроллерами доступны кнопки миниатюрной конфигурации тактового типа (рис. 3.3, *а*, второй слева).
- **Нажимные выключатели (кнопки с фиксацией)**. Эти типы кнопок (пример см. рис. 3.3, *а*, второй справа) также доступны в нормально замкнутой или нормально разомкнутой конфигурации. В нормально разомкнутой конфигурации выключатель нажимается для установления соединения между двумя контактами. Чтобы разорвать соединение, кнопку необходимо нажать еще раз.
- **Шестнадцатеричные поворотные переключатели**. Для микроконтроллеров доступны поворотные переключатели небольшого размера (рис. 3.3, *а*, крайний справа). Эти переключатели обычно имеют шестнадцать положений поворотного элемента (движка). При повороте движка в каждое положение на контактах переключателя создается уникальный четырехбитный двоичный код.

Общий принцип подключения контактов переключателя или кнопки показан на рис. 3.3, *б*. Этот метод позволяет правильно установить логическую единицу или ноль на входном выводе микроконтроллера. Базовый интерфейс состоит из контактов, включенных последовательно с токоограничивающим резистором. Соединение между переключателем и резистором подключается к входному выводу микроконтроллера. В показанной конфигурации резистор подтягивает вход микроконтроллера к напряжению питания VDD, равному 3,3 В постоянного тока. Когда переключатель замкнут, место соединения замыкается через контакты на общий провод, и на входной вывод микроконтроллера подается логический ноль. Чтобы изменить логику работы контактов (т. е. получить лог. ноль в разомкнутом состоянии и лог. единицу в замкнутом), резистор и контакты просто меняются местами.

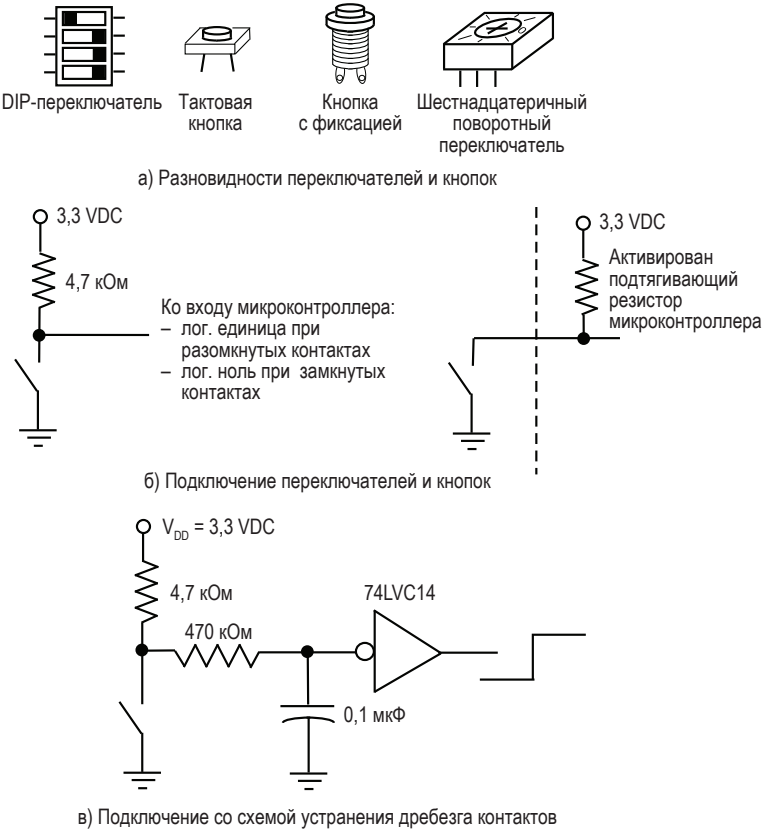


Рис. 3.3. Подключение переключателей и кнопок

3.5.1.1. Устранение дребезга контактов

Механические переключатели и кнопки не обеспечивают плавного перехода из одного положения (включено) в другое (выключено). Когда переключатель перемещается из одного положения в другое, он несколько раз замыкает и размыкает контакт (дребезжит). Этот процесс может продолжаться десятки миллисекунд, а микроконтроллер реагирует гораздо быстрее. Таким образом, микроконтроллер способен ошибочно распознавать дребезг контактов как серию отдельных переключений.

Для устранения явления дребезга контактов можно использовать дополнительные внешние аппаратные компоненты или

программные методы. Аппаратная схема устранения дребезга показана на рис. 3.3, *в*. Место соединения токоограничительного резистора и контакта подключается не напрямую ко входу микроконтроллера, как на рисунке выше, а через фильтр нижних частот (ФНЧ), образованный резистором 470 кОм и конденсатором 0,1 мкФ, ко входу триггера Шмитта 74LVC14, представляющего собой просто инвертор с гистерезисом[1]. ФНЧ предотвращает резкие изменения входного сигнала микроконтроллера, а наличие триггера Шмитта еще больше ограничивает дребезг переключателя.

Дребезг контактов также можно устранить с помощью программных методов. Это достигается путем введения задержки срабатывания от 30 до 50 мс в функцию, реагирующую на изменения контактов порта. Задержка не позволяет микроконтроллеру реагировать на множественные переключения, связанные с дребезгом[2].

Вы должны тщательно проанализировать данную конструкцию, чтобы определить, будут ли использоваться аппаратные или программные методы устранения дребезга переключателей. Важно помнить, что все переключатели демонстрируют явление дребезга и, следовательно, их необходимо устранять. Пример представлен в Arduino IDE в разделе **Examples** (*Примеры*) -> **Digital** (*Цифровые*) -> **Debounce** (*Антидребезг*).

---

[1] Микросхему 74LVC14 (питанием от 1,2 до 3,6 В) можно заменить на другие микросхемы КМОП-серий, содержащие триггеры Шмитта, например на 74HC14, 74HC132, 74AC14 и т. д. (с питанием 2–6 В). Следует учитывать, что во всех этих микросхемах содержится не один инвертирующий элемент Шмитта, а сразу 6 (74xx14) или 4 (74xx132). Входы (но не выходы!) незадействованных элементов в корпусе нельзя оставить «висящими в воздухе», а следует подключать к фиксированному уровню (GND или питанию), что при необходимости подключения всего одной пары контактов усложняет и загромождает схему. В связи с этим представляет интерес буфер с триггером Шмитта без инверсии уровня 74LVC1G17 (питание 1,65–5,5 В), содержащий только один элемент. – *Прим. перев.*

[2] Более надежный метод программного антидребезга заключается в опросе состояния входного порта до тех пор, пока в течение заданного времени не совпадут несколько последовательных отсчетов, что свидетельствует об установлении уровня. Подобный метод применяется в Arduino-библиотеке Bounce (описание на русском, например, по адресу: https://greenoakst.blogspot.com/2012/06/arduino-bounce.html). – *Прим. перев.*

3.6. Выходные устройства

Внешнее устройство не следует подключать к микроконтроллеру без предварительного тщательного анализа интерфейса, чтобы убедиться в требованиях к напряжению, току и синхронизации микроконтроллера и внешнего устройства. В этом разделе мы описываем особенности подключения некоторых внешних устройств. Начнем с интерфейса для одного светодиода.

3.6.1. Светоизлучающие диоды (LED)

Светодиод обычно используется в качестве логического индикатора, информирующего о наличии логической единицы или логического нуля на определенном выводе микроконтроллера. Светодиод имеет два вывода: анод (положительный вывод) и катод (отрицательный вывод). Чтобы правильно подключить светодиод, анодный вывод должен иметь напряжение на уровне примерно на 1,7–2,2 В выше, чем катодный вывод. Эта характеристика известна как прямое напряжение (V_f) светодиода. Ток светодиода также должен быть ограничен безопасной величиной, называемой прямым током светодиода (I_f). Характеристики прямого напряжения и максимально допустимого прямого тока диода обычно предоставляются производителем.

Пример схемы подключения светодиода представлен на рис. 3.4. Логическая единица подается микроконтроллером на базу NPN-транзистора малой мощности (например, PN2222 или MPQ2222[1]) через базовый резистор номиналом 10 кОм. Вывод эмиттера транзистора заземлен. К выводу коллектора транзистора последовательно со светодиодом подключен резистор номиналом 220 Ом к источнику питания 3,3 В постоянного тока. Резистор номиналом 220 Ом ограничивает ток через светодиод. Правильное значение сопротивления резистора R можно рассчитать, используя формулу $R = (Vcc - V_f)/I_f$[2].

---

[1] Можно заменить на более доступный на отечественном рынке BC337 (любой разновидности) или на отечественный КТ3102 (с любой буквой). – *Прим. перев.*

[2] Отметим, что прямое напряжение светодиода V_f связано с прямым током I_f – оно увеличивается с увеличением последнего (узнать конкретные значения можно из графиков в документации производителя).

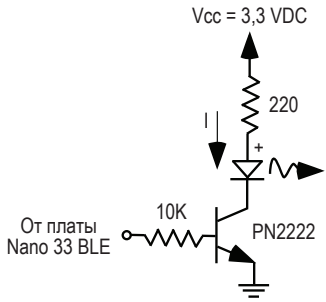


Рис. 3.4. Устройство светодиодной индикации

3.6.2. Жидкокристаллический дисплей (ЖК-дисплей, LCD)

ЖК-дисплей – это устройство вывода для отображения текстовой информации. ЖК-дисплеи бывают самых разных конфигураций, включая многосимвольный и многострочный формат. Широко распространен ЖК-формат 16×2, то есть он имеет возможность отображать две строки по 16 символов каждая. Каждый символ и строка дисплея имеют определенный адрес. Символы передаются на ЖК-дисплей в формате американского стандартного кода обмена информацией (ASCII) по одному символу за раз. Подключение ЖК-дисплеев с различными интерфейсами к Nano 33 BLE Sense подробно обсуждалось в главе 2 для способов параллельной и последовательной связи через порты USART, SPI и I2C.

3.7. Принципы управления двигателем

В этом разделе мы рассмотрим различные типы двигателей постоянного тока. Двигатели можно использовать для передвижения, позиционирования датчиков или исполнительных механизмов. Для каждого типа двигателя мы предоставляем базовую теорию работы, методы интерфейса микроконтроллера и примеры при-

---

Кроме того, в общем случае синие/белые светодиоды имеют большее значение прямого напряжения, чем красные/желтые/зеленые. При максимально допустимом прямом токе падение напряжения V_f на синем светодиоде может составить до 4 В, то есть превысить напряжение питания *V*cc = 3,3 В, и светодиод не загорится вовсе. – *Прим. перев.*

ложений. Некоторые типы двигателей, доступные для различных применений, показаны на рис. 3.5.

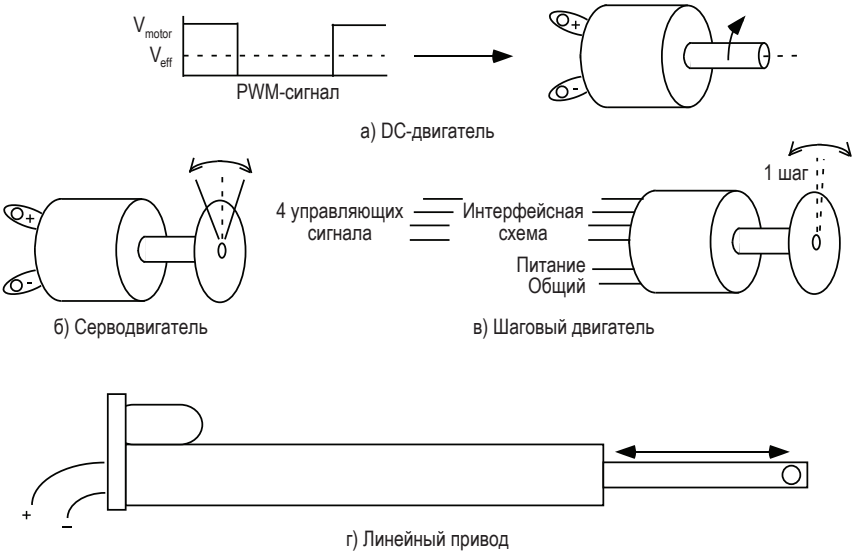


Рис. 3.5. Типы двигателей

- **Двигатель постоянного тока** (DC). Двигатель постоянного тока имеет положительный и отрицательный выводы. Когда к DC-двигателю подключается источник постоянного тока с подходящим напряжением и номинальным током, он начинает вращаться. Если полярность питания переключить относительно выводов двигателя, двигатель будет вращаться в противоположном направлении. Скорость двигателя примерно пропорциональна приложенному напряжению вплоть до номинального напряжения двигателя[1].
- **Серводвигатель**. Серводвигатель обеспечивает точное угловое вращение для заданного коэффициента заполнения широтно-импульсной модуляции (PWM). Поскольку коэффици-

[1] Скорость вращения DC-двигателя в равной степени зависит от напряжения питания и от нагрузки на валу, образуя семейство *регулировочных характеристик* (см., например, https://www.asutpp.ru/elektrodvigatel-postoyannogo-toka.html). – *Прим. перев.*

ент заполнения приложенного сигнала изменяется, угловое смещение двигателя также меняется. Это позволяет использовать двигатель в приложениях для изменения механических положений, таких как угол поворота колеса, положение или сканирование датчика, или в качестве основного приводного двигателя для мини-робота.

- **Шаговый двигатель.** Шаговый двигатель, как следует из его названия, обеспечивает постепенное ступенчатое вращение (обычно 2,5° на шаг) в соответствии со ступенчатым изменением последовательности управляющих сигналов. Шаговый двигатель обычно управляется по двух- или четырехпроводному интерфейсу. Для четырехпроводного шагового двигателя микроконтроллер обеспечивает четырехбитную последовательность управляющих сигналов для вращения двигателя по часовой стрелке. Чтобы повернуть двигатель против часовой стрелки, последовательность управляющих сигналов меняется на обратную. Сигналы управления малой мощности от микроконтроллера подаются на двигатель через МОП-транзисторы или силовые транзисторы, дабы обеспечить надлежащие требования к напряжению и току импульсов. Шаговый двигатель можно использовать, например, для позиционирования датчиков робота.
- **Линейный привод**. Линейный привод на самом деле представляет собой обычный двигатель постоянного тока, оснащенный шестернями для преобразования вращательного движения в линейное[1]. Линейный привод используется, когда требуется повторяемое линейное движение, как толкающее, так и тянущее. Их можно использовать в роботах для рулевого управления или для размещения датчиков.

Закончив этот краткий обзор двигателей, давайте более подробно рассмотрим управление скоростью двигателя постоянного тока.

---

[1] Необходимо заметить, что существуют также и линейные двигатели, осуществляющие поступательное движение чисто электромеханическим способом, без преобразования вращения в линейное перемещение за счет редукторов. Линейный двигатель по принципу устройства представляет собой разновидность обычного электродвигателя (например, шагового или DC-двигателя), у которого статор развернут в одной плоскости (см. статью «Линейный двигатель» на ru.wikipedia.org). – *Прим. перев.*

3.7.1. Двигатель постоянного тока

Двигатель постоянного тока, или DC-двигатель, обычно используется в роботах в качестве основного источника передвижения. Источником питания двигателя постоянного тока чаще является встроенный аккумулятор робота. Мы обсудим выбор питания двигателей робота в разделе 3.8.2.

Двигатель постоянного тока имеет положительный и отрицательный выводы. Когда к DC-двигателю подключается источник постоянного тока с подходящим напряжением и номинальным током, он начинает вращаться. Если полярность питания переключить относительно выводов двигателя, двигатель будет вращаться в противоположном направлении. Скорость двигателя примерно пропорциональна приложенному напряжению вплоть до номинального напряжения двигателя.

3.7.1.1. Характеристики двигателей постоянного тока

Двигатель постоянного тока оценивается с использованием следующих общих параметров. Требования к конструкции применяются для выбора соответствующего двигателя [3, 4].

- **Напряжение.** Максимальное рабочее напряжение двигателя указывается в вольтах постоянного тока (В, VDC). При этом напряжении двигатель вращается с максимальной скоростью, измеряемой в оборотах в минуту (об/мин).
- **Ток.** При вращении двигатель потребляет ток от источника постоянного тока. Потребляемый ток, измеряемый в амперах (А), зависит от нагрузки, которую испытывает двигатель. Для двигателей постоянного тока заданы следующие токи: пусковой ток, рабочий ток холостого хода и тормозной ток. При проектировании интерфейса двигателя он должен выдерживать все эти значения тока.
  - **Пусковой ток** – импульсный ток, возникающий при первом запуске двигателя. Скачок происходит из-за преодоления двигателем механической инерции.
  - Рабочий **ток холостого хода** – значение тока, потребляемого двигателем при номинальном напряжении и отсутствии механической нагрузки.

  - **Тормозной ток** (длительный ток при заторможенном роторе) – это ток, потребляемый от источника питания при остановке двигателя из-за механической перегрузки.

- **Скорость вращения** двигателя указывается в оборотах в минуту (об/мин).
- **Крутящий момент** – угловая сила, создаваемая двигателем на определенном расстоянии по радиусу от вала двигателя. Измеряется в единицах системы МКС[1] – ньютон-метрах (Н·м).
- **Зависимость скорости от крутящего момента**: двигатели постоянного тока имеют несколько различных типов конфигураций. Зависимость скорости от крутящего момента у них различная, но в общем случае можно утверждать, что когда скорость вала двигателя уменьшается (например, с помощью зубчатых передач), крутящий момент двигателя увеличивается.
- **Эффективность**. Двигатель постоянного тока преобразует электрическую энергию постоянного тока в механическую энергию. В идеале мы хотим, чтобы вся электрическая энергия преобразовывалась в механическую, обеспечивая 100%-ный КПД. КПД определяется как отношение механической мощности к электрической мощности.
- **Редуктор**. Многие двигатели постоянного тока оснащены редуктором с шестернями, которые используются для уменьшения скорости вращения вала при одновременном увеличении крутящего момента двигателя.

3.7.1.2. Однонаправленное управление двигателем постоянного тока

В этом разделе мы опишем интерфейс между маломощным микроконтроллером Arduino Nano 33 BLE Sense и двигателем постоянного тока. Напомним, что цифровой выходной контакт Nano 33 BLE Sense ограничен напряжением 3,3 В постоянного тока с лимитом тока 10 мА.

Этих значений напряжения и тока недостаточно для прямого питания двигателей любого типа. Следовательно, необходим

[1] МКС (метр–килограмм–секунда) – система единиц измерения, составная часть международной системы единиц (СИ), в настоящее время самостоятельного значения не имеет. – *Прим. перев.*

интерфейс для повышения напряжения и тока до значений, соответствующих заданной спецификации двигателя. Два распространенных метода предполагают использование конфигурации транзистора Дарлингтона или мощного полевого транзистора структуры металл–оксид–полупроводник (MOSFET).

NPN-транзистор Дарлингтона

Конфигурация Дарлингтона состоит из двух транзисторов, соединенных, как схематически показано на рис. 3.6, *а*. Эмиттер первого транзистора соединен с базой второго транзистора. Эта конфигурация обеспечивает высокий коэффициент усиления по току[1]. При использовании в качестве интерфейса двигателя сигнал управления низким выходным током от Arduino усиливается до значения тока, подходящего для управления двигателем, как показано на рис. 3.6, *б*. Ряд кремниевых диодов (1N4001), каждый с падением напряжения 0,7 В постоянного тока, используется для понижения напряжения источника питания до номинального напряжения питания двигателя. Кроме того, параллельно диодной цепочке и двигателю установлен защитный диод с обратным смещением [2, 5].

Пример **Dagu Magician – популярный недорогой двухплатформенный робот. Робот оснащен двумя приводными двигателями с напряжением 6 В постоянного тока, максимальным током 400 мА и тормозным током 1,5 А. В этом приложении двигатели робота питаются от внешнего источника питания 9 В постоянного тока через гибкий кабель.**

Мы предполагаем, что выходное напряжение вывода Nano 33 в состоянии высокого уровня (V_{OH}) равно 3,3 В постоянного тока с максимальным выходным током примерно 10 миллиампер (зададимся величиной I_B = 5 мА). С учетом этой информации значение

---

[1] Заметим, что на схематическом рис. 3.6, *а* структура транзистора Дарлингтона показана не полностью, но это не имеет принципиального значения, так как автор использует транзисторы Дарлингтона в виде законченного компонента (см. далее). Для управления мощной нагрузкой удобнее применять MOSFET-транзисторы (о подключении двигателя см., например, статью по адресу https://cxem.net/arduino/arduino11.php), но они имеют свои особенности, и более подробное рассмотрение этого вопроса выходит за рамки данной книги. – *Прим. перев.*

базового резистора транзистора Дарлингтона (R_B) получается примерно равным 380 Ом[1]:

$$-V_{OH} + (I_B \times R_B) + (2 \times V_{BE}) = 0;$$

$$-3.3 + (0.005 \times R_B) + (2 \times 0.7) = 0;$$

откуда R_B = 380 Ом.

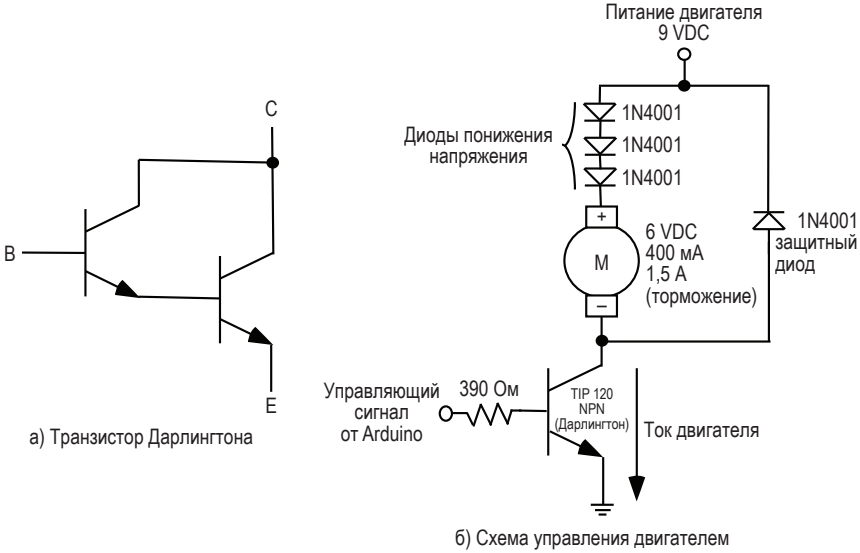


Рис. 3.6: *а* – конфигурация Дарлингтона; *б* – схема управления двигателем

Мы используем резистор близкого стандартного значения 390 Ом, как показано на рис. 3.6, *б*. Транзистор Дарлингтона NPN (TIP 120[2]) повышает ток базы до значения тока коллектора, подходящего для питания двигателя. Для каждого двигателя требуется отдельная подобная интерфейсная схема.

---

[1] В формуле ниже значение V_{BE} – падение напряжения перехода база–эмиттер кремниевого транзистора (0,7 В); так как в дарлингтоновской структуре (рис. 3.6, *а*) включены два таких перехода последовательно, то эта величина в формуле умножается на 2. – *Прим. перев.*

[2] Доступны на отечественном рынке. – *Прим. перев.*

3.7.1.3. Управление скоростью двигателя постоянного тока – широтно-импульсная модуляция (PWM)

Как упоминалось ранее, скорость двигателя постоянного тока можно изменять путем изменения приложенного напряжения постоянного тока. Однако для этой цели можно также применить метод широтно-импульсной модуляции (PWM). С помощью PWM-сигнала интерфейсная схема двигателя обеспечивает подачу напряжения с заданным коэффициентом заполнения. Как показано на рис. 3.7, коэффициент заполнения PWM-сигнала определяет долю напряжения питания, т. е. величину эффективного напряжения постоянного тока, приложенного к двигателю, и, следовательно,

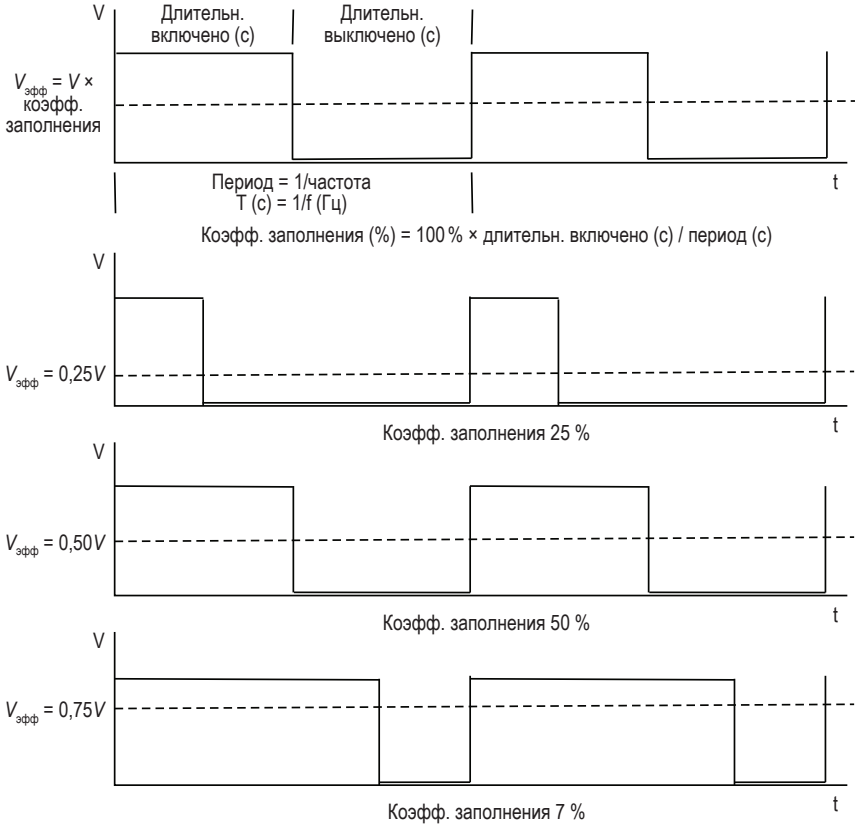


Рис. 3.7. Широтно-импульсная модуляция (PWM)

определенный процент номинальной полной скорости, с которой двигатель будет вращаться.

Плата Nano 33 BLE Sense оснащена пятью каналами широтно-импульсной модуляции. Выход PWM может быть обеспечен на цифровых контактах 1–13 и аналоговых контактах A0–A7. Мы ограничиваем использование контактами 21, 23, 24, 27 и 28, как показано на схеме контактов на рис. 2.2. Базовая частота PWM-сигнала Nano 33 BLE Sens установлена на уровне 500 Гц.

3.8. Приложение: Dagu Magician робот

Автономный робот, перемещающийся по лабиринту, оснащен датчиками для обнаружения наличия стен лабиринта и навигации по лабиринту. Робот не имеет предварительных знаний о конфигурации лабиринта, он использует датчики и встроенный алгоритм для определения следующего действия. Общая цель – как можно быстрее пройти от начальной точки лабиринта до конечной, не натыкаясь на стены лабиринта (см. рис. 3.8). Стены лабиринта окрашиваются в белый цвет, чтобы обеспечить хорошую светоотражающую поверхность; тогда как пол лабиринта окрашен в матовый черный цвет, чтобы минимизировать отражение света.

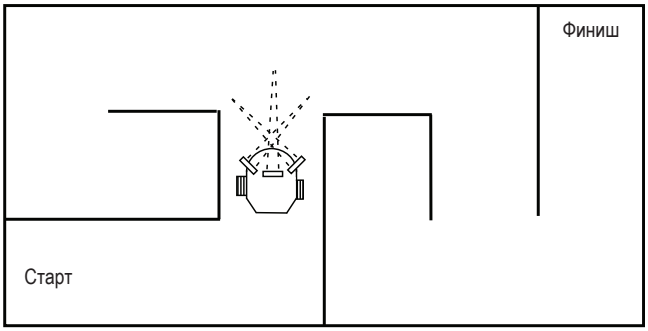


Рис. 3.8. Автономный робот в лабиринте

Было бы полезно рассмотреть основы управления роботом и управления двигателем. Рисунок 3.9 иллюстрирует фундаментальные концепции. Управление роботом зависит от количества привод-

ных колес и от того, оснащены ли колеса однонаправленным или двунаправленным управлением. Возможны дополнительные конфигурации рулевого управления. Для двунаправленного управления двигателем постоянного тока обычно требуется Н-мост[1]. В данном примере мы используем однонаправленное управление двигателем.

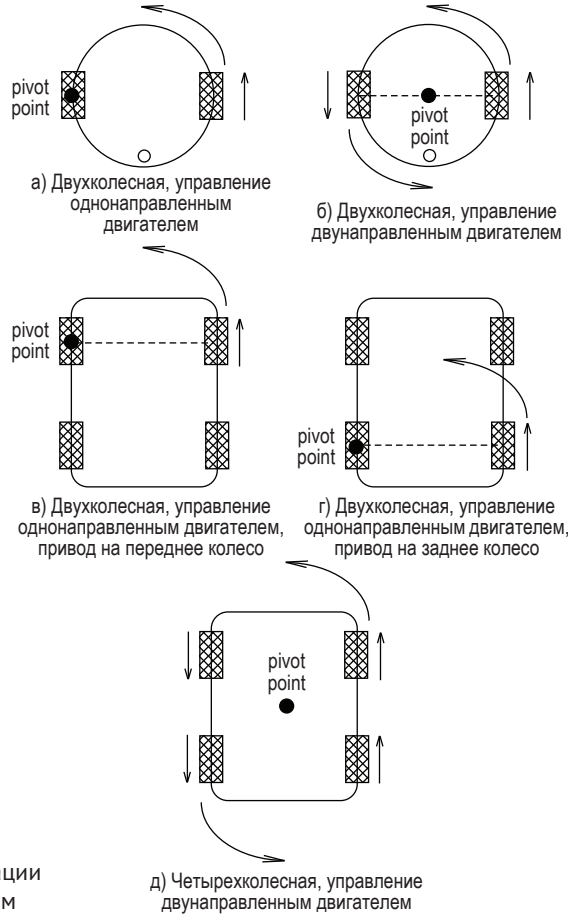


Рис. 3.9. Конфигурации управления роботом

[1] Н-мост – конфигурация из четырех управляющих ключей (например, транзисторов Дарлингтона, как на рис. 3.6, или MOSFET-транзисторов), которая позволяет переключать их попарно, меняя полярность приложенного напряжения. См. статью «Н-мост» в «Википедии» (ru.wikipedia.org). – *Прим. перев.*

Мы оснащаем робота Dagu Magician микроконтроллером Arduino Nano 33 BLE Sense для навигации по лабиринту (см. рис. 3.10).

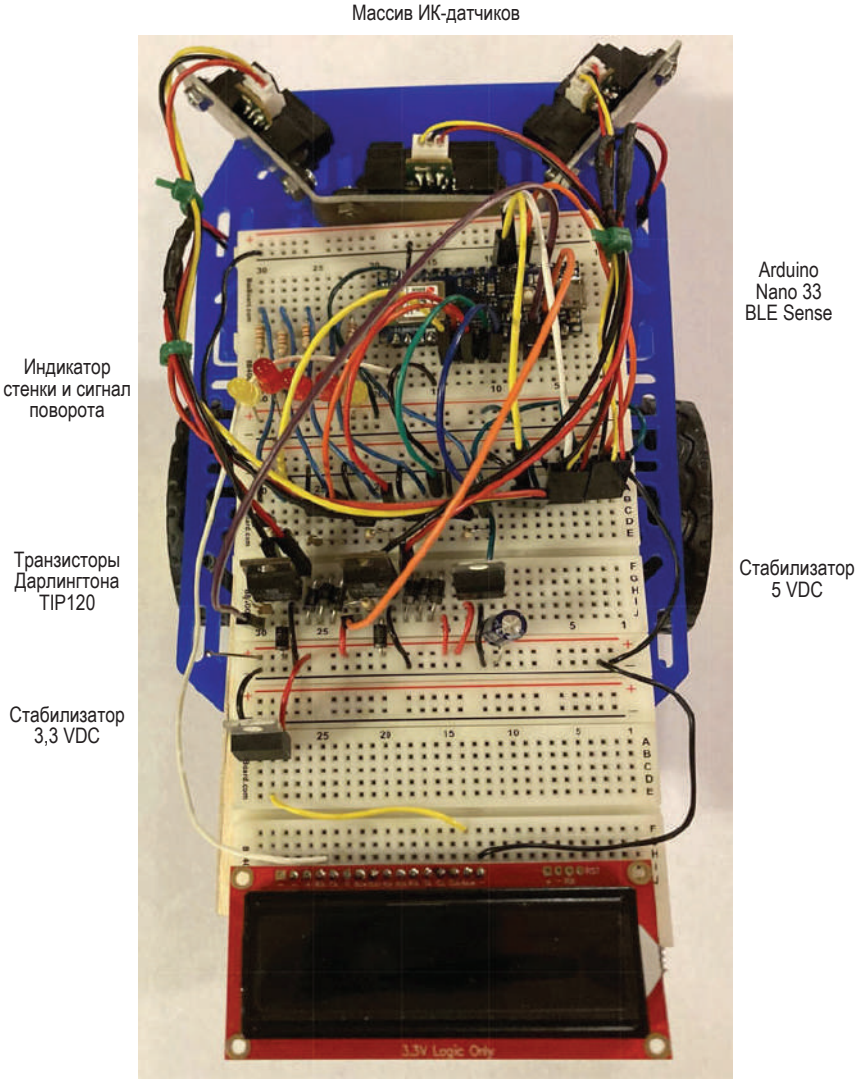


Рис. 3.10. Робот Dagu Magician

Примечание **Напоминаем: Nano 33 – это микроконтроллер с напряжением 3,3 В постоянного тока. Необходимо позаботиться о том, чтобы входное напряжение микроконтроллера не превышало 3,3 В постоянного тока, а интерфейсы периферийных устройств вывода соответствовали выходному напряжению 3,3 В постоянного тока, рассчитанному максимум на 10 мА постоянного тока на каждый контакт ввода/вывода.**

Робот управляется двумя двигателями постоянного тока напряжением 6 В, которые независимо приводят в движение левое и правое колеса. Третье колесо без привода обеспечивает устойчивость робота на платформе. Мы также оборудуем платформу тремя инфракрасными датчиками (ИК-датчиками) Sharp GP2Y0A21YK0F (см. рис. 3.11), которые можно приобрести в компании SparkFun Electronics (www.sparkfun.com)[1]. Датчики монтируем на кронштейне из тонкого алюминия (размеры указаны на рис. 3.11). В качестве альтернативы ИК-датчики можно закрепить на платформе робота с помощью L-образных кронштейнов, которые можно приобрести в местном хозяйственном магазине.

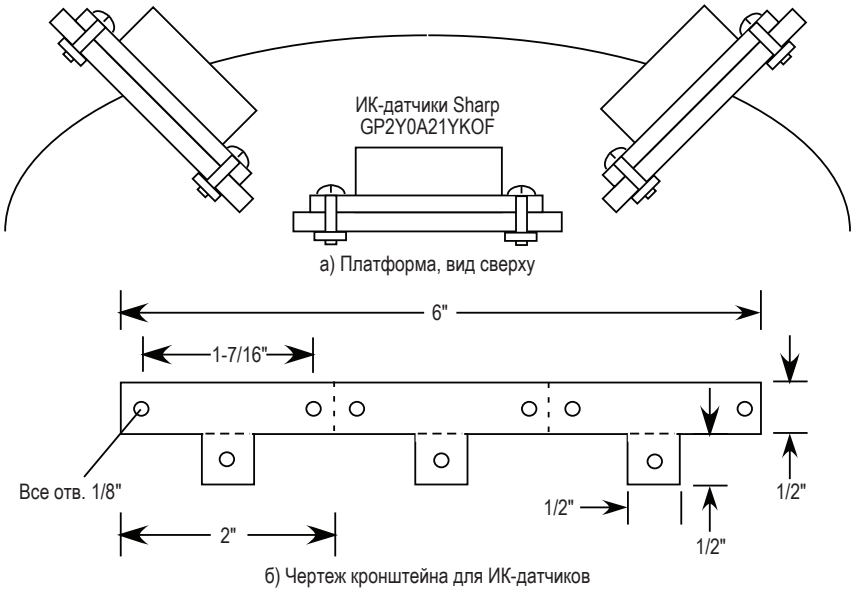


Рис. 3.11. Платформа робота Dagu Magician с добавлением трех ИК-датчиков

[1] В российских условиях датчики GP2Y0A21YK0F можно приобрести во многих интернет-магазинах, торгующих товарами для робототехники (например, в «Чип и Дип», www.chipdip.ru). – *Прим. перев.*

Характеристики ИК-датчика представлены на рис. 3.12, *а*. Обратите внимание, что датчик обеспечивает одинаковое выходное напряжение для двух разных диапазонов. Чтобы устранить эту неоднозначность, мы устанавливаем кронштейн датчика в передней части робота лицом вперед так, чтобы левый датчик контролировал наличие стены лабиринта справа от робота, и наоборот. Неоднозначность разрешается за счет размещения датчика: таким образом левый или правый датчик не может находиться ближе, чем на 5 см к правой или левой стене лабиринта соответственно. Следовательно, существует только одно значение диапазона для данного выходного напряжения каждого датчика (см. рис. 3.12, *б*).

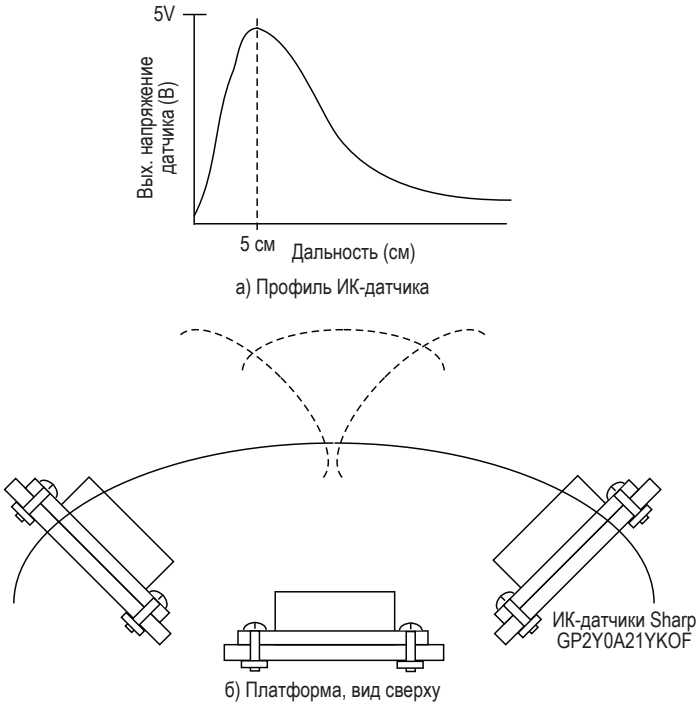


Рис. 3.12. Профиль ИК-датчика Sharp GP2Y0A21YKOF

3.8.1. Требования

Требования к этому проекту простые: робот должен автономно перемещаться по лабиринту, не касаясь стенок.

3.8.2. Принципиальная схема

Принципиальная схема робота представлена на рис. 3.13. Три ИК-датчика (левый, средний и правый) установлены на передней части робота для обнаружения лабиринтов. Выходной сигнал датчиков подается на три канала Arduino Nano 33 BLE Sense (аналоговые входы A0–A2). Двигатели робота будут управляться каналами с широтно-импульсной модуляцией (PWM) с цифровых выводов D9 и D10).

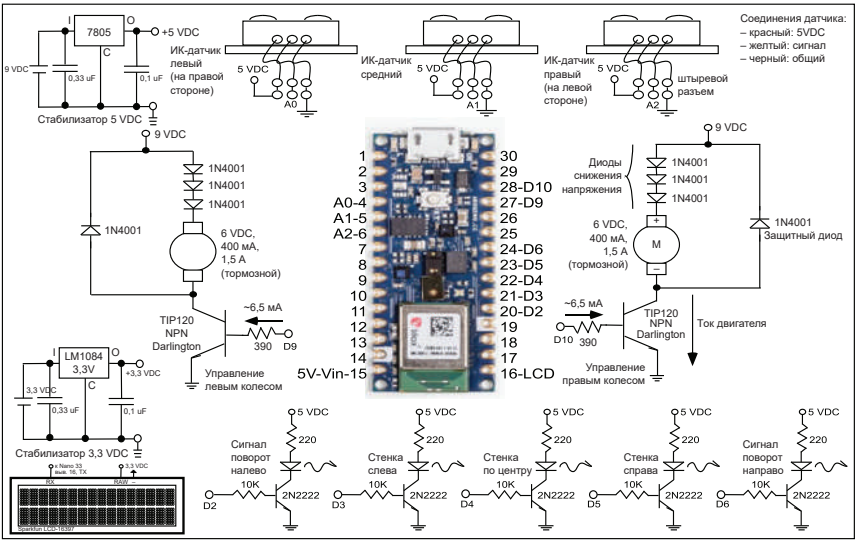


Рис. 3.13. Принципиальная схема робота

Плата Arduino Nano 33 BLE Sense подключена к двигателям через NPN-транзистор Дарлингтона (TIP120) с достаточными возможностями, чтобы удовлетворить требования к максимальному току двигателя (тормозной ток 1,5 А). Робот питается от источника питания 9 В постоянного тока. Три диода 1N4001 включены последовательно с двигателем, чтобы снизить напряжение питания двигателя примерно до 6,9 В постоянного тока. Источник питания 9 В постоянного тока также подается на стабилизатор напряжения 5 В постоянного тока для питания Arduino Nano 33 через контакт VIN (контакт 15). Чтобы сэкономить на расходе батареи, для питания рекомендуется использовать недорогой сетевой источник питания 9 В постоянного тока, 2 А. Для подачи питания роботу во

время перемещения по лабиринту можно использовать гибкий силовой кабель из плетеного провода.

Структурная схема проекта робота представлена на рис. 3.14.

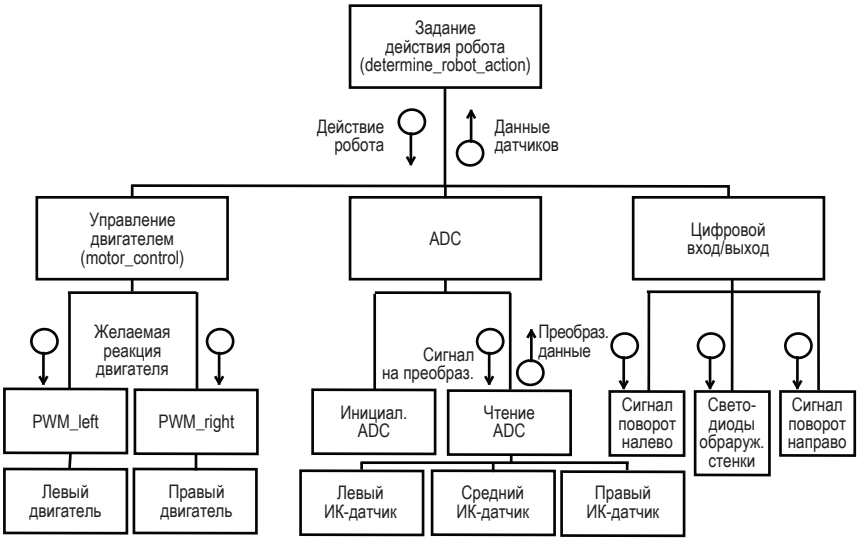


Рис. 3.14. Структурная схема робота

UML-диаграммы действий для робота представлены на рис. 3.15.

3.8.3. Алгоритм управления роботом DaguMagician

В этом разделе мы предоставляем базовую основу для алгоритма управления роботом. Алгоритм управления будет считывать показания ИК-датчиков, подключенных к аналоговому входу Arduino Nano 33 (контакты A0–A2). В ответ на обнаружение стены алгоритм подает сигналы для поворота робота, чтобы избежать стен лабиринта. На рис. 3.16 представлена таблица истинности, показывающая все возможные положения в лабиринте, с которыми может столкнуться робот. Обнаруженная стена представляется логической единицей. Сигналы на двигатели также представляются логической единицей. Как упоминалось ранее, из-за физического

размещения массива датчиков на задней кромке робота датчик, обнаруживающий стены лабиринта справа от робота, физически расположен с левой стороны, и наоборот.



Рис. 3.15. UML-диаграмма действий робота

	Левый датч.	Средн. датч.	Прав. датч.	Стена слева	Стена серед.	Стена справа	Левый двиг.	Прав. двиг.	Левый сигн.	Прав. сигн.	Коммент.
0	0	0	0	0	0	0	1	1	0	0	вперед
1	0	0	1	0	0	1	1	1	0	0	вперед
2	0	1	0	0	1	0	1	0	0	1	вправо
3	0	1	1	0	1	1	0	1	1	0	влево
4	1	0	0	1	0	0	1	1	0	0	вперед
5	1	0	1	1	0	1	1	1	0	0	вперед
6	1	1	0	1	1	0	1	0	0	1	вправо
7	1	1	1	1	1	1	1	0	0	1	вправо

Примечание: левый датчик стенки монтируется на правой стороне робота, и наоборот.

Рис. 3.16. Таблица истинности для действий робота

Учитывая используемую схему подключения, двигатели робота можно заставить вращаться только в прямом направлении. Для выполнения поворота налево левый двигатель останавливается, а правый включается до тех пор, пока робот не завершит поворот. Для выполнения поворота направо требуется обратное действие.

Задача при написании алгоритма управления состоит в том, чтобы взять UML-диаграмму действий, представленную на рис. 3.15, и действия, указанные в таблице истинности действий робота (рис. 3.16), и оформить их в скетч Arduino. Это может показаться неподъемной задачей, но мы сделаем это шаг за шагом.

Алгоритм управления начинается с задания выводов платы Arduino Nano 33. Затем объявляются переменные для показаний трех ИК-датчиков. Далее следуют две необходимые функции Arduino: `setup()` и `loop()`. В функции `setup()` нужные выводы платы Arduino Nano 33 объявляются как выходные. Функция `loop()` начинается со считывания текущего значения трех ИК-датчиков. Значение кода ADC = 512, соответствует определенному диапазону ИК-датчика. Это значение можно отрегулировать для изменения диапазона дальности, на которой обнаруживается стенка лабиринта. После чтения данных ИК-датчиков следует восемь условных операторов `if - else`. Операторы реализуют условия для каждой строки таблицы истинности, представленной на рис. 3.16. Для каждой комбинации обнаруженных стенок загораются соответствующие светодиоды, после чего подаются команды на активацию двигателей (`analogWrite()`) и включение соответствующих указателей поворота. Команда `analogWrite()` выдает сигнал от 0 до 3,3 В постоянного тока, отправляя константу от 0 до 255 с использованием методов широтно-импульсной модуляции (PWM). Команды включения поворотников обеспечивают следующие действия: мигают соответствующие поворотники и обеспечивается общая задержка 1,5 с. Это дает роботу 1,5 с на выполнение поворота. Эту задержку, возможно, потребуется скорректировать на этапе тестирования.

```
//*********************************************************************
//Nano_33_robot
//
//The robot is equipped with a Sparkfun LCD-16397
//- Nano 33 BLE Sense, USART TX pin 16 is connected to
// LCD USART RX pin
//- Provide 3.3 VDC power to the LCD
//*********************************************************************
//analog input pins
#define left_IR_sensor A0   //analog pin A0 (pin 4)-left IR sensor
```

```
#define center_IR_sensor A1 //analog pin A1 (pin 5)-ctr IR sensor
#define right_IR_sensor A2  //analog pin A2 (pin 6)-right IR sensor
//digital output pins
//LED indicators - wall detectors
#define wall_left 3         //digital pin - wall_left
#define wall_center 4       //digital pin - wall_center
#define wall_right 5        //digital pin - wall_right
//LED indicators - turn signals
#define left_turn_signal 2   //digital pin - left_turn_signal
#define right_turn_signal 6  //digital pin - right_turn_signal
//motor outputs
#define left_motor 9         //digital pin - left_motor
#define right_motor 10       //digital pin - right_motor
int left_IR_sensor_value;    //variable for left IR sensor
int center_IR_sensor_value;  //variable for center IR sensor
int right_IR_sensor_value;   //variable for right IR sensor
void setup()
{
  Serial1.begin(9600);       //Baud rate: 9600 Baud
  delay(500);                //Delay for display
//LED indicators - wall detectors
  pinMode(wall_left, OUTPUT); //configure pin 1 for digital output
  pinMode(wall_center, OUTPUT); //configure pin 2 for digital output
  pinMode(wall_right, OUTPUT); //configure pin 3 for digital output
//LED indicators - turn signals
  pinMode(left_turn_signal,OUTPUT); //configure pin 0 for digital output
  pinMode(right_turn_signal,OUTPUT);//configure pin 4 for digital output
//motor outputs Â- PWM
  pinMode(left_motor, OUTPUT); //configure pin 11 for digital output
  pinMode(right_motor, OUTPUT); //configure pin 10 for digital output
}
void loop()
{
//read analog output from IR sensors
  left_IR_sensor_value = analogRead(left_IR_sensor);    //pin 4 A0
  center_IR_sensor_value= analogRead(center_IR_sensor); //pin 5 A1
  right_IR_sensor_value = analogRead(right_IR_sensor);  //pin 6 A2
//Clear LCD
//Cursor to line one, character one
  Serial1.write(254);                //Command prefix
  Serial1.write(128);                //Command
//clear display
  Serial1.write(" ");
  Serial1.write(" ");
//Cursor to line one, character one
  Serial1.write(254);                //Command prefix
  Serial1.write(128);                //Command
  Serial1.write("Left Ctr Right");
  delay(50);
  Serial1.write(254);                //Command to LCD
  delay(5);
```

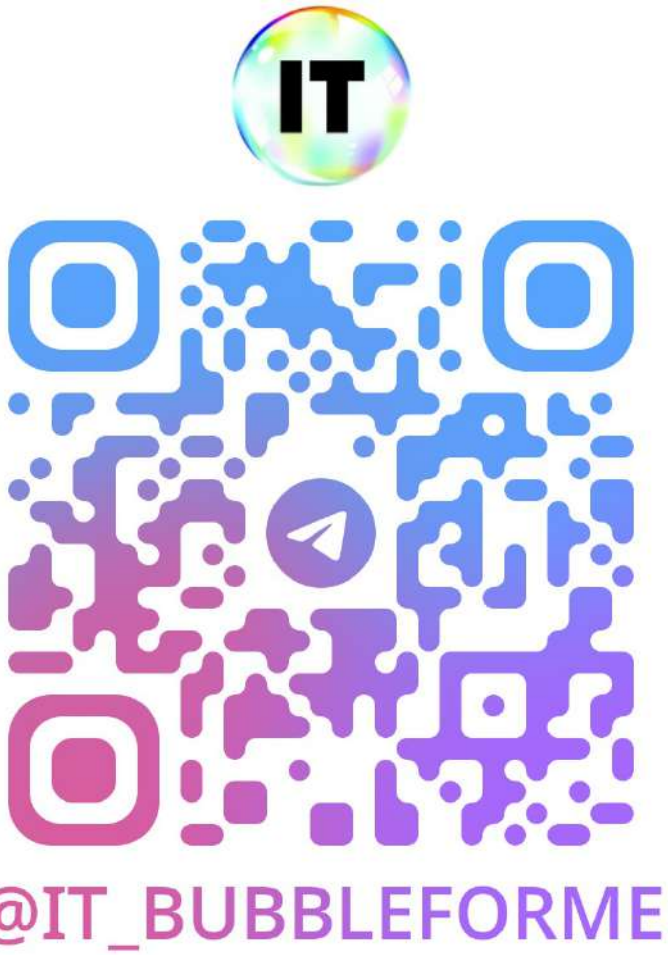
```
  Serial1.write(192);               //Cursor line 2, position 1
  delay(5);
  Serial1.print(left_IR_sensor_value);
  delay(5);
  Serial1.write(254);               //Command to LCD
  delay(5);
  Serial1.write(198);               //Cursor line 2, position 8
  delay(5);
  Serial1.print(center_IR_sensor_value);
  delay(5);
  Serial1.write(254);               //Command to LCD
  delay(5);
  Serial1.write(203);               //Cursor line 2, position 13
  delay(5);
  Serial1.print(right_IR_sensor_value);
  delay(5);
  delay(500);
//robot action table row 0
  if((left_IR_sensor_value < 512)&&(center_IR_sensor_value < 512)&&
(right_IR_sensor_value < 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, LOW);     //turn LED off
    digitalWrite(wall_center, LOW);   //turn LED off
    digitalWrite(wall_right, LOW);    //turn LED off
//motor control
    analogWrite(left_motor, 128);     //0 (off) to 255 (full speed)
    analogWrite(right_motor, 128);    //0 (off) to 255 (full speed)
//turn signals
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500);                           //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    analogWrite(left_motor, 0);           //turn motor off
    analogWrite(right_motor,0);           //turn motor off
  }
//robot action table row 1
  else if((left_IR_sensor_value < 512)&&(center_IR_sensor_value < 512)&&(right_IR_sensor_value
> 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, LOW);     //turn LED off
    digitalWrite(wall_center, LOW);   //turn LED off
    digitalWrite(wall_right, HIGH);   //turn LED on
```

```
//motor control
    analogWrite(left_motor, 128);     //0(off) to 255(full)
    analogWrite(right_motor, 128);    //0(off) to 255(full)
//turn signals
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    analogWrite(left_motor, 0);            //turn motor off
    analogWrite(right_motor,0);            //turn motor off
  }
//robot action table row 2
  else if((left_IR_sensor_value < 512)&&(center_IR_sensor_value > 512)&&(right_IR_sensor_value
< 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, LOW);       //turn LED off
    digitalWrite(wall_center, HIGH);    //turn LED on
    digitalWrite(wall_right, LOW);      //turn LED off
//motor control
    analogWrite(left_motor, 128);       //0(off) to 255 (full)
    analogWrite(right_motor, 0);        //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED on
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED on
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    analogWrite(left_motor, 0);            //turn motor off
    analogWrite(right_motor,0);            //turn motor off
  }
//robot action table row 3
  else if((left_IR_sensor_value < 512)&&(center_IR_sensor_value > 512)&&(right_IR_sensor_value
> 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, LOW);          //turn LED off
    digitalWrite(wall_center, HIGH);       //turn LED on
```

```
    digitalWrite(wall_right, HIGH);        //turn LED on
    analogWrite(left_motor, 0);            //0(off) to 255 (full)
    analogWrite(right_motor, 128);         //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, HIGH);  //turn LED on
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, HIGH);  //turn LED on
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    analogWrite(left_motor, 0);          //turn motor off
    analogWrite(right_motor,0);          //turn motor off
  }
//robot action table row 4
  else if((left_IR_sensor_value > 512)&&(center_IR_sensor_value < 512)&&(right_IR_sensor_value
< 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, HIGH);       //turn LED on
    digitalWrite(wall_center, LOW);      //turn LED off
    digitalWrite(wall_right, LOW);       //turn LED off
//motor control
    analogWrite(left_motor, 128);        //0(off) to 255 (full)
    analogWrite(right_motor, 128);       //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    analogWrite(left_motor, 0);           //turn motor off
    analogWrite(right_motor,0);           //turn motor off
  }
//robot action table row 5
  else if((left_IR_sensor_value > 512)&&(center_IR_sensor_value < 512)&&(right_IR_sensor_value
> 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, HIGH);       //turn LED on
    digitalWrite(wall_center, LOW);      //turn LED off
```

Эта книга из Telegram-канала

@IT_BUBBLEFORME



Читай бесплатно в Telegram книги по IT, программированию и ИИ

Сканируй QR или переходи по ссылке

https://t.me/IT_bubbleForMe

```
    digitalWrite(wall_right, HIGH);      //turn LED on
//motor control
    analogWrite(left_motor, 128);         //0(off) to 255 (full)
    analogWrite(right_motor, 128);        //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);  //turn LED off
    digitalWrite(right_turn_signal, LOW); //turn LED off
    analogWrite(left_motor, 0);           //turn motor off
    analogWrite(right_motor,0);           //turn motor off
  }
//robot action table row 6
  else if((left_IR_sensor_value > 512)&&(center_IR_sensor_value > 512)&&(right_IR_sensor_value
< 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, HIGH);        //turn LED on
    digitalWrite(wall_center, HIGH);      //turn LED on
    digitalWrite(wall_right, LOW);        //turn LED off
//motor control
    analogWrite(left_motor, 128);         //0(off) to 255 (full)
    analogWrite(right_motor, 0);          //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED on
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED OFF
    digitalWrite(right_turn_signal, LOW);  //turn LED OFF
    analogWrite(left_motor, 0);            //turn motor off
    analogWrite(right_motor,0);            //turn motor off
  }
//robot action table row 7
  else if((left_IR_sensor_value > 512)&&(center_IR_sensor_value > 512)&&(right_IR_sensor_value
> 512))
  {
//wall detection LEDs
    digitalWrite(wall_left, HIGH);        //turn LED on
    digitalWrite(wall_center, HIGH);      //turn LED on
```

```
    digitalWrite(wall_right, HIGH);       //turn LED on
//motor control
    analogWrite(left_motor, 128);         //0(off) to 255 (full)
    analogWrite(right_motor, 0);          //0(off) to 255 (full)
//turn signals
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED on
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, HIGH); //turn LED on
    delay(500); //delay 500 ms
    digitalWrite(left_turn_signal, LOW);   //turn LED off
    digitalWrite(right_turn_signal, LOW);  //turn LED off
    analogWrite(left_motor, 0);            //turn motor off
    analogWrite(right_motor,0);            //turn motor off
  }
}
//*********************************************************************
```

3.8.4. Тестирование алгоритма управления

Рекомендуется сначала протестировать алгоритм отдельно от платформы робота. Это можно сделать, подключив три ИК-датчика и светодиоды к соответствующим контактам на Arduino Nano 33, как показано на рис. 3.13. Вместо двух двигателей и их схем подключения можно использовать два светодиода с необходимой интерфейсной схемой. При различных сценариях тестирования светодиоды будут зажигаться, показывая, что включены соответствующие двигатели. После того как этот алгоритм будет полностью протестирован, Arduino Nano 33 можно будет установить на платформу робота и подключить к двигателям. Могут начаться полноценные испытания в лабиринте. Вперед!

3.9. Выводы

Мы начали эту главу с изучения требований к источнику питания для платы Arduino Nano 33 BLE Sense. Затем рассмотрели, как обеспечить питание из различных источников. В оставшейся части главы представлена информация о том, как подключить некоторые входные и мощные выходные устройства к процессору Nano.

3.10. Задания

1. Перечислите рекомендации по интерфейсу для Arduino Nano 33 BLE Sense.
2. Что произойдет, если микроконтроллер будет использоваться за пределами предписанного рабочего диапазона?
3. Каковы требования к питанию для Arduino Nano 33 BLE Sense? Описать варианты подачи питания на микроконтроллер.
4. Каковы роль и функция стабилизатора напряжения?
5. Создайте регулируемый батарейный блок питания для Arduino Nano 33 BLE Sense. Укажите все компоненты. Как долго разработанный вами блок будет питать Arduino Nano 33 BLE? Объясните[1].
6. Может ли светодиод с последовательным ограничительным резистором напрямую управляться микроконтроллером Nano 33 BLE Sense? Объясните.
7. Что такое дребезг контактов? Опишите два метода минимизации дребезга. Что произойдет, если дребезг переключателя или кнопки не устранен?
8. Какая функция в Arduino IDE обеспечивает широтно-импульсную модуляцию? Объясните все необходимые переменные для функции.

Источники

1. Arduino homepage, www.arduino.cc.
2. Boylestad, R. and L. Nashelsky, (1982). Electronic Devices and Circuit Theory, third edition, Prentice–Hall.

[1] Ответ в общем случае неопределенный (все зависит от задействованных модулей и в еще большей степени от подключенных внешних устройств), и тем более на этот вопрос невозможно ответить на основе только текста данной книги. Общий прием в таких случаях заключается в экспериментальном определении среднего потребления макета конкретного устройства с помощью анализатора мощности (такого как, например, Joulescope https://www.joulescope.com) и последующего расчета времени работы при известной емкости батарей или аккумуляторов. – *Прим. перев.*

3. Clark D. and M. Owings, (2003). Building Robot Drive Trains, McGraw Hill.
4. Fitzgerald, A., C. Kingsley, and S. Umans, (2003). Electric Machinery, sixth edition, McGraw Hill.
5. Sedra, A and K. Smith, (2004). Microelectronic Circuits, fifth edition, Oxford University Press.
6. TIP31C Power Transistors (NPN), ST Microelectronics, st.com, 2006.
7. TIP32C Power Transistors (PNP), ST Microelectronics, st.com, 2006.
8. TIP120, TIP121, TIP122 (NPN); TIP125, TIP126, TIP127 (PNP) plastic medium–power complementary silicon transistors (TIP120D), ON Semiconductor, onsemi.com, 2007.

ГЛАВА

4

Искусственный интеллект и машинное обучение

Прочитав эту главу, читатель должен уметь:

- описать взаимосвязь между концепциями искусственного интеллекта (artificial intelligence, AI), машинного обучения (machine learning, ML) и глубокого обучения (deep learning, DL);
- предоставить список примеров приложений искусственного интеллекта, машинного обучения и глубокого обучения;
- описать развитие ключевых событий в мире искусственного интеллекта;
- предоставить краткое изложение теории приложения метода K ближайших соседей (K nearest neighbor, KNN);
- спроектировать и реализовать приложение KNN на процессоре Arduino.
- предоставить краткое изложение теории, лежащей в основе построения и анализа дерева решений (decision tree);
- спроектировать и реализовать дерево решений на процессоре Arduino.

4.1. Обзор

На этом введение в Arduino IDE (глава 1), Nano 33 BLE Sense (глава 2) и методы подключения внешних устройств (глава 3) завершено. В оставшейся части книги мы сосредоточимся на концепциях и приложениях искусственного интеллекта (AI) и машинного обучения (ML) для систем на базе микроконтроллеров.

В недавнем выпуске команда Arduino заявила: «Миссия Arduino – сделать машинное обучение достаточно простым, чтобы его мог использовать каждый» (https://www.blog.arduino.cc). Те, кто знаком с концепциями искусственного интеллекта и машинного обучения, могут возразить, что эти концепции наиболее подходят для более мощных вычислительных платформ. Однако недавние разработки позволили запускать после обучения определенные приложения искусственного интеллекта на микроконтроллерах. Мы увидим, что задача обучения в некоторых приложениях также может быть решена на микроконтроллере. Кроме того, мы исследуем приложения, которые подходят для удаленных приложений искусственного интеллекта на базе микроконтроллеров с батарейным питанием [4]. В оставшейся части книги мы ограничиваем наши обсуждения методами искусственного интеллекта и машинного обучения специально для микроконтроллеров. Цель состоит в том, чтобы познакомить вас с концепциями и позволить вам попрактиковаться на недорогом и доступном оборудовании и программном обеспечении Arduino.

Рисунок 4.1 иллюстрирует взаимосвязь между искусственным интеллектом, машинным обучением и глубоким обучением. Цель искусственного интеллекта состоит в том, чтобы вычислительная техника имитировала разумное человеческое поведение. Некоторые относят возникновение ИИ к 1300 г. до н. э. [4]. Мы ограничиваем наш исторический обзор развитием ИИ в XX веке и далее. После краткого исторического обзора в этой главе исследуется концепция K ближайших соседей (K nearest neighbor, KNN) и методы классификации дерева решений (decision tree).

В главе 5 мы будем исследовать также нечеткую логику. Общая цель нечеткой логики – управлять системой с помощью серии задания правил в форме «if – then» («если – то», «условие – следствие»). Входные априорные условия оператора `if` формируются через получение точной и четкой входной информации от входных датчиков и преобразователей и сопоставления ее с нечетки-

ми входными лингвистическими (словесными) переменными. Это называется этапом введения нечеткости. Часть правил оператора then (последствий) – это команды управления, возвращающиеся в систему. Опять же, лингвистические переменные используются для описания управляющих выходных данных. Нечеткий управляющий выход преобразуется для получения точных и четких выходных управляющих сигналов. Сопоставление входов и выходов с помощью операторов «if – then» обеспечивается разработчиком системы.

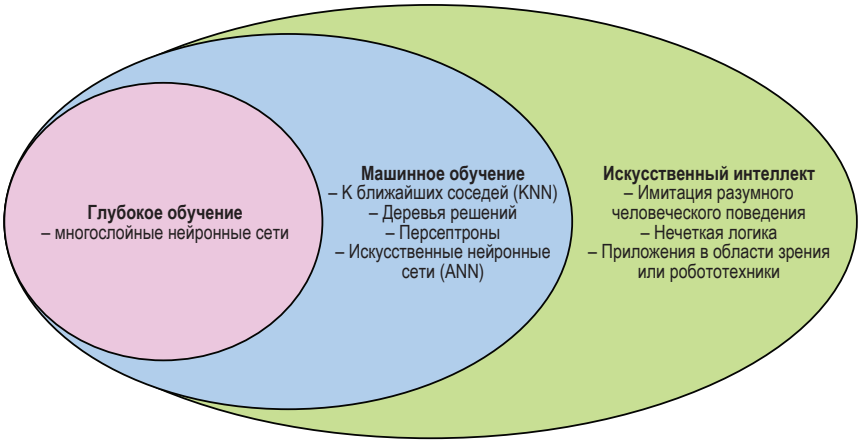


Рис. 4. 1. Искусственный интеллект и машинное обучение [3]

Машинное обучение – это категория, входящая в состав широкого круга искусственного интеллекта. Его цель – разработать алгоритмы управления процессом. Разработанный алгоритм проходит этап обучения, на котором входные данные используются для подтверждения или определения желаемых выходных данных контроллера. В процессе обучения алгоритма определенные веса и смещения корректируются для повышения производительности алгоритма. В области ML мы исследуем деревья решений, алгоритмы К ближайших соседей (KNN), персептроны, искусственные нейроны и искусственные нейронные сети (artificial neural networks, ANN).

В этой главе мы исследуем деревья решений и приложения KNN, остальные темы – в главах 5 и 6. Глубокое обучение предполагает разработку алгоритмов с использованием многослойных искусственных нейронных сетей (multi-layer ANN). Понятия, представ-

ленные в главе 6, позволят читателю разрабатывать подобные сети глубокого обучения. Завершает главу 6 краткое введение в передовые инструменты и приложения искусственного интеллекта и машинного обучения.

4.2. Краткая история развития искусственного интеллекта и машинного обучения

На рис. 4.2 представлено краткое изложение ключевых разработок в области искусственного интеллекта и машинного обучения. Следует признать, что многие работы там не уместились[1].

Мы начнем с работы Джорджа Буля (George Boole) по развитию булевой алгебры, которую он считал «фундаментальными законами операций разума». В конце 1930-х годов в диссертации Клода Шеннона (Claude Shannon) было описано, как использовать булеву алгебру для оптимизации релейных схем в телефонной связи. Это привело к повсеместному использованию булевой алгебры при разработке первых компьютеров.

Несколько лет спустя, в разгар Второй мировой войны, Маккалок (Warren McCulloch) и Питтс (Walter Pitts) разработали первую математическую модель нейрона[2]. В 1949 году Дональд Хебб (Donald Hebb) описал, как синапсы между нейронами укрепляются при многократном использовании. В 1956 году Дартмутский колледж провел первую конференцию по искусственному интеллекту. Конференция собрала ведущих ученых в области искусственного интеллекта для дискуссий, продолжавшихся два месяца. Год спустя Фрэнк Розенблатт (Frank Rosenblatt) разработал модель и реали-

---

[1] Рекомендуем ознакомиться с книгой Клиффорда Пиковера [4], где вы найдете подробное и увлекательное рассмотрение этой темы, начиная с 1300 года до нашей эры. – *Прим. авт.*

[2] Для лучшего понимания содержания и значения работ этих ученых, а также атмосферы того времени рекомендуется статья «О том, как гениальный беспризорник и профессор придумывали первую модель искусственного нейрона» (https://habr.com/ru/companies/sberdevices/articles/525508). Кроме того, автор упускает важнейшее концептуальное направление в развитии AI, связанное с именем Алана Тьюринга (https://habr.com/ru/companies/ua-hosting/articles/471308). – *Прим. перев.*

зацию одиночного персептрона. Мы рассмотрим эту модель более подробно в этой книге далее.

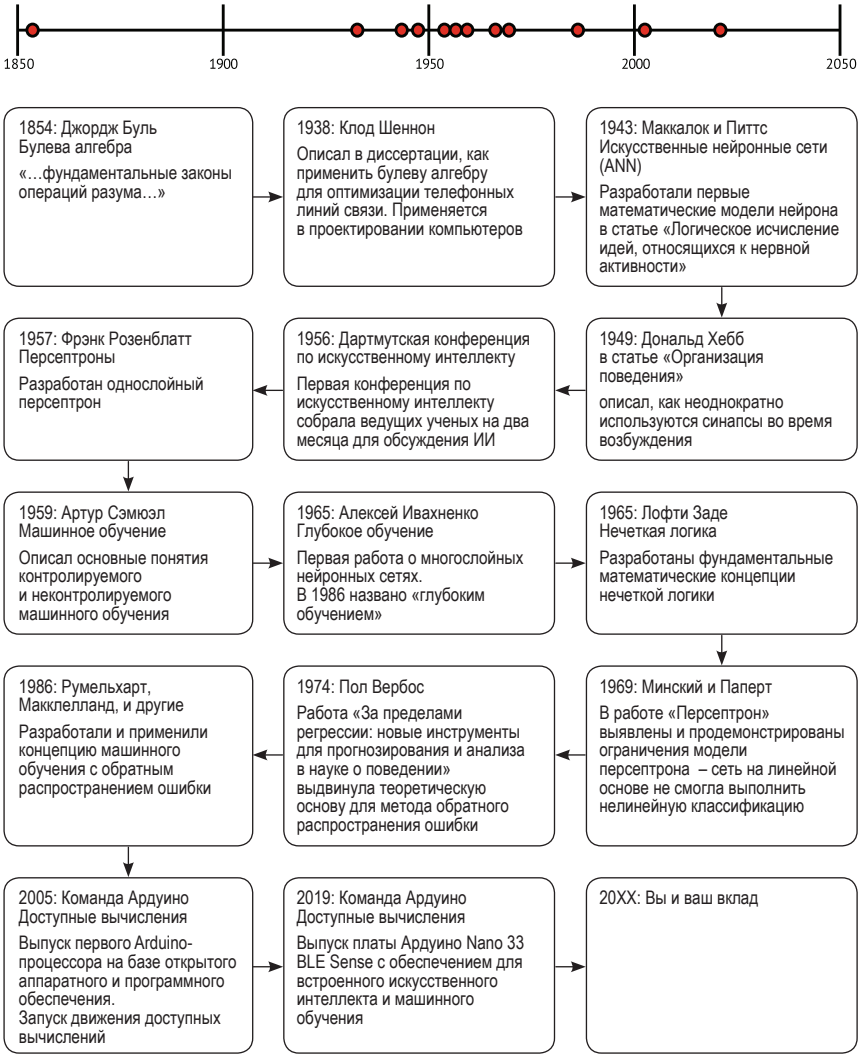


Рис. 4.2. Краткий график развития ИИ и МО [4, 9]

В 1959 году Артур Сэмюэл (Arthur Samuel) разработал основные концепции машинного обучения, включая контролируемые

и неконтролируемые методы обучения (методы с «учителем» и без «учителя»). В середине 1960-х годов Алексей Ивахненко (Alexey Ivakhnenko) провел пионерскую работу в области многослойных нейронных сетей, которая примерно два десятилетия спустя получила название «глубокое обучение» (Deep Learning, DL). Примерно в то же время Лофти Заде (Lofti Zadeh) разработал фундаментальные математические концепции нечеткой логики. Более подробно нечеткую логику мы рассмотрим далее в этой книге.

В 1969 году Мински (Marvin Minsky) и Паперт (Seymour Papert) выявили и продемонстрировали ограничения модели персептрона при решении задач нелинейной классификации. Это стимулировало дополнительную работу по созданию нейронных сетей, способных строить модели нелинейной классификации. Пять лет спустя Пол Вербос (Paul Werbos) в своей докторской диссертации представил фундаментальную теоретическую работу по обратному распространению ошибки (backpropagation) при обучении нейронной сети. В 1986 году Румельхарт (David Rumelhart), Макклелланд (James L. McClelland) и другие исследователи разработали и применили концепции машинного обучения с обратным распространением ошибки.

Теперь мы перенесемся в 2005 год. В это время команда Arduino выпустила первый процессор Arduino, основанный на концепции аппаратного и программного обеспечения с открытым исходным кодом, который положил начало глобальному движению за доступные вычисления. Следуя той же философии, в недавнем выпуске команда Arduino заявила: «Миссия Arduino – сделать машинное обучение достаточно простым, чтобы его мог использовать каждый» (https://www.blog.arduino.cc). В 2019 году команда Arduino выпускает процессор Nano 33 BLE Sense, обеспечивающий доступность машинного обучения и искусственного интеллекта.

4.3. Метод К ближайших соседей

Общая цель метода К ближайших соседей[1] (K Nearest Neighbours, KNN) – классифицировать новый объект путем сравнения его ха-

[1] Подробное рассмотрение принципов метода ближайших соседей, а также дерева решений (см. следующий раздел) на русском языке см. в выдержке из курса лекций по машинному обучению по адресу: https://habr.com/ru/companies/ods/articles/322534 (примеры там ориентированы на Python-библиотеки применительно к другим контроллерам, но принцип изложен очень доходчиво). – *Прим. перев.*

рактеристик с заданным набором данных. Метод основан на предположении, что объекты со схожими характеристиками и общей классификацией будут группироваться вместе при отображении в пространстве признаков. Этот метод классификации объектов восходит к работе Фикса (E. Fix) и Ходжеса (J. L. Hodges) 1951 года [6]. Метод KNN считается методом контролируемой классификации (методом с «учителем»). То есть вначале алгоритму KNN предоставляется обучающий набор объектов с известной классификацией, после чего его можно будет использовать для классификации объектов за пределами обучающего набора.

Например, на рис. 4.3 изображен классификатор цветов. Алгоритм обучался с использованием набора из десяти точек данных

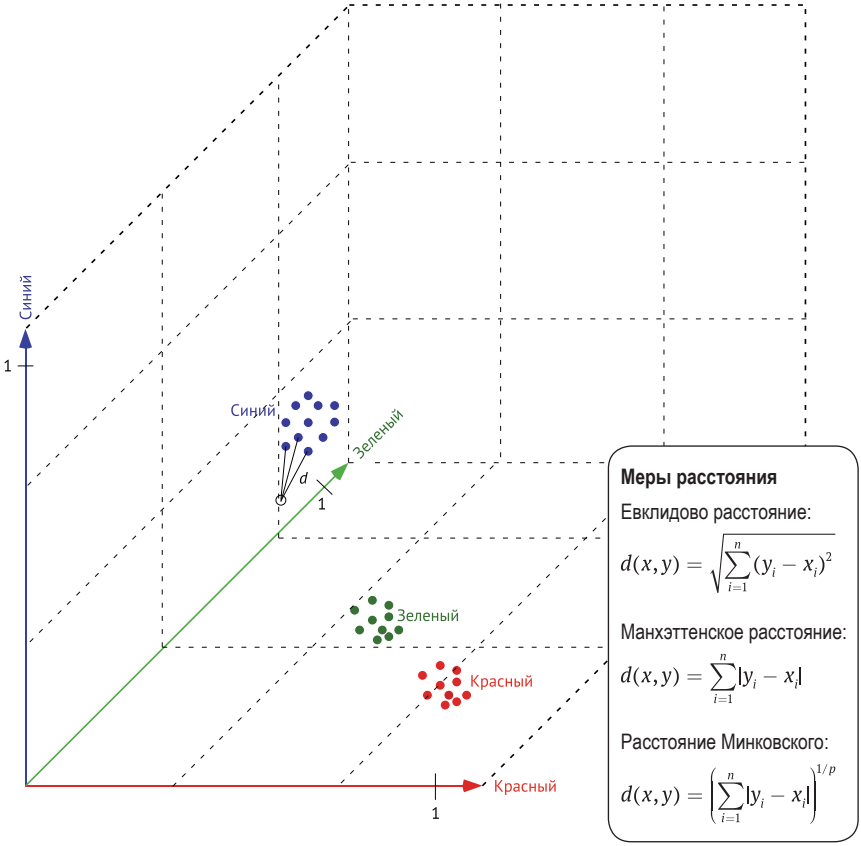


Рис. 4.3. Классификатор K Nearest Neighbor (KNN)

для каждой желаемой известной классификации (цвета). В этом примере компоненты красного, зеленого и синего цветов (RGB) каждого образца сохранялись вместе с соответствующей классификацией (тегом).

После обучения RGB-компоненты нового неклассифицированного объекта передаются алгоритму для его классификации. Компоненты RGB сравниваются с ближайшими K соседними компонентами RGB, уже имеющимися в наборе размеченных данных. На рисунке алгоритм использует значение K, равное трем, для определения расстояния до трех ближайших соседей. Показаны три распространенные меры расстояния, включая евклидово расстояние, манхэттенское расстояние и меру расстояния Минковского.

Ниже представлен скетч Arduino для классификатора цветов на основе KNN. Он адаптирован из библиотеки «ColorClassifer» Arduino KNN. UML-диаграмма действий для скетча представлена на рис. 4.4. Классификатор цветов использует датчик ADPS-9960 на плате Arduino Nano 33 BLE Sense.

Чтобы протестировать скетч, загрузите библиотеку Arduino KNN на вкладке **Tools** (*Инструменты*) -> **Manage libraries** (*Управление библиотеками*) в Arduino IDE. Образцы цветов красного, зеленого и синего можно сделать из цветной плотной бумаги. Скомпилируйте и загрузите скетч в Arduino Nano 33 BLE Sense. Следуйте пользовательским подсказкам в мониторе последовательного порта для обучения, а затем используйте классификатор цвета.

```
//*********************************************************************
//KNN color classification
//
//This sketch classifies objects using a color sensor.
//
//First you 'teach' the Arduino by putting an example of each object
//close to the color sensor.
//
//After the training step, the Arduino will guess the name of objects
//it is shown based on how similar the color is to the examples.
//
//The "k" is adjustable and set to 5.
//
//Processor: Arduino Nano 33 BLE Sense
//
//When compiled and uploaded, open the serial console.
//- Follow prompts to provide sample objects.
//- Once trained, user is prompted for objects.
//- Algorithm will guess the color of the object provided.
//
```

```
//Sketch adapted from ColorClassifer from Arduino KNN Library
//
//This example code is in the public domain.
//*****************************************************************
#include <Arduino_KNN.h>
#include <Arduino_APDS9960.h>
const int INPUTS = 3; //Number of input values passed to classifier.
                      //Input is color sensor R, G, B level data
```

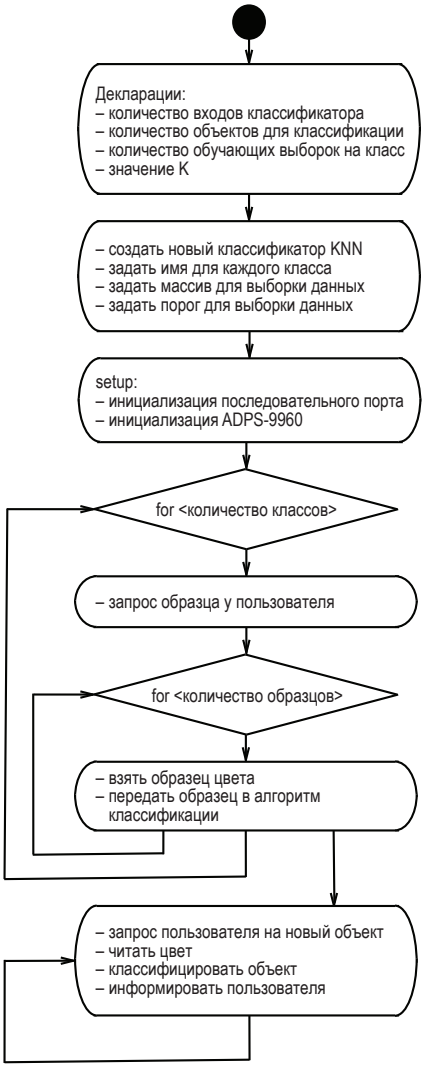


Рис. 4.4. Классификатор цветов KNN

```
const int CLASSES = 3; //Number of objects for classification
                       //(e.g. Red, Blue, Green)
const int EXAMPLES_PER_CLASS = 10; //Number of data samples per object for training
const int K = 5;     //number of neighbors considered
float color[INPUTS]; //Data storage array
                     //Used to pass data to KNN alogorithm
const int THRESHOLD = 5; //Threshold for color brightness
KNNClassifier myKNN(INPUTS);        //Create a new KNNClassifier
//Names for each class (object type)
String label[CLASSES] = {"Red", "Blue", "Green"};
void setup()
{
  Serial.begin(9600);
  while (!Serial);
  if(!APDS.begin())                 //Ensure ADPS is present
  {
    Serial.println("Failed to initialize APDS!");
    while (1);
  }
  Serial.println("Arduino k-NN color classifier");
//Prompt user for the name of each object
  for(int currentClass = 0; currentClass < CLASSES; currentClass++)
  {
//Prompt user for samples of object
    for(int currentExample = 0; currentExample < EXAMPLES_PER_CLASS; currentExample++)
    {
      Serial.print("Show me an example ");
      Serial.println(label[currentClass]);
      readColor(color);    //Wait for an object then read its color
      myKNN.addExample(color, currentClass); //Add example color to the k-NN model
    }
//Wait for user to move object away
    while(!APDS.proximityAvailable()||APDS.readProximity()==0){}
  }
}
void loop()
{
  int classification;
//Wait for the object to move away again
  while (!APDS.proximityAvailable() || APDS.readProximity()==0){}
  Serial.println("Let me guess your object");
  readColor(color); //Wait for an object then read its color
  classification = myKNN.classify(color, K);  //Classify the object
  Serial.print("You showed me: ");    //Print the classification
  Serial.println(label[classification]);
}
//*********************************************************************
void readColor(float color[])
{
  int red, green, blue, proximity, colorTotal = 0;
//Wait for the object to move close
```

```
  while (!APDS.proximityAvailable() || APDS.readProximity() > 0){}
//Wait until color is bright enough
  while (colorTotal < THRESHOLD)
  {
//Sample if color is available and object is close
    if(APDS.colorAvailable())
    {
      APDS.readColor(red, green, blue);  // Read color and proximity
      colorTotal = (red + green + blue);
    }
  }
//Normalise the color sample data and put it in the classifier input array
  color[0] = (float)red / colorTotal;
  color[1] = (float)green/ colorTotal;
  color[2] = (float)blue / colorTotal;
//Print the red, green and blue percentage values
  Serial.print(color[0]); Serial.print(",");
  Serial.print(color[1]); Serial.print(",");
  Serial.println(color[2]);
}
//*****************************************************************
```

4.4. Дерево решений

Дерево решений – это мощный графический метод для иллюстрации сложного процесса принятия решений. Дерево решений строится с использованием контролируемого обучения (метода с «учителем»). То есть набор наблюдений или объектов с готовыми результатами классификации используется для последовательного построения дерева решений[1].

Набор данных состоит из ряда атрибутов или свойств (признаков), описывающих объект для классификации. Каждый атрибут имеет ряд описывающих его значений. Атрибуты и их значения сопровождаются наблюдаемым результатом классификации.

Набор данных для построения дерева решений может быть извлечен из существующей базы данных или подготовлен экспертом в данной области. После того как дерево решений построено, его можно использовать для представления решения задачи классификации с учетом новых наблюдений за пределами исходного набора данных.

---

[1] Дополнительную информацию для этого раздела см. [5, 8]. – *Прим. авт.* См. также предыдущую сноску на стр. 129. – *Прим. перев.*

Дж. Р. Куинлан (J.R. Quinlan) в классической работе «Induction of Decision Trees» [5] приводит пример определения того, играть или нет в теннис в конкретный день, на основе ряда погодных признаков и предыдущих решений, принятых, как показано на рис. 4.5.

Наблюдение	Атрибут (признак, свойство)				Выход/решение Играть в теннис?
	Прогноз	Температура	Влажность	Ветрено?	
1	солнечно	жарко	высокая	F	N
2	солнечно	жарко	высокая	T	N
3	пасмурно	жарко	высокая	F	Y
4	дождь	умеренно	высокая	F	Y
5	дождь	прохладно	нормальная	F	Y
6	дождь	прохладно	нормальная	T	N
7	пасмурно	прохладно	нормальная	T	Y
8	солнечно	умеренно	высокая	F	N
9	солнечно	прохладно	нормальная	F	Y
10	дождь	умеренно	нормальная	F	Y
11	солнечно	умеренно	нормальная	T	Y
12	пасмурно	умеренно	высокая	T	Y
13	пасмурно	жарко	нормальная	F	Y
14	дождь	умеренно	высокая	T	N

Атрибут: значение 1, значение 2, значение 3.
Прогноз: солнечно (sunny), пасмурно (overcast), дождь (rain).
Температура: жарко (hot), умеренно (mild), прохладно (cool).
Влажность: высокая (high), нормальная (normal).
Ветрено: правда (T), ложь (F).

Рис. 4.5. Набор данных для игры в теннис [5]

К признакам (атрибутам), описывающим день, относятся: прогноз, температура, влажность и наличие ветра. Значения каждого признака представлены в таблице на рис. 4.5. Набор данных состоит из 14 наблюдений, сделанных в разные субботы. Для каждого наблюдения записывалось наблюдаемое значение каждого признака, а также отмечалось, играли ли в этот день в теннис. Цель состоит в том, чтобы разработать дерево решений для обработки значений признаков в другой день, не присутствующий в наборе данных, дабы определить, будут играть в теннис или нет.

Этот пример может показаться тривиальным. Вы можете возразить: «Мне не нужно дерево решений, чтобы определить, играть в теннис или нет». Однако этот пример служит доступным шаблоном для изучения и реализации основ дерева решений. Деревья решений могут использоваться для самых разных приложений, включая прогнозирование погоды, медицинскую диагностику и управление роботами.

Чтобы начать построение дерева решений, собирается статистика набора данных. Эти статистические данные могут быть собраны вручную или определены с применением алгоритма дерева решений. В данном примере мы используем смесь обоих подходов, а также дополнение массивов для хранения собранной информации. Как показано на рис. 4.6, для реализации дерева решений используются следующие массивы:

- `dt_array`: содержит набор данных наблюдений;
- `attrib_array`: содержит признаки и связанные значения;
- `attr_cnt_array`: счетчик для каждого признака, приводящего к игре в теннис («Y») или к отказу от игры («N»);

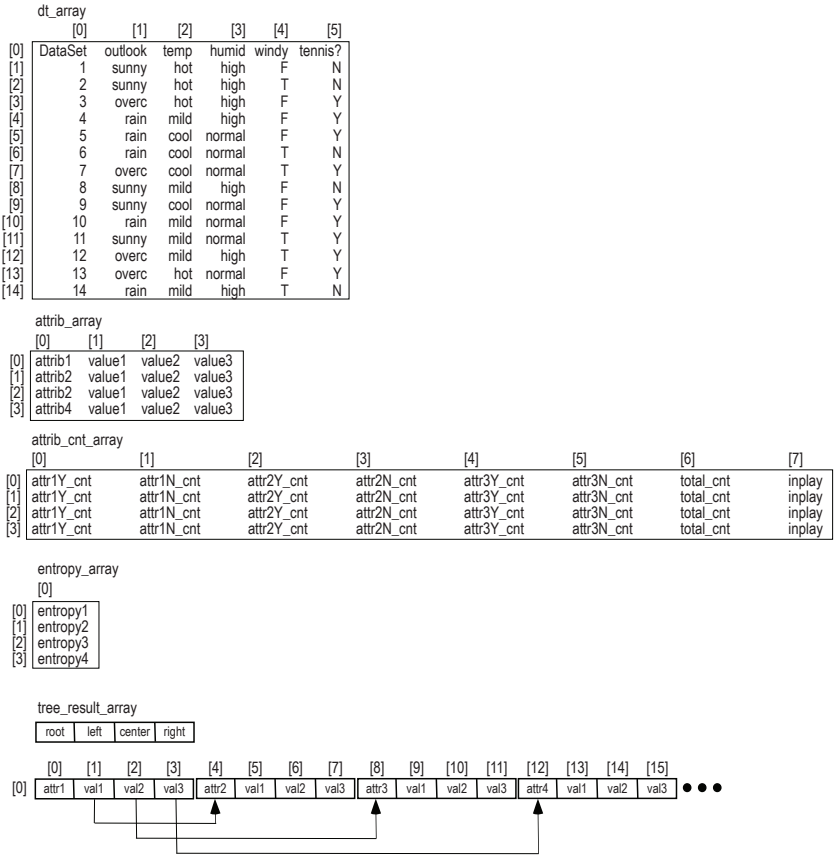


Рис. 4.6. Набор массивов, используемый для построения дерева решений

- `entropy_array`: содержит расчет энтропии для каждого признака;
- `tree_result_array`: содержит результирующее дерево решений.

Важно отметить, что для построения дерева решений можно использовать множество различных структур данных. Примеры включают структуры, бинарные деревья и рекурсивные алгоритмы. Мы используем одномерный массив (`tree_result_array`) для хранения дерева решений. Массив, хотя и одномерный, может использоваться для хранения и построения двумерного дерева, как показано на рис. 4.7.

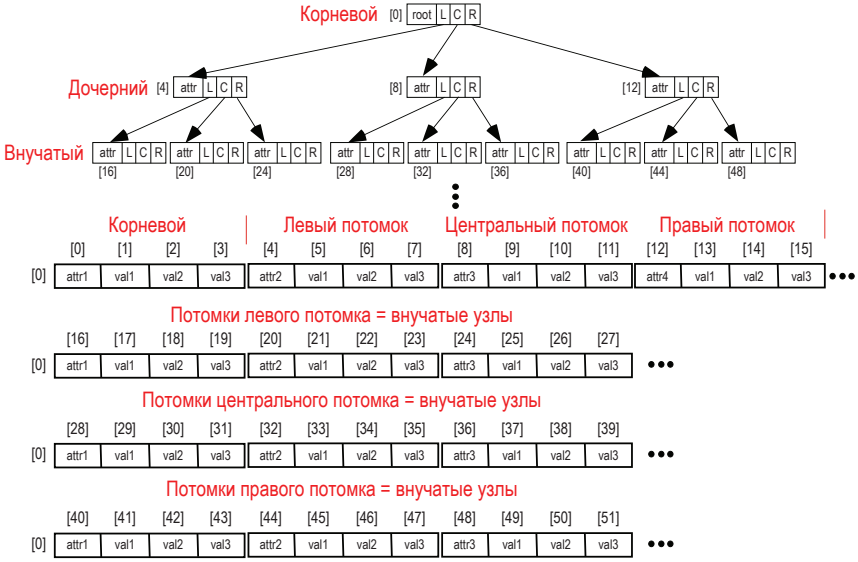


Рис. 4.7. Двумерный массив, используемый для формирования дерева решений

Основание дерева называется корневым узлом. Каждый узел содержит имя признака и три сопутствующих поля значений (слева, по центру и справа). Каждое из полей имеет фиксированную связь со своими дочерними узлами. Поля внутри дочерних узлов также имеют фиксированную связь с дочерними узлами. Как показано на рисунке, структура дерева фиксирована, а отношения между корневыми узлами, дочерними узлами и узлами, следующими по иерархии, известны и фиксированы.

UML-диаграмма действий (блок-схема) для разработки алгоритма дерева решений представлена на рис. 4.8.

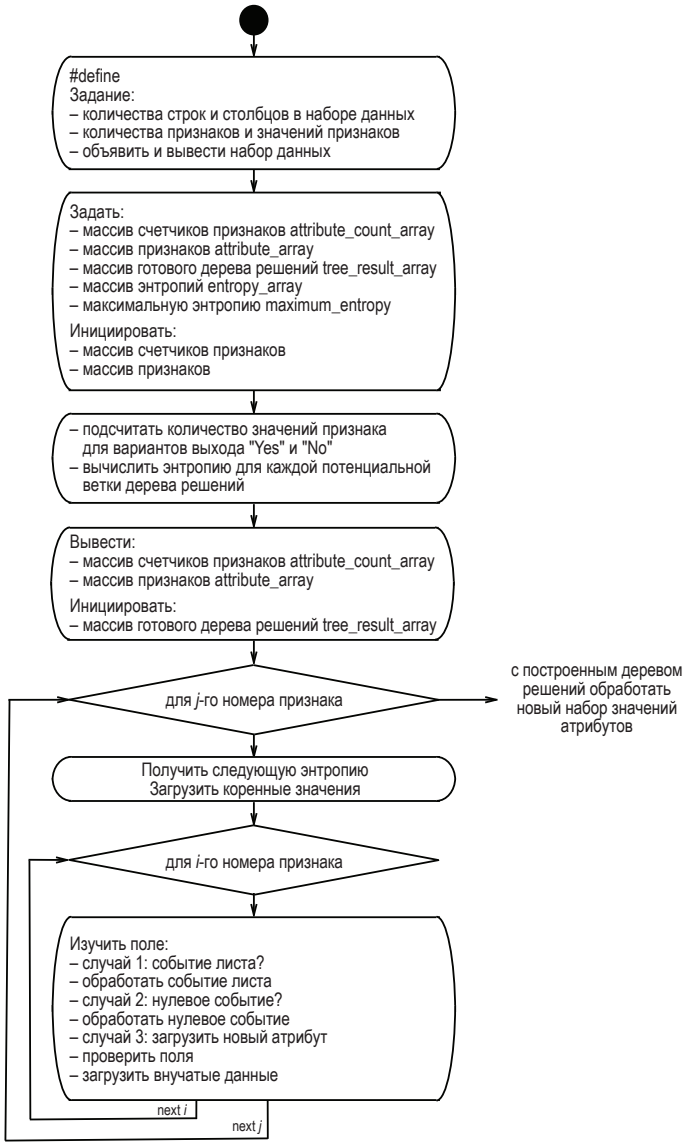


Рис. 4.8. UML-диаграмма действий построения дерева решений

Дерево решений собирается поэтапно, начиная с корневого узла. Корневой узел – это атрибут, имеющий наибольшее значение меры беспорядка – энтропии. Расчет энтропии для данного атрибута показан на рис. 4.9. Поля корневого узла (левое, центральное и правое) содержат имена значений, связанных с корневым атрибутом. Подсчет для каждого значения атрибута, приводящего к игре или отказу от игры, предоставляется в массиве `attribute_count_array`.

Value 1	Value 2	1st term	2nd term	entropy	entropy total
2	3	0.4	0.6	0.97	
4	0	1	0	0.00	
3	2	0.6	0.4	0.97	0.694
2	2	0.50	0.50	1.00	
4	2	0.67	0.33	0.92	
3	1	0.75	0.25	0.81	0.911
3	4	0.43	0.57	0.99	
6	1	0.86	0.14	0.59	0.788
6	2	0.75	0.25	0.81	
3	3	0.50	0.50	1.00	0.892

Entropy(value1, value2) = - value1/(value1+value2)*log2(value1/(value1+value2))
- value2/(value1+value2)*log2(value2/(value1+value2));

1st_term = value 1/(value 1 + value 2);

2nd_term = value 2/(value 1 + value 2);

Entropy (value 1, value 2) = - (1st_term * log10(1st_term)/log10(2))
- (2nd_term * log10(2nd_term)/log10(2)).

Рис. 4.9. Расчет энтропии

После загрузки корневого узла в массив результатов (`tree_result_array`) проверяются левое, центральное и правое поля, чтобы установить содержимое дочерних узлов. Дочерние узлы могут содержать следующий доступный атрибут с меньшей энтропией или значениями признаков каждого листа[1]. Затем поля дочерних узлов проверяются, чтобы установить содержимое внучатых

[1] Понятие листа (leaf – прямоугольник в дереве решений, содержащий набор признаков и расчетных величин для одного случая) иллюстрируется рис. 4.10. – *Прим. перев.*

узлов. Этот процесс продолжается до тех пор, пока не останется признаков для формирования дерева. При построении дерева решений каждый узел выполняет этап принятия решения, чтобы разбить данные на более мелкие наборы и уменьшить энтропию в наборе данных.

Результирующее дерево решений для игры в теннис представлено на рис. 4.10. Обратите внимание, что признак температуры не отображается в окончательном дереве решений. Алгоритм исключил признаки с энтропией более 0,90. Проверка атрибута температуры в массиве `array_count_array` показывает, что эта функция не влияет на решение играть в теннис.

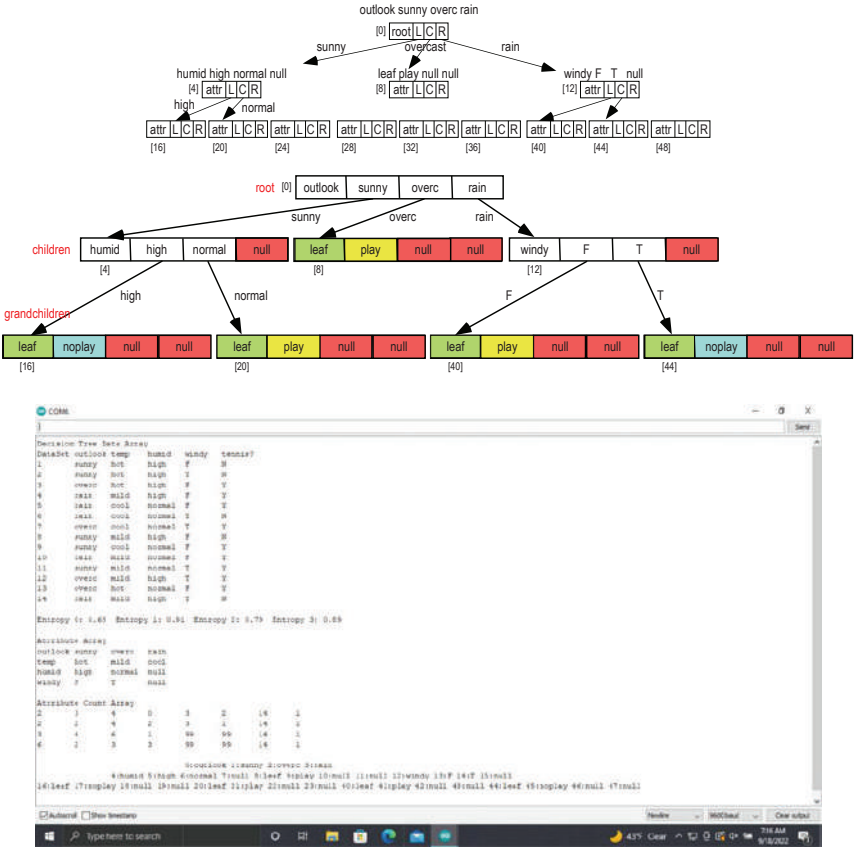


Рис. 4.10. Дерево решений для игры в теннис

Ниже приведен скетч Arduino для реализации дерева решений для игры в теннис. Он был реализован с помощью Arduino Nano 33 BLE Sense.

Примечание **Основной цикл `loop()` из кода ниже приведен далее в разделе 4.6.**

```
//*********************************************************************
//decision_tree
//*********************************************************************
#define num_rows 15
#define num_cols 6
#define max_num_attrib_vals 4
#define num_attrib 4
unsigned int first_attrib = 1, last_attrib = (num_cols - 1);
unsigned int child_incr, gr_child_incr;
const char* dt_array[num_rows][num_cols] =
{{"DataSet", "outlook", "temp", "humid", "windy", "tennis?"},
{ "1", "sunny", "hot", "high", "F", "N"},
{ "2", "sunny", "hot", "high", "T", "N"},
{ "3", "overc", "hot", "high", "F", "Y"},
{ "4", "rain", "mild", "high", "F", "Y"},
{ "5", "rain", "cool", "normal", "F", "Y"},
{ "6", "rain", "cool", "normal", "T", "N"},
{ "7", "overc", "cool", "normal", "T", "Y"},
{ "8", "sunny", "mild", "high", "F", "N"},
{ "9", "sunny", "cool", "normal", "F", "Y"},
{ "10", "rain", "mild", "normal", "F", "Y"},
{ "11", "sunny", "mild", "normal", "T", "Y"},
{ "12", "overc", "mild", "high", "T", "Y"},
{ "13", "overc", "hot", "normal", "F", "Y"},
{ "14", "rain", "mild", "high", "T", "N"}};
unsigned int attr_cnt_array[num_attrib][(max_num_attrib_vals * 2)];
const char* attrib_array[num_attrib][max_num_attrib_vals];
const char* tree_result_array[num_attrib * 4 * 4];
double entropy_array[num_attrib][1];
double entropy_total, entropy_val;
double entropy_max = 0.90;
void setup()
{
  Serial.begin(9600);
  while(!Serial);
  print_DT_array();
  reset_attrib_array();
  reset_attr_cnt_array();
  num_values_per_attrib();
  print_attrib_array();
  print_attr_cnt_array();
  reset_tree_result_array();
  build_decision_tree();
```

```
  print_tree_result_array();
}
void loop()
{
//navigate decision tree with new values of attribute values
}
//*********************************************************************
void print_DT_array(void)
{
  unsigned int i,j;
  while(!Serial);
  Serial.println("Decision Tree Data Array");
  for(i=0; i< num_rows; i++)              //row selector
  {
    for(j=0; j< num_cols; j++)            //column selector
    {
      Serial.print(dt_array[i][j]);
      Serial.print("\t");
    }
    Serial.println("");
  }
  Serial.println("");
}
//*********************************************************************
void num_values_per_attrib(void)
{
  const char* value1;
  const char* value2;
  const char* value3;
  unsigned int val2_set, val3_set;
  unsigned int attr1Y_cnt, attr2Y_cnt, attr3Y_cnt;
  unsigned int attr1N_cnt, attr2N_cnt, attr3N_cnt;
  unsigned int total_attr_cnt;
  unsigned int i, j, p;
  total_attr_cnt = 0;
  for(j=first_attrib; j< last_attrib; j++) //col selector
  {                                       //set attr
    value1 = "null"; value2 = "null"; value3 = "null";
    value1 = dt_array[1][j];              //set 1st attr
    val2_set = 0; val3_set = 0;           //reset flags
    total_attr_cnt = 0;
    for(i=2; i< num_rows; i++)
    {                                     //set 2nd attr
      if((value1 != dt_array [i][j])&&(val2_set == 0))
      {
        value2 = dt_array [i][j];
        val2_set = 1;
      }                                   //set 3rd attr
      else if ((value1!=dt_array [i][j])&&(value2!=dt_array [i][j])&&(val3_set == 0))
      {
        value3 = dt_array [i][j];
```

```
        val3_set = 1;
      }
      else
      {
        ;
      }
    }
//store attribute values to attrib_array
    attrib_array[j-1][0] = dt_array[0][j];
    attrib_array[j-1][1] = value1;
    attrib_array[j-1][2] = value2;
    attrib_array[j-1][3] = value3;
    attr1Y_cnt = 0; attr2Y_cnt = 0; attr3Y_cnt = 0; //reset attr cnt
    attr1N_cnt = 0; attr2N_cnt = 0; attr3N_cnt = 0; //reset attr cnt
    for(i=1; i< num_rows; i++)
    {
//count number of attrb in each category
      if((value1 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "Y"))
      {attr1Y_cnt++;}
      else if ((value1 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "N"))
      {attr1N_cnt++;}
      else if((value2 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "Y"))
      {attr2Y_cnt++;}
      else if ((value2 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "N"))
      {attr2N_cnt++;}
      else if((value3 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "Y"))
      {attr3Y_cnt++;}
      else if ((value3 == dt_array[i][j])&&(dt_array[i][num_cols - 1] == "N"))
      {attr3N_cnt++;}
      else
      {
        ;
      }
    }
//end for i
//store results to attr_cnt_array
    if((attr1Y_cnt + attr1N_cnt) != 0)
    {
      attr_cnt_array[j-1][0] = attr1Y_cnt;
      attr_cnt_array[j-1][1] = attr1N_cnt;
      total_attr_cnt = total_attr_cnt + attr1Y_cnt + attr1N_cnt;
    }
    else
    {
      attr_cnt_array[j-1][0] = 99;
      attr_cnt_array[j-1][1] = 99;
    }
    if((attr2Y_cnt + attr2N_cnt) != 0)
    {
      attr_cnt_array[j-1][2] = attr2Y_cnt;
      attr_cnt_array[j-1][3] = attr2N_cnt;
```

```
      total_attr_cnt = total_attr_cnt + attr2Y_cnt + attr2N_cnt;
    }
    else
    {
      attr_cnt_array[j-1][2] = 99;
      attr_cnt_array[j-1][3] = 99;
    }
    if((attr3Y_cnt + attr3N_cnt) != 0)
    {
      attr_cnt_array[j-1][4] = attr3Y_cnt;
      attr_cnt_array[j-1][5] = attr3N_cnt;
      total_attr_cnt = total_attr_cnt + attr3Y_cnt + attr3N_cnt;
    }
    else
    {
      attr_cnt_array[j-1][4] = 99;
      attr_cnt_array[j-1][5] = 99;
    }
    attr_cnt_array[j-1][6] = attr_cnt_array[j-1][6] + total_attr_cnt;
    attr_cnt_array[j-1][7] = 1; //indicates attribute still in play
//calculate entropy of attribute and attribute values
    entropy_total = 0;
    for(p = 0; p < ((max_num_attrib_vals-1) * 2); p=p+2)
    {
      if(attr_cnt_array[j-1][p] != 99)
      {
        entropy_val = calculate_entropy(attr_cnt_array[j-1][p],
        attr_cnt_array[j-1][p+1]);
        entropy_val = (((double)((attr_cnt_array[j-1][p])) + (double)(attr_cnt_array[j-1]
[p+1]))/ ((double)((attr_cnt_array[j-1][6])))) * entropy_val;
        entropy_total = entropy_total + entropy_val;
      }
    }
     entropy_array[j-1][0] = entropy_total;
  }                                        //end for j
  print_entropy_array();
  delay(100);
}
//*********************************************************************
void print_attrib_array(void)
{
  unsigned int k, m;
  while(!Serial);
  Serial.println("Atrribute Array");
  for(k=0; k < num_attrib; k++)              //row selector
  {
    for(m=0; m < max_num_attrib_vals; m++)   //column selector
    {
      Serial.print(attrib_array[k][m]);
      delay(10);
      Serial.print("\t");
```

```
      delay(10);
    }
    Serial.println("");
  }
  Serial.println("");
}
//*********************************************************************
void reset_attrib_array(void)
{
  unsigned int k, m;
  for(k=0; k < num_attrib; k++)                    //row selector
  {
    for(m=0; m < max_num_attrib_vals; m++)         //column selector
    {
      attrib_array[k][m] = "null";
    }
  }
}
//*********************************************************************
void print_attr_cnt_array(void)
{
  unsigned int k,m;
  Serial.println("Atrribute Count Array");
  for(k=0; k < (num_attrib); k++)                  //row selector
  {
    for(m=0; m < (max_num_attrib_vals * 2); m++)   //col selector
    {
      Serial.print(attr_cnt_array[k][m]);
      Serial.print("\t");
    }
    Serial.println("");
  }
  Serial.println("");
}
//*********************************************************************
void reset_attr_cnt_array(void)
{
  unsigned int k,m;
  for(k=0; k < (num_attrib); k++)                  //row selector
  {
    for(m=0; m < (max_num_attrib_vals * 2); m++)   //col selector
    {
      attr_cnt_array[k][m] = 0;
    }
  }
}
//*********************************************************************
double calculate_entropy(unsigned int first_val, unsigned int second_val)
{
  double entropy_val;
  double first_val_dbl, second_val_dbl;
```

```
  double first_val_dbl_sub, second_val_dbl_sub;
  first_val_dbl = (double)(first_val);
  second_val_dbl = (double)(second_val);
//prevents NaN in log calculation
  if(first_val_dbl < 1.0)
    first_val_dbl_sub = 0.005;
  else
    first_val_dbl_sub = first_val_dbl;
  if(second_val_dbl < 1.0)
    second_val_dbl_sub = 0.005;
  else
    second_val_dbl_sub = second_val_dbl;
  entropy_val = - ((first_val_dbl/(first_val_dbl + second_val_dbl)) * (log10((first_val_
dbl_sub/(first_val_dbl_sub+second_val_dbl_sub)))/log10(2))- ((second_val_dbl/(first_val_
dbl+second_val_dbl))* (log10((second_val_dbl_sub/(first_val_dbl_sub+second_val_dbl_sub)))/
log10(2));
    return entropy_val;
}
//*********************************************************************
void print_entropy_array(void)
{
  unsigned int k;
  for(k=0; k < (num_attrib); k++)     //row selector
  {
    Serial.print("Entropy ");
    Serial.print(k);
    Serial.print(": ");
    Serial.print(entropy_array[k][0]);
    Serial.print(" ");
  }
  Serial.println(" ");
  Serial.println(" ");
}
//*********************************************************************
//void build_decision_tree(void)
//*********************************************************************
void build_decision_tree(void)
{
  unsigned int i,j,k;
  unsigned int attr_found;            //provides row# for attr array
  unsigned int new_attr_found;        //provides row# for attr array
  unsigned int array_counter = 0;     //position in tree_result_array
  for(j=0; j < num_attrib; j++)       //find entropy values in order
  {                                   //lowest to highest
    attr_found = get_next_entropy();  //provides row# attrib_array
    if((attr_found!=99)&&(entropy_array[attr_found][0]<= entropy_max))
                                      //reports 99 if no new value
    {                                 //load attrib to tree_result_array
    tree_result_array[array_counter]=attrib_array[attr_found][0];    //attr name
    tree_result_array[array_counter+1]=attrib_array[attr_found][1];  //attr val1
    tree_result_array[array_counter+2]=attrib_array[attr_found][2];  //attr val2
```

```
    tree_result_array[array_counter+3]= attrib_array[attr_found][3]; //attr val3
    for(i=0; i <= 2; i++)
    {
//Case 1: examine for leaf event
      child_incr =((i+1)*4);
      if(((attr_cnt_array[attr_found][(2*i)])==0)|| ((attr_cnt_array[attr_found]
[(2*i)+1])==0))
      {
//Set leaf values of child nodes
         tree_result_array[array_counter+child_incr+0] = "leaf";
         if((attr_cnt_array[attr_found][2*i])>  (attr_cnt_array[attr_found][(2*i)+1]))
         {
           tree_result_array[array_counter+child_incr+1] = "play";
           tree_result_array[array_counter+child_incr+2] = "null";
           tree_result_array[array_counter+child_incr+3] = "null";
         }
         else
         {
           tree_result_array[array_counter+child_incr+1] = "no play";
           tree_result_array[array_counter+child_incr+2] = "null";
           tree_result_array[array_counter+child_incr+3] = "null";
         }
      }
//Case 2: null encountered - load null
      else if((tree_result_array[array_counter+1]) == "null")
      {
        tree_result_array[array_counter+child_incr+0] = "null"; //attr name
        tree_result_array[array_counter+child_incr+1] = "null"; //attr val1
        tree_result_array[array_counter+child_incr+2] = "null"; //attr val2
        tree_result_array[array_counter+child_incr+3] = "null"; //attr val3
       }
       else
       {
//Case 3: load new attribute
         new_attr_found = get_next_entropy();   //provides row# attrib_array
         if((new_attr_found!=99)&& (entropy_array[new_attr_found][0]<=entropy_max))
          {
//load attrib to tree_result_array
            tree_result_array[array_counter+child_incr+0]= attrib_array[new_attr_found][0];
//attr name
            tree_result_array[array_counter+child_incr+1] = attrib_array[new_attr_found][1];
//attr val1
            tree_result_array[array_counter+child_incr+2] = attrib_array[new_attr_found][2];
//attr val2
            tree_result_array[array_counter+child_incr+3] = attrib_array[new_attr_found][3];
//attr val3
//determine grandchild increment
            if(child_incr == 4) //left child
              gr_child_incr = child_incr + 12;
            else if (child_incr == 8) //center child
              gr_child_incr = child_incr + 20;
```

```
            else if (child_incr == 12) //right child
              gr_child_incr = child_incr + 28;
            for(k=0; k<=2; k++)
            {
//load grand child leaves
            if(((attr_cnt_array[new_attr_found][2*k])!=99)&& ((attr_cnt_array[new_attr_found]
[(2*k)+1])!=99)&& (tree_result_array[array_counter+child_incr+0] != "leaf")&& (tree_result_
array[array_counter+child_incr+k] != "null"))
            {
              tree_result_array[array_counter+gr_child_incr+0+(k*4)] = "leaf";
              if((attr_cnt_array[new_attr_found][2*k])> (attr_cnt_array[new_attr_found]
[(2*k)+1]))
              {
              tree_result_array[array_counter+gr_child_incr+1+(k*4)] = "play";
              tree_result_array[array_counter+gr_child_incr+2+(k*4)] = "null";
              tree_result_array[array_counter+gr_child_incr+3+(k*4)] = "null";
              }
              else
              {
              tree_result_array[array_counter+gr_child_incr+1+(k*4)] = "noplay";
              tree_result_array[array_counter+gr_child_incr+2+(k*4)] = "null";
              tree_result_array[array_counter+gr_child_incr+3+(k*4)] = "null";
              }
            }
          }
        }
        else
        {
//load 'null"
        tree_result_array[array_counter+child_incr+0] = "null";  //attr name
        tree_result_array[array_counter+child_incr+1] = "null";  //attr val1
         tree_result_array[array_counter+child_incr+2] = "null"; //attr val2
         tree_result_array[array_counter+child_incr+3] = "null"; //attr val3
        }
      }
    }               //end for(i...
   }                //end if(attrb_found != 99)
   else             //if(attrb_found != 99)
   {
    ;
   }
   array_counter = array_counter + 16;
  }                 //end for (j..
}
//*****************************************************************
unsigned int get_next_entropy(void)
{
  unsigned int n;
  double low_entropy = 1.0;
  unsigned int row_found;
  unsigned int found_one = 0;
```

```
  unsigned int lowest_row_found = 10;
  for(n=0; n <= num_attrib; n++)
  {
    if((entropy_array[n][0] < low_entropy)&&(attr_cnt_array[n][7] == 1))
    {
      low_entropy = entropy_array[n][0];
      lowest_row_found = n;
      found_one = 1;
    }
  }                    //end for
  attr_cnt_array[lowest_row_found][7] = 0;
  row_found = lowest_row_found;
  if(found_one == 0) row_found = 99;
  return row_found;
}
//*********************************************************************
void print_tree_result_array(void)
{
  unsigned int i;
  Serial.print("\t\t\t\t");
  for(i=0; i< (num_attrib * 4 * 4); i++)
  {
    if(tree_result_array[i] != "x")
    {
      Serial.print(i);
      Serial.print(":");
      Serial.print(tree_result_array[i]);
      Serial.print(" ");
    }
    if((i==3)||(i==15)||(i==47))
    {
      Serial.println(" ");
      if(i==3)
      Serial.print("\t\t");
    }
  }
  Serial.println(" ");
}
//*********************************************************************
void reset_tree_result_array(void)
{
  unsigned int i;
  for(i=0; i< (num_attrib * 4 * 4); i++)
  {
     tree_result_array[i] = "x" ;
   }
}
//*********************************************************************
```

После того как дерево собрано, его можно использовать для обработки новых значений признаков за пределами исходного на-

бора данных с целью принятия решения. Мы рассмотрим это далее в разделах «Приложения».

4.5. Приложение: классификатор KNN

Ранее в этой главе мы исследовали классификатор цветов KNN для трех цветов (красный, зеленый и синий). Расширьте пример до восьми разных цветов. Изучите компромиссы, связанные с изменением значения «K» и количества образцов, взятых для каждого цвета.

4.6. Приложение: дерево решений

В предыдущем разделе мы рассмотрели проектирование и реализацию дерева решений. В этом разделе мы расширим наши результаты.

Проделайте следующее.

1. Функция `print_tree_result_array` предоставляет базовое изображение результирующего дерева решений. Измените эту функцию так, чтобы отобразить взаимосвязи между различными узлами дерева.
2. Ниже представлен цикл `loop()` из кода дерева решений, приведенного в предыдущем разделе. Цикл обеспечивает обход дерева и принятие решений. Объедините две части кода и протестируйте дерево. Добавьте «циклическую» часть кода в UML-диаграмму действий, представленную на рис. 4.8.
3. Разработайте новое дерево решений, используя выбранный вами набор данных.

```
//*********************************************************************
void loop()
{
  unsigned int i = 0;
  unsigned int i_stop = 0;
  int j;
  i_stop = 0;
//navigate decision tree with new values of attribute values
  while((tree_result_array[i] != "x")&&(i_stop!=99))
```

```
  {
//root node processing
    Serial.println(" ");
    Serial.println("Decision Tree Processing");
    Serial.println(tree_result_array[i]);
    if(tree_result_array[i] != "leaf")
    {
      Serial.println("Select the attribute value:");
    }
    if(tree_result_array[i] != "leaf")
    {
      Serial.print("1: ");
      Serial.println(tree_result_array[i+1]);
    }
    else
    {
      Serial.println(tree_result_array[i+1]);
      i_stop = 99;
    }
    if(tree_result_array[i+2] != "null")
//&&(tree_result_array[i] != "leaf"))
    {
      Serial.print("2: ");
      Serial.println(tree_result_array[i+2]);
    }
    if(tree_result_array[i+3] != "null")
//&&(tree_result_array[i] != "leaf"))
    {
      Serial.print("3: ");
      Serial.println(tree_result_array[i+3]);
    }
    if(i_stop != 99)
    {
//flush input buffer
      while(Serial.available() >0)
      {
        Serial.read();
      }
//request input value from user via serial monitor
      Serial.println("Enter a value, press [Send]");
      while(Serial.available()==0){}                  //wait for user input data
      j = Serial.parseInt();
//Serial.print("i: "); Serial.print(i);
//Serial.print(" j: "); Serial.println(j);
//children level
      if((j==1)&&(i==0))
        i = 4;                        //left
      else if((j==2)&&(i==0))
        i = 8;                        //center
      else if ((j==3)&&(i==0))
        i = 12;                       //right
```

```
//grandchildren level
      else if((j==1)&&(i==4))
        i = 16;                     //left
      else if((j==2)&&(i==4))
        i = 20;                     //center
      else if((j==3)&&(i==4))
        i = 24;                     //right
//grandchildren level
      else if((j==1)&&(i==8))
        i_stop = 99;  //28          //left
      else if((j==2)&&(i==8))
        i_stop = 99;  //32          //center
      else if((j==3)&&(i==8))
        i_stop = 99;  //36          //right
//grandchildren level
      else if((j==1)&&(i==12))
        i = 40;                     //left
      else if((j==2)&&(i==12))
        i = 44;                     //center
      else if((j==3)&&(i==12))
        i_stop = 99;                //right
      else
        i = 0;
    } //i!=99
  }//while
    i = 0;
}//loop
//*********************************************************************
```

4.7. Выводы

Мы начали эту главу с обзора приложений искусственного интеллекта и машинного обучения, а также изложения истории данного направления. Затем изучили способы классификации: метод K ближайших соседей и метод дерева решений. В оставшейся части книги мы сосредоточимся на дополнительных концепциях и приложениях искусственного интеллекта и машинного обучения для систем на базе микроконтроллеров.

4.8. Задания

1. Выберите какое-нибудь одно нововведение в новейшей истории искусственного интеллекта и машинного обучения. Изучи-

те предложенные в нем инновации и напишите одностраничный реферат. Представьте краткое изложение базовой теории инновации и ее основного вклада в науку об искусственном интеллекте и машинном обучении.
2. В алгоритме KNN какое значение имеет выбор «K»? Каково значение выбора меньшего или большего значения «K»?
3. Как в алгоритме KNN алгоритм будет реагировать на образец, для которого он не был обучен?
4. Каковы компромиссы в алгоритме KNN при использовании меньшего или большего обучающего набора?
5. Ранее в этой главе мы исследовали классификатор цветов KNN для трех цветов (красный, зеленый и синий). Расширьте пример до восьми разных цветов.
6. Что такое дерево решений? Как оно построено?
7. Что такое энтропия? Что она собой представляет? Как рассчитывается?
8. Функция дерева решений `print_tree_result_array` предоставляет базовое изображение результирующего дерева решений. Измените эту функцию, чтобы отобразить взаимосвязи между различными узлами дерева.
9. Разработайте новое дерево решений, используя приготовленный вами набор данных.

Источники

1. Arduino Team, Get startedwith machine learning on Arduino, https://www.blog.arduino.cc, October 15, 2019.
2. G. Lawton, Machine Learning on Microcontrollers Enables AI, https://www.targettech.com, November 17, 2021.
3. J. P. Mueller and L. Massaron, Artificial Intelligence forDummies, JohnWiley and Sons, Inc, 2018.
4. C. A. Pickover, Artificial Intelligence an Illustrated History, Sterling Publishing Co., Inc., 2019.
5. J. R. Quinlan, Induction of Decision Trees, Machine Learning 1: pages 81–106, 1986.
6. B. W. Silverman and M. C. Jones, textitE. Fix and J. L. Hodges (1951): An Important Contribution to Nonparametric Discriminant Analysis andDensity Estimation: Commentary on Fix andHodges (1951), In-

ternational Statistical Review, Dec. 1989, Vol. 57, No. 3 (Dec. 1989), pp. 233–238.
7. AI Computing Comes to Memory Chips, IEEE Spectrum, Jan 2022.
8. O. Theobald, Machine Learning for Absolute Beginners: A Plain English Introduction, second edition, 2017.
9. B. J. Wythoff, Backpropagation Neural Networks – A Tutorial, Chemometrics and Intelligent Laboratory Systems, 18 (1993), 115–155.

ГЛАВА

5

Нечеткая логика

Прочитав эту главу, читатель должен иметь возможность:

- предоставить реальный пример системы управления;
- описать различия между традиционным построением системы управления и системой, реализованной с использованием методов нечеткой логики;
- описать сходства и различия между булевой двухзначной логикой и многозначной нечеткой логикой;
- кратко описать истоки подхода нечеткой логики к проектированию систем управления;
- нарисовать схему и описать этапы проектирования контроллера нечеткой логики;
- реализовать контроллер нечеткой логики с использованием встроенной библиотеки нечеткой логики arduino (efll);
- разработать систему управления на нечеткой логике для управления автономным роботом, перемещающимся по лабиринту.

5.1. Обзор концепций

В этой главе описывается, как управлять процессом с помощью методов нечеткой логики. До этого момента в рассказе об Arduino мы использовали методы булевой логики. То есть цифровые логические входы и выходы микроконтроллера имели либо логическую единицу (истина, true), либо логический ноль (ложь, false). Фактически мы при-

ложили немало усилий, чтобы описать правильные методы подключения периферии, позволяющие сохранить эти логические значения.

В этой главе мы исследуем нечеткую логику, которая допускает несколько уровней истины между логической единицей и нулем. Мы обнаруживаем, что многие реальные проблемы управления поддаются реализации с помощью нечеткой логики. Например, во время путешествия наша цель – благополучно добраться до желаемого пункта назначения. По пути мы должны постоянно быть внимательными к меняющимся дорожным условиям, другим транспортным средствам и даже диким и домашним животным. Многие автомобили теперь оборудованы системами управления, автоматически регулирующими скорость в ответ на обнаруженные препятствия. Препятствия могут варьироваться от медленно движущихся транспортных средств до крупных животных[1].

Интуитивно понятно, что система будет по-разному реагировать на препятствия, находящиеся на близком и далеком расстояниях. Если большое животное (например, лось) выходит прямо перед моим автомобилем, я бы хотел, чтобы система управления быстро приложила максимальное тормозное усилие, чтобы привести транспортное средство к контролируемой резкой остановке. С другой стороны, если система управления обнаруживает препятствие, находящееся намного дальше от автомобиля, тормозная система может быть задействована плавно, чтобы только слегка притормозить автомобиль. Нечеткая логика позволяет спроектировать систему такого типа. Далее в этой главе мы более подробно исследуем подобную систему управления транспортным средством.

Традиционное проектирование системы управления основано на разработке математической модели системы. Такие системы десятилетиями использовались для управления всеми типами сложных систем. В 1965 году Л. А. Заде (L. A. Zadeh) в своей плодотворной работе «Нечеткие множества» («Fuzzy Sets», [11]) ввел концепцию «континуума степеней принадлежности» со значениями принадлежности в диапазоне от нуля до единицы. Он также предоставил применительно к нечеткой логике функции, обычно встречающиеся в булевой логике (например, И, ИЛИ и т. д.). Л. А. Заде отметил, что многие процессы принятия решений человеком не используют точные математические модели.

---

[1] Несколько лет назад во время рыбалки моя машина столкнулась со своенравным оленем на межштатной автомагистрали в Восточной Монтане. Я благодарен судьбе, что я и мои пассажиры пережили эту встречу. – *Прим. авт.*

В этой главе мы более подробно исследуем конструкцию контроллеров нечеткой логики. Мы начнем с краткого обзора ключевых концепций. Далее рассмотрим встроенную библиотеку нечеткой логики Arduino (eFLL). Библиотека была разработана командой (см. [1]) из исследовательской группы робототехники Государственного университета Пиауи в Терсини, Пиауи, Бразилия. Библиотека содержит функции для проектирования сложных систем управления с нечеткой логикой, а также предоставляет несколько отличных примеров, которые можно использовать в качестве шаблонов для проектирования других систем [1]. Мы рассмотрим примеры довольно подробно. В более раннем писательском проекте [8] мы с Дэниелом Паком (D. J. Pack) исследовали автономного робота, перемещающегося по лабиринту и оснащенного двумя ИК-датчиками для обнаружения стен лабиринта. В разделе «Приложение» мы экипируем робота Dagu Magician из главы 3 системой управления на основе нечеткой логики с помощью eFLL.

5.2. Теория

В этом разделе мы даем обзор проектирования системы управления на основе нечеткой логики. Наша цель – обеспечить систематический, поэтапный процесс проектирования. Попутно мы введем соответствующую терминологию и понятия. Для иллюстрации ключевых понятий мы намеренно используем много цифр. Источники для этого раздела включают [1, 4, 5, 8].

На рис. 5.1 представлен обзор процесса проектирования системы нечеткого управления. Общая цель – управлять системой, получая точную и четкую входную информацию от входных датчиков и преобразователей и сопоставляя ее с четкими выходными управляющими сигналами с использованием серии правил, представленных в форме «if–then» («если–то»).

По ходу дела четкие входные сигналы подвергаются размытию и преобразуются в лингвистические (словесные) переменные. Затем входные данные объединяются с использованием логических соотношений AND (И) и OR (ИЛИ) для разработки серии правил в стиле «if–then». Правила связывают входные переменные с желаемыми выходными управляющими сигналами. Правила разрабатывает «эксперт» – то есть вы как разработчик системы. Может существовать несколько правил, связывающих входные условия

с желаемыми выходными управляющими сигналами. Каждое правило оценивается, и если оно активировано, ему присваивается определенный вес. Для данного набора входных условий может сработать несколько правил одновременно, каждое со своим весом.

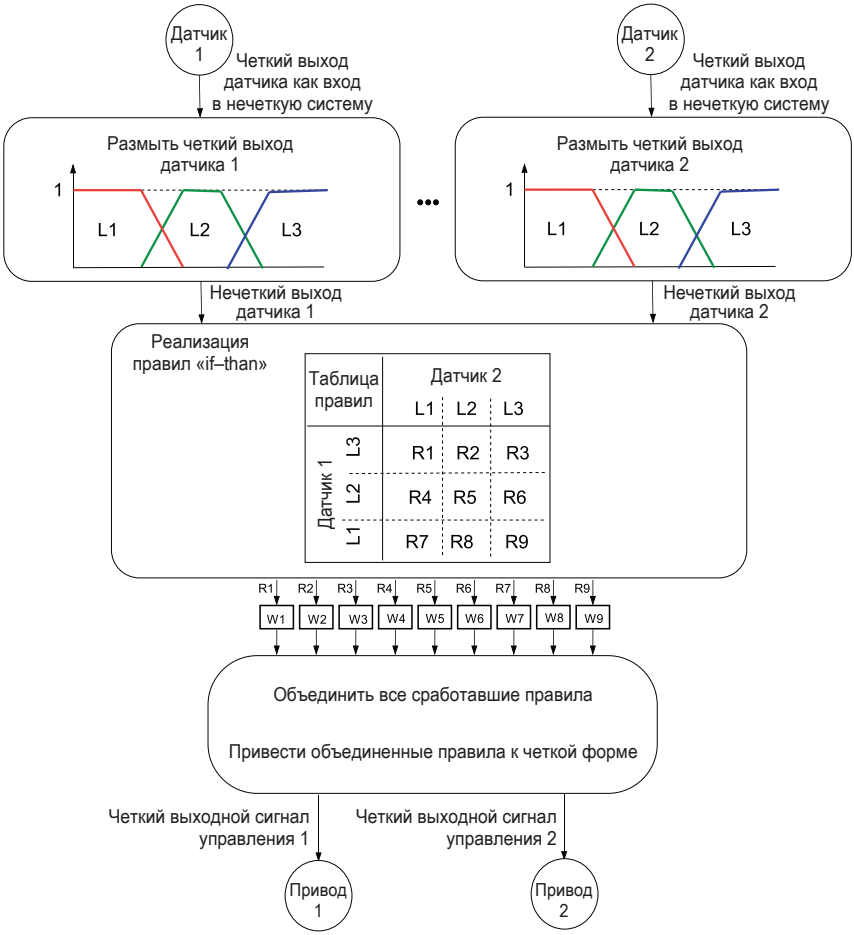


Рис. 5.1. Построение системы нечеткого управления

Сработавшие правила затем объединяются, чтобы определить общий нечеткий ответ. Затем нечеткий ответ преобразуется в однозначный, чтобы обеспечить четкие выходные управляющие сигналы.

Давайте подробнее рассмотрим каждый шаг. Для иллюстрации каждого шага мы используем расширенный пример робота, следующего по нечеткому управляемому лабиринту.

5.2.1. Установить цель, входы и выходы системы нечеткого управления

Прежде чем приступить к детальному проектированию системы нечеткого управления, очень важно определить общую цель системы, доступные входные данные и желаемые выходные данные. Входные данные будут сопоставлены с выходными в процессе проектирования с использованием набора правил.

Пример **В этом разделе мы используем пример робота с нечетким управлением, следующего по лабиринту. В разделе «Приложение» главы 3 мы представили роботизированную платформу Dagu Magician. Мы там разработали систему управления, состоящую из трех ИК-датчиков в качестве входных устройств и двух двигателей постоянного тока в качестве выходных для автономной навигации робота по лабиринту. Общая цель заключалась в том, чтобы как можно быстрее пройти по лабиринту, не касаясь стен лабиринта. В этом случае мы снабжаем робота Dagu системой нечеткого управления для достижения той же цели.**

5.2.2. Размыть четкий сигнал датчика

После того как цели системы установлены, следующим шагом является определение функций принадлежности нечетких входных данных для каждого из входных сигналов датчика. Четкие сигналы датчиков передаются серией входных преобразователей. Для каждого датчика разрабатывается входная функция принадлежности, как показано на рис. 5.1. Входные функции состоят из ряда лингвистических (словесных) переменных. Диапазон лингвистических переменных определяется функцией-трапецией. Различные формы функций-трапеций показаны на рис. 5.2 [1].

Конкретные функции определяются с помощью профиля датчика. Четкий числовой выходной сигнал датчика сопоставляется с определенной лингвистической переменной. Если четкий числовой выходной сигнал датчика соответствует двум разным лингвистическим переменным, выбирается лингвистическая переменная с меньшим значением.

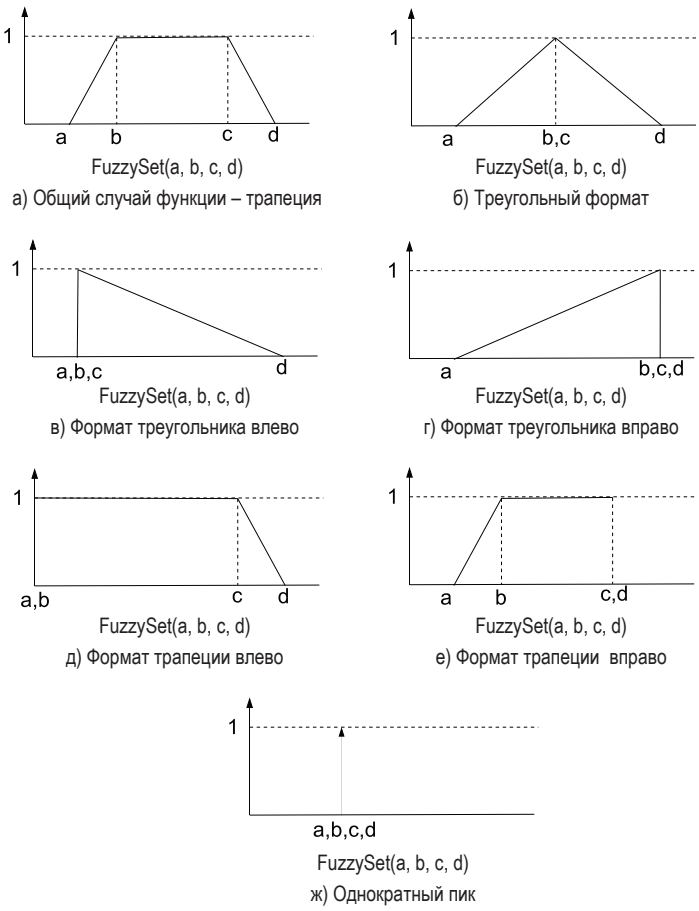
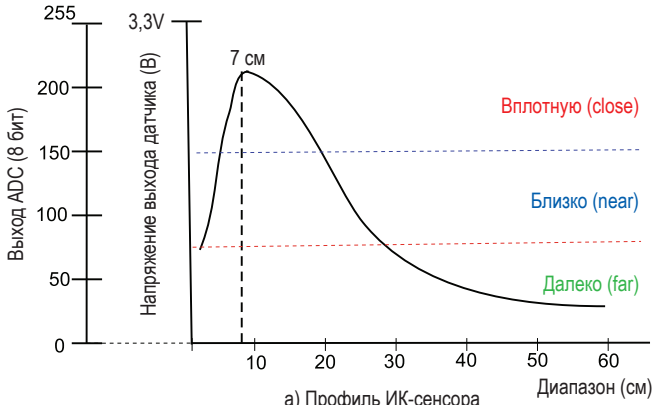


Рис. 5.2. Нечеткие функции [1]

Пример **В случае с роботом мы рассмотрим только два ИК-датчика для навигации по лабиринту. Чтобы робот мог обнаруживать препятствия прямо перед собой, служит фронтальный ИК-датчик, кроме того, используется правый ИК-датчик.**

Чтобы спроектировать входные функции принадлежности для переднего и правого ИК-датчиков, профиль ИК-датчика разделен на три различные зоны: препятствие вплотную (close), препятствие недалеко (near) и препятствие далеко (far), как показано на рис. 5.3, *а*. Профиль ИК-датчика используется вместе со значением выходного сигнала для построения входных функций принадлеж-

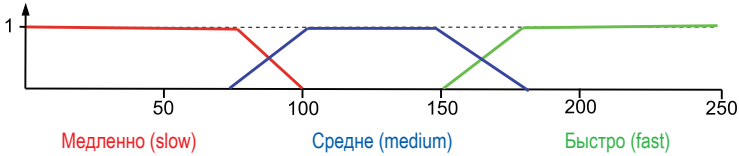
ности, как показано на рис. 5.3, *б*. Входная функция принадлежности предусмотрена как для переднего, так и для правого датчика.



а) Профиль ИК-сенсора



б) Входное множество функции принадлежности для ИК-датчиков



в) Выходное множество функции принадлежности для левого и правого двигателей

right sensor	front sensor: f_close	f_near	f_far
r_close	① l_med / l_slow / r_med	② f_slow / l_slow / r_slow	③ f_fast / l_fast / r_fast
r_near	④ f_slow / l_slow / r_slow	⑤ f_slow / l_slow / r_slow	⑥ f_fast / l_fast / r_fast
r_far	⑦ r_med / l_med / r_slow	⑧ f_slow / l_slow / r_slow	⑨ f_fast / l_fast / r_fast

г) Набор правил

right sensor	front sensor: f_close	f_near	f_far
r_close	① l_med / l_slow / r_med	② f_slow / l_slow / r_slow	③ f_fast / l_fast / r_fast
r_near	② f_slow / l_slow / r_slow	② f_slow / l_slow / r_slow	③ f_fast / l_fast / r_fast
r_far	④ r_med / l_med / r_slow	② f_slow / l_slow / r_slow	③ f_fast / l_fast / r_fast

д) Итоговый набор правил

Рис. 5.3. Разработка входной и выходной нечетких функций принадлежности

5.2.3. Применение правил

Пусть разработан набор правил в форме «if (условие) – then (следствие)» для связывания лингвистических переменных входных функций принадлежности с желаемыми значениями выходных функций принадлежности. Конкретные правила разрабатываются путем рассмотрения различных комбинаций входных переменных для формирования условий. Несколько входных переменных могут быть связаны с помощью логических связок AND и OR. Для связки AND выбирается минимальное значение входной лингвистической переменной. Для связки OR выбирается, наоборот, максимальное значение входной лингвистической переменной. Желаемый результат (следствие) для данной комбинации входных переменных определяется экспертом (то есть вами – разработчиком системы). Выходные величины функции принадлежности определяются путем привязки выходных лингвистических переменных к желаемым четким числовым значениям.

Пример **На рис. 5.3, *в* мы разработали выходные функции принадлежности для левого и правого двигателей. Четкие числовые выходные значения находятся в диапазоне от 0 до 250. Они будут служить входными данными для функции широтно-импульсной модуляции (PWM) с целью управления скоростью левого и правого двигателей для реализации различных поворотов.**

Для построения правил, связывающих входы с выходами, комбинации входных лингвистических переменных для правого и переднего датчиков помещаются в таблицу. Желаемый результат для каждой комбинации входных данных затем определяется экспертом (вами). Например, для правого датчика при «r_close» и переднего датчика при «f_close» желаемым действием робота является поворот влево со средней скоростью («l_med»). Это достигается путем установки левого двигателя на медленный режим («l_slow») и правого двигателя на средний режим («r_med»). Итоговая таблица представлена на рис. 5.3, *г*.

5.2.4. Объединение активных правил и восстановление четкости выхода

Чтобы определить четкие выходные данные для управления системой, все правила, которые предоставили выходные данные, объ-

единяются вместе, дабы получить единый выходной результат. Для этого существуют различные методы. Встроенная библиотека нечеткой логики Arduino (eFLL) использует метод минимума Мамдани (Mamdani) для объединения выходных данных и метод центра площади для получения четких выходных данных из нечетких (defuzzification) [3, 7, 9, 10][1].

Пример **На основе описанных процессов предоставляются два четких числовых сигнала на двигатели для реализации желаемого поворота при обходе стен лабиринта.**

В разделе «Приложение» далее мы разрабатываем реальный код с использованием встроенной библиотеки нечеткой логики Arduino (eFLL).

5.3. Arduino-библиотека eFLL

Далее мы рассмотрим встроенную библиотеку нечеткой логики Arduino (eFLL). Библиотека была разработана командой [1] из исследовательской группы робототехники Государственного университета Пиауи в Терсини, Пиауи, Бразилия. Библиотека содержит функции и репрезентативные примеры для проектирования сложных систем управления с нечеткой логикой. Примеры могут служить шаблонами для проектирования других систем. Команда предоставила выдающуюся услугу по обеспечению легкого доступа к концепциям нечеткой логики для сообщества Arduino [1].

Здесь мы более подробно рассмотрим некоторые примеры библиотеки нечеткой логики eFLL.

5.3.1. Простой пример

В начале главы мы обсуждали систему обхода препятствий для управления транспортным средством. Интуитивно мы хотели создать систему, которая по-разному реагировала бы на препятствие, находящееся близко или далеко. Скажем, если большое животное

[1] Обзоры этих методов на русском языке см., например, по адресу https://habr.com/ru/articles/113020/ и https://studopedia.su/21_52335_privedenie-rezultatov-k-chetkosti.html. – *Прим. перев.*

(например, лось) выходит прямо перед моим автомобилем, я бы хотел, чтобы система управления быстро приложила сильное тормозное усилие, дабы привести транспортное средство к контролируемому, но резкому торможению. С другой стороны, если система управления обнаруживает препятствие, находящееся намного дальше от автомобиля, тормозная система может включаться плавно, чтобы немного замедлить автомобиль.

Библиотека eFLL предоставляет пример под названием «Arduino_simple_sample», реализующий такую систему. Система содержит единственный вход под названием «distance» («расстояние») с лингвистическими переменными «small» («маленькое»), «safe» («безопасное») и «big» («большое»). Входная функция принадлежности показана на рис. 5.4. Система содержит один выходной сигнал под названием «speed» («скорость») с лингвистическими переменными «slow» («медленная»), «average» («средняя») и «fast» («высокая»). Выходная функция принадлежности показана на рис. 5.4 [1].

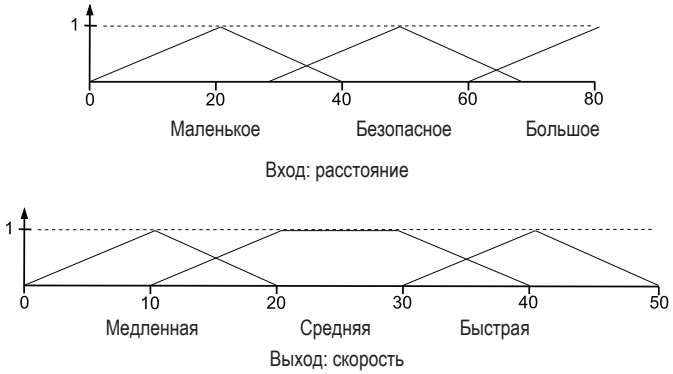


Рис. 5.4. Нечеткие входы и выходы

В системе реализованы следующие правила:

- правило 1: IF <расстояние> = <маленькое> THEN <скорость> = <медленная>;
- правило 2: IF <расстояние> = <безопасное> THEN <скорость> = <средняя>;
- правило 3: IF <расстояние> = <большое> THEN <скорость> = <высокая>.

Настоятельно рекомендуется читателю загрузить и запустить скетч «Arduino_simple_sample» и ознакомиться с его работой. Модифицированная версия этого скетча представлена ниже. Вместо генератора случайных чисел для формирования входных данных системы для имитации входного сигнала датчика расстояния используется потенциометр, а в качестве выходного сигнала индикатора скорости используется светодиод (см. рис. 5.5). Кроме того, были добавлены дополнительные операции для определения соответствия нечетких лингвистических переменных входных и выходных функций принадлежности.

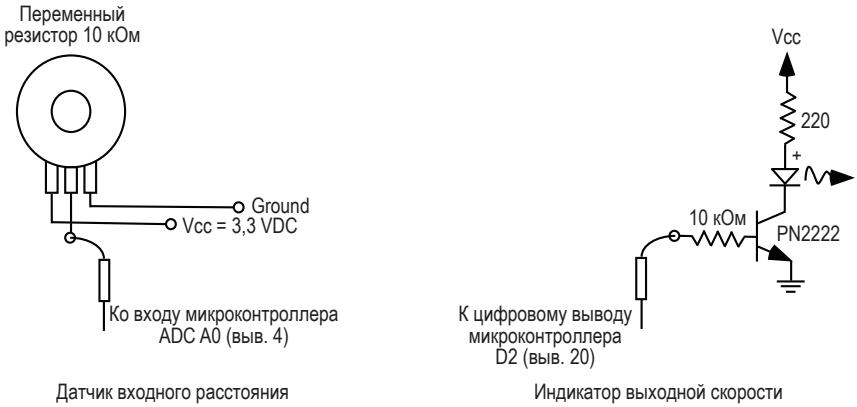


Рис. 5.5. Симулятор входа и выхода

```
//*********************************************************
//arduino_simple_sample_adc: this sketch provides a
//basic example of the Arduino Embedded Fuzzy Logic
//Library (eFLL).
//
//This sketch simulates a speed control feature of an
//automatic braking system.
//- Fuzzy Input: distance to obstruction (e.g. another
// vehicle or an animal)
//- Fuzzy Output: vehicle speed
//
//Author of eFLL Library: A.J. Alves
//
//Example adapted to include sensor speed input,
//output to an LED, and display of input and output
//pertinence
//- Use 10K pot on ADC A0 (pin 4) as input distance
//- Use LED on D3 (pin 21) to indicate speed
```

```
//*********************************************************
#include <Fuzzy.h>
#define speed_LED 3 //physical pin 21
//Instantiating a Fuzzy object
Fuzzy *fuzzy = new Fuzzy();
//Fuzzy Input
//Instantiating a FuzzySet objects
FuzzySet *small = new FuzzySet(0, 20, 20, 40);
FuzzySet *safe = new FuzzySet(30, 50, 50, 70);
FuzzySet *big = new FuzzySet(60, 80, 80, 80);
//Fuzzy Output
//Instantiating a FuzzySet object
FuzzySet *slow = new FuzzySet(0, 10, 10, 20);
FuzzySet *average= new FuzzySet(10, 20, 30, 40);
FuzzySet *fast = new FuzzySet(30, 40, 40, 50);
void setup()
{
  Serial.begin(9600); //Set the Serial output
//randomSeed(analogRead(0)); //Set a random seed
//Instantiating a FuzzyInput object
//Including the FuzzySet into FuzzyInputs
  FuzzyInput *distance = new FuzzyInput(1);
  distance->addFuzzySet(small);
  distance->addFuzzySet(safe);
  distance->addFuzzySet(big);
  fuzzy->addFuzzyInput(distance);
//Instantiating a FuzzyOutput objects
//Including the FuzzySet into FuzzyOutputs
  FuzzyOutput *speed = new FuzzyOutput(1);
  speed->addFuzzySet(slow);
  speed->addFuzzySet(average);
  speed->addFuzzySet(fast);
  fuzzy->addFuzzyOutput(speed);
//Building FuzzyRule "IF distance = small then speed = slow"
//Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *ifDistanceSmall = new FuzzyRuleAntecedent();
//Creating a FuzzyRuleAntecedent with just a single FuzzySet
  ifDistanceSmall->joinSingle(small);
//Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenSpeedSlow = new FuzzyRuleConsequent();
//Including a FuzzySet to this FuzzyRuleConsequent
  thenSpeedSlow->addOutput(slow);
//Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule01 = new FuzzyRule(1, ifDistanceSmall, thenSpeedSlow);
//Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule01);
//Building FuzzyRule "IF distance = safe THEN speed = average"
//Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *ifDistanceSafe = new FuzzyRuleAntecedent();
//Creating a FuzzyRuleAntecedent with just a single FuzzySet
  ifDistanceSafe->joinSingle(safe);
```

```
//Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenSpeedAverage = new FuzzyRuleConsequent();
//Including a FuzzySet to this FuzzyRuleConsequent
  thenSpeedAverage->addOutput(average);
//Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule02 = new FuzzyRule(2, ifDistanceSafe, thenSpeedAverage);
//Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule02);
//Building FuzzyRule "IF distance = big THEN speed = high"
//Instantiating a FuzzyRuleAntecedent objects
  FuzzyRuleAntecedent *ifDistanceBig = new FuzzyRuleAntecedent();
//Creating a FuzzyRuleAntecedent with just a single FuzzySet
  ifDistanceBig->joinSingle(big);
//Instantiating a FuzzyRuleConsequent objects
  FuzzyRuleConsequent *thenSpeedFast = new FuzzyRuleConsequent();
//Including a FuzzySet to this FuzzyRuleConsequent
  thenSpeedFast->addOutput(fast);
//Instantiating a FuzzyRule objects
  FuzzyRule *fuzzyRule03 = new FuzzyRule(3, ifDistanceBig, thenSpeedFast);
//Including the FuzzyRule into Fuzzy
  fuzzy->addFuzzyRule(fuzzyRule03);
}
void loop()
{
//Getting a random value
//int input = random(0, 80);
  int input_sensor = analogRead(0);
  int input = map(input_sensor, 0, 1023, 0, 80);
//Printing something
  Serial.println("\n\n\nEntrance: ");
  Serial.print("\t\t\tDistance: ");
  Serial.println(input);
//Set the random value as an input
  fuzzy->setInput(1, input);
//Running the Fuzzification
  fuzzy->fuzzify();
//Running the Defuzzification
  float output = fuzzy->defuzzify(1);
//Printing something
  Serial.println("Result: ");
  Serial.print("\t\t\tSpeed: ");
  Serial.println(output);
  Serial.println("Input: ");
  Serial.print("\tDistance: small-> ");
  Serial.print(small->getPertinence());
  Serial.print(", safe-> ");
  Serial.print(safe->getPertinence());
  Serial.print(", big-> ");
  Serial.println(big->getPertinence());
  Serial.print("\tSpeed: slow-> ");
  Serial.print(slow->getPertinence());
```

```
  Serial.print(", average-> ");
  Serial.print(average->getPertinence());
  Serial.print(", fast-> ");
  Serial.print(fast->getPertinence());
//illuminate LED at intensity consistent with speed
  int output_int = (int)(output);     //cast output speed to int
//analogWrite range 0 to 255
  int output_int_PWM = map(output_int, 0, 80, 0, 255);
  analogWrite(speed_LED, output_int_PWM);
//wait 2 seconds
  delay(2000);
}
//*********************************************************
```

Для проверки реализованного контроллера нечеткой логики выполняется несколько тестовых прогонов, как показано на

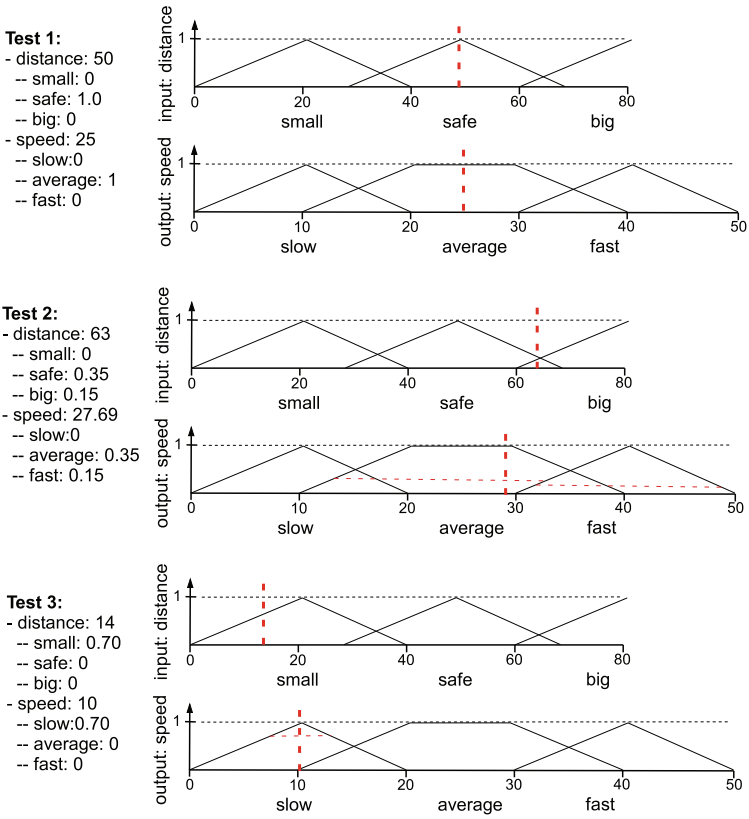


Рис. 5.6. Примеры тестирования нечеткого управления

рис. 5.6. Результаты представлены для трех тестовых запусков. Читателю рекомендуется применить три правила управления с соответствующими данными, предоставленными для каждой входной и выходной функции членства, чтобы проверить работу контроллера. Напомним, на этапе получения четких выходных данных из нечетких (defuzzification) контроллер применяет подход минимума Мамдани для объединения сработавших правил, а затем выполняет расчет центра области для определения четкого результата [7].

5.3.2. Расширенный пример

В примере далее представлена более сложная система управления транспортным средством. Этот пример также имеется в библиотеке eFLL и называется «Arduino_advanced_sample». Система содержит три входа: «distance» («расстояние»), «speedInput» («входная скорость») и «temperature» («температура»). Лингвистические переменные для каждой входной функции принадлежности показаны на рис. 5.7. Система также содержит две выходные переменные, называемые «risk» («риск») и «speedOutput» («выходная скорость»), с лингвистическими переменными, показанными на рис. 5.7.

В системе реализованы следующие правила:

- правило 1: IF <расстояние вплотную> AND <скорость велика> OR <температура холодно> THEN <риск высокий> AND <скорость медленная>;
- правило 2: IF <расстояние безопасное> AND <скорость нормальная> OR <температура хорошая> THEN <риск средний> AND <скорость нормальная>;
- правило 3: IF <расстояние большое> AND <скорость медленная> OR <температура жарко> THEN <риск низкий> AND <скорость высокая>.

Настоятельно рекомендуется читателю загрузить и запустить скетч «Arduino_advanced_sample» и ознакомиться с его работой. В целях экономии места код здесь дублироваться не будет.

Код моделирует входные данные датчика с использованием случайных чисел. Результаты нескольких тестовых запусков представлены на рис. 5.8 и 5.9. Читателю рекомендуется применить правила управления с соответствующими данными, предоставленными для каждой входной и выходной функции принадлежности, чтобы про-

верить работу контроллера. Для сложного правила, включающего логические операторы AND и OR, напомним, что минимальное значение выбирается для операции AND, а максимальное значение выбирается для OR. Кроме того, снова напомним, что на этапе получения четких выходных данных из нечетких (defuzzification) контроллер применяет подход минимума Мамдани для объединения сработавших правил, а затем выполняет расчет центра области для определения четкого результата [7].

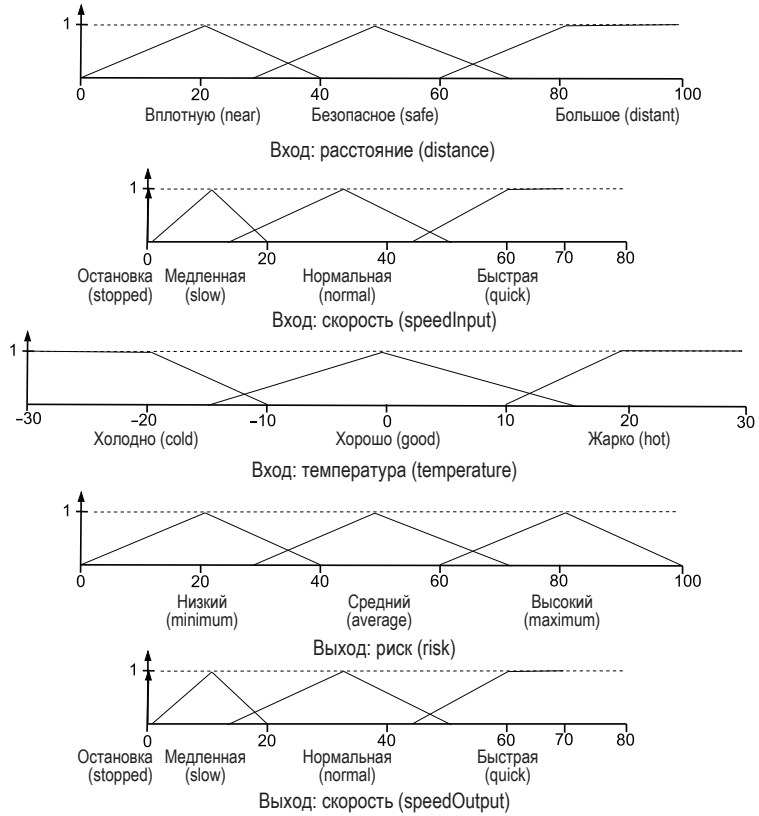


Рис. 5.7. Нечеткие входы и выходы

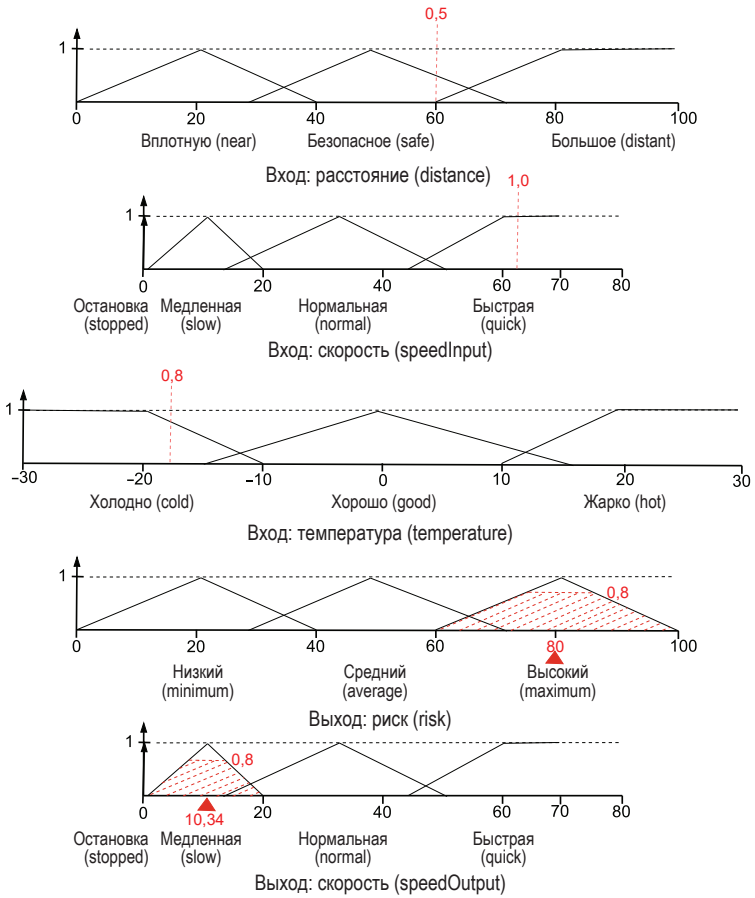


Рис. 5.8. Нечеткие входы и выходы. Тест 1

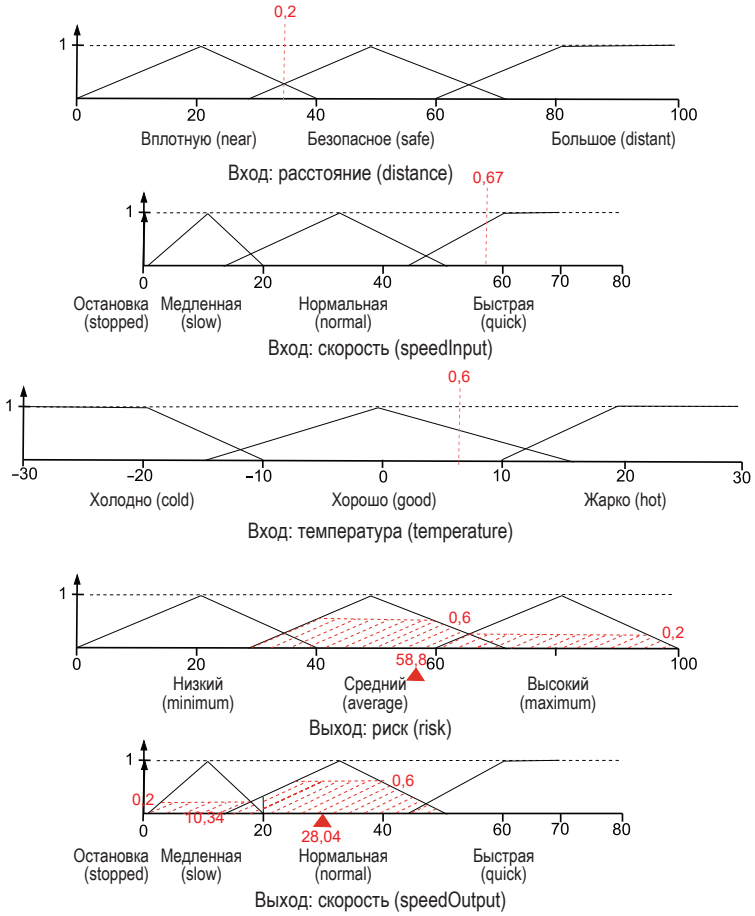


Рис. 5.9. Нечеткие входы и выходы. Тест 2

5.4. Применение

В разделе «Приложение» главы 3 мы оснастили робота Dagu Magician тремя ИК-датчиками для обнаружения стен лабиринта. Серия традиционных (четких) утверждений «if – then» использовалась для определения двигательных действий, необходимых для перемещения по лабиринту, не натыкаясь на стены лабиринта.

Мы уже возвращались к роботу ранее в этой главе при обсуждении теории нечеткой логики. Теперь реализуем и тестируем контроллер нечеткой логики для робота с использованием библиотеки eFLL на примере «Arduino_advanced_sample» в качестве шаблонного образца. Мы тестируем робота, используя смоделированные входы ИК-датчиков и выходы управления двигателем, как показано на рис. 5.10. Обратите внимание, что мы реализуем алгоритм нечеткого управления на классическом Arduino UNO R3. Этот процессор содержит 32 Кбайта флеш-памяти и 2 Кбайта ОЗУ.

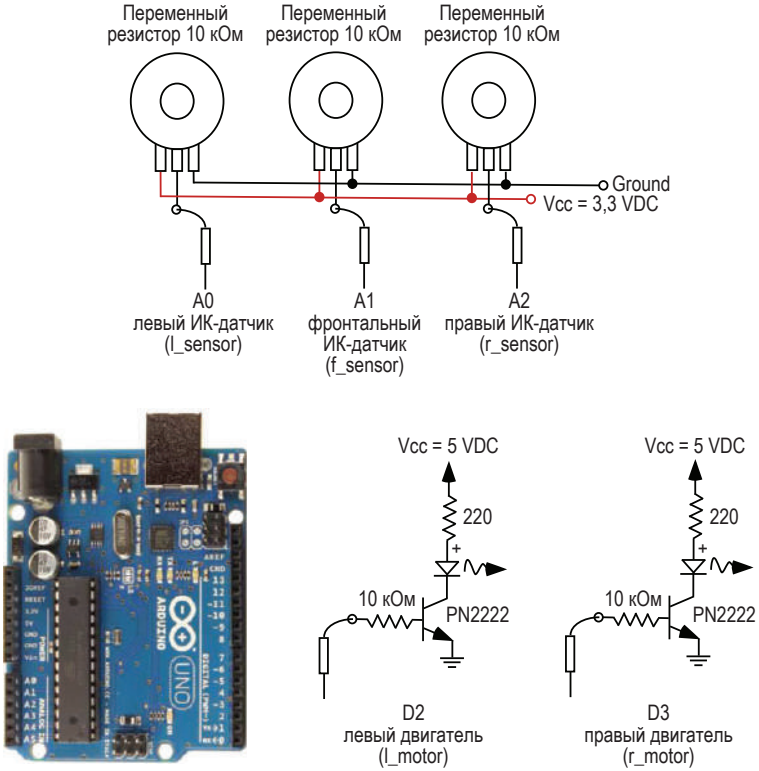


Рис. 5.10. Симулятор входов и выходов

При разработке нечеткого примера мы ограничиваем разработку скетча двумя (фронтальным и правым) ИК-датчиками. Использование трех ИК-датчиков оставляем в качестве домашнего

задания. Прежде чем продолжить, ознакомьтесь с разработкой функций измерения входных параметров, выходными функциями принадлежности и реализацией правил, представленными ранее в этой главе.

Как показано на рис. 5.3, *d*, девять правил, разработанных для управления роботом, включают:

- правило 1: IF (правый датчик вплотную) AND (фронтальный датчик вплотную) THEN (левый двигатель медленно) AND (правый двигатель средне);
- правило 2: IF (правый датчик вплотную) AND (фронтальный датчик близко) THEN (левый двигатель медленно) AND (правый двигатель медленно);
- правило 3: IF (правый датчик вплотную) AND (фронтальный датчик далеко) THEN (левый двигатель быстро) AND (правый двигатель быстро);
- правило 4: IF (правый датчик близко) AND (фронтальный датчик вплотную) THEN (левый двигатель медленно) AND (правый двигатель медленно);
- правило 5: IF (правый датчик близко) AND (фронтальный датчик близко) THEN (левый двигатель медленно) AND (правый двигатель медленно);
- правило 6: IF (правый датчик близко) AND (фронтальный датчик далеко) THEN (левый двигатель быстро) AND (правый двигатель быстро);
- правило 7: IF (правый датчик далеко) AND (фронтальный датчик вплотную) THEN (левый двигатель средне) AND (правый двигатель медленно);
- правило 8: IF (правый датчик далеко) AND (фронтальный датчик близко) THEN (левый двигатель медленно) AND (правый двигатель медленно);
- правило 9: IF (правый датчик далеко) AND (фронтальный датчик далеко) THEN (левый двигатель быстро) AND (правый двигатель быстро).

Для корректной работы скетча правила с одинаковыми следствиями следует объединить в одно правило, как показано на рис. 5.3, *∂*. Это позволяет управлять роботом с помощью всего четырех правил. Правила можно еще больше упростить, внимательно изучив таблицу правил. Например, когда фронтальный датчик сообщает

«далеко», не имеет значения, какое значение сообщает правый датчик, поскольку результирующее действие робота одинаково. В результате получается упрощенный набор правил:

- правило 1: IF (правый датчик вплотную) AND (фронтальный датчик вплотную) THEN (левый двигатель медленно) AND (правый двигатель средне);
- правило 2: IF (правый датчик близко) AND (фронтальный датчик вплотную) THEN (левый двигатель медленно) AND (правый двигатель медленно);
- правило 3: IF (фронтальный датчик далеко) THEN (левый двигатель быстро) AND (правый двигатель быстро);
- правило 4: IF (правый датчик далеко) AND (фронтальный датчик вплотную) THEN (левый двигатель средне) AND (правый двигатель медленно).

Ниже представлен скетч, реализующий алгоритм нечеткого управления.

```
//*************************************************************
//two_sensor_fuzzy_robot: provides fuzzy control for a two
//sensor robot (front and right facing) for autonomous maze
//navigation.
//
//eFLL example arduino_advanced_sample (Alves) used as
//template for development.
//
//Fuzzy control inputs
//- left IR sensor A0 - not used in this sketch
//- front IR sensor A1
//- right IR sensor A2
//
//Fuzzy control outputs
//- left motor 9
//- right motor 10
//*************************************************************
#include <Fuzzy.h>
#include <FuzzyComposition.h>
#include <FuzzyInput.h>
#include <FuzzyIO.h>
#include <FuzzyOutput.h>
#include <FuzzyRule.h>
#include <FuzzyRuleAntecedent.h>
#include <FuzzyRuleConsequent.h>
#include <FuzzySet.h>
//define robot conncetions
#define left_IR_sensor A0     //pin 4
```

```
#define front_IR_sensor A1    //pin 5
#define right_IR_sensor A2    //pin 6
#define left_motor 9
#define right_motor 10
//indicate use of simulator (0) or random generator (1)
bool simulator = 1;
int input1, input2;
//Declare membership functions globally for pertinence access.
//Instantiate all objects in loop()
// Fuzzy
Fuzzy *fuzzy = new Fuzzy();
//FuzzyInput - right IR sensor
FuzzySet *r_far = new FuzzySet( 0, 0, 75,100);
FuzzySet *r_near = new FuzzySet( 75,100,150,175);
FuzzySet *r_close = new FuzzySet(150,175,255,255);
//FuzzyInput - front IR sensor
FuzzySet *f_far = new FuzzySet( 0, 0, 75,100);
FuzzySet *f_near = new FuzzySet( 75,100,150,175);
FuzzySet *f_close = new FuzzySet(150,175,255,255);
//FuzzyOutput - left motor
FuzzySet *l_slow = new FuzzySet( 0, 0, 75,100);
FuzzySet *l_medium= new FuzzySet( 75,100,150,175);
FuzzySet *l_fast = new FuzzySet(150,175,255,255);
//FuzzyOutput - right motor
FuzzySet *r_slow = new FuzzySet( 0, 0, 75,100);
FuzzySet *r_medium= new FuzzySet( 75,100,150,175);
FuzzySet *r_fast = new FuzzySet(150,175,255,255);
//sensor readings
//int left_IR_sensor_value;   //left IR sensor variable
int front_IR_sensor_value;    //front IR sensor variable
int right_IR_sensor_value;    //right IR sensor variable
void setup()
{
//Set the Serial output
  Serial.begin(9600);
//Set a random seed
  randomSeed(analogRead(0));
//FuzzyInput - right IR sensor r_sensor
  FuzzyInput *r_sensor = new FuzzyInput(1);
  r_sensor->addFuzzySet(r_far);
  r_sensor->addFuzzySet(r_near);
  r_sensor->addFuzzySet(r_close);
  fuzzy->addFuzzyInput(r_sensor);
//FuzzyInput - front IR sensor f_sensor
  FuzzyInput *f_sensor = new FuzzyInput(2);
  f_sensor->addFuzzySet(f_far);
  f_sensor->addFuzzySet(f_near);
  f_sensor->addFuzzySet(f_close);
  fuzzy->addFuzzyInput(f_sensor);
//FuzzyOutput - left motor
  FuzzyOutput *l_motor = new FuzzyOutput(1);
```

```
  l_motor->addFuzzySet(l_slow);
  l_motor->addFuzzySet(l_medium);
  l_motor->addFuzzySet(l_fast);
  fuzzy->addFuzzyOutput(l_motor);
//FuzzyOutput - right motor
  FuzzyOutput *r_motor = new FuzzyOutput(2);
  r_motor->addFuzzySet(r_slow);
  r_motor->addFuzzySet(r_medium);
  r_motor->addFuzzySet(r_fast);
  fuzzy->addFuzzyOutput(r_motor);
//Fuzzy Rule Set
//Building Fuzzy Rule 1
//Antecedent
  FuzzyRuleAntecedent *rsensorcloseAndfsensorclose = new    FuzzyRuleAntecedent();
rsensorcloseAndfsensorclose->joinWithAND(r_close, f_close);
//Consequent
  FuzzyRuleConsequent *thenlmotorslowAndrmotormed = new FuzzyRuleConsequent();
  thenlmotorslowAndrmotormed->addOutput(l_slow);
  thenlmotorslowAndrmotormed->addOutput(r_medium);
//Assemble Rule
  FuzzyRule *fuzzyRule1 = new FuzzyRule(1,rsensorcloseAndfsensorclose,
thenlmotorslowAndrmotormed);
  fuzzy->addFuzzyRule(fuzzyRule1);
//Building Fuzzy Rule 2
//Antecedent
  FuzzyRuleAntecedent *fsensornear = new FuzzyRuleAntecedent();
fsensornear->joinSingle(f_near);
  FuzzyRuleAntecedent *rsensornearAndfsensorclose = new FuzzyRuleAntecedent();
  rsensornearAndfsensorclose->joinWithAND(r_near, f_close);
  FuzzyRuleAntecedent *ifrsensornearAndfsensorcloseOrfsensornear =
new FuzzyRuleAntecedent();
  ifrsensornearAndfsensorcloseOrfsensornear-> joinWithOR(rsensornearAndfsensorclose,
fsensornear);
//Consequent
  FuzzyRuleConsequent *thenlmotorslowAndrmotorslow = new FuzzyRuleConsequent();
  thenlmotorslowAndrmotorslow->addOutput(l_slow);
  thenlmotorslowAndrmotorslow->addOutput(r_slow);
//Assemble Rule
  FuzzyRule *fuzzyRule2 = new FuzzyRule(2,ifrsensornearAndfsensorcloseOrfsensornear,
  thenlmotorslowAndrmotorslow);
  fuzzy->addFuzzyRule(fuzzyRule2);
//Building Fuzzy Rule 3
//Antecedent
  FuzzyRuleAntecedent *fsensorfar = new FuzzyRuleAntecedent();
  fsensorfar->joinSingle(f_far);
//Consequent
  FuzzyRuleConsequent *thenlmotorfastAndrmotorfast = new FuzzyRuleConsequent();
  thenlmotorfastAndrmotorfast->addOutput(l_fast);
  thenlmotorfastAndrmotorfast->addOutput(r_fast);
//Assemble Rule
  FuzzyRule *fuzzyRule3 = new FuzzyRule(3,fsensorfar,thenlmotorfastAndrmotorfast);
```

```
  fuzzy->addFuzzyRule(fuzzyRule3);
//Building Fuzzy Rule 4
//Antecedent
  FuzzyRuleAntecedent *rsensorfarAndfsensorclose = new FuzzyRuleAntecedent();
  rsensorfarAndfsensorclose->joinWithAND(r_far, f_close);
//Consequent
  FuzzyRuleConsequent *thenlmotormedAndrmotorslow = new FuzzyRuleConsequent();
  thenlmotormedAndrmotorslow->addOutput(l_medium);
  thenlmotormedAndrmotorslow->addOutput(r_slow);
//Assemble Rule
  FuzzyRule *fuzzyRule4 = new FuzzyRule(4,rsensorfarAndfsensorclose,
thenlmotormedAndrmotorslow);
  fuzzy->addFuzzyRule(fuzzyRule4);
//motor pin configuration
  pinMode(left_motor, OUTPUT);
  pinMode(right_motor, OUTPUT);
}
void loop()
{
  if (!simulator)                                          //use random generator for input
  {
//get random entrance value for right and front IR sensors
    input1 = random(0, 255);
    input2 = random(0, 255);
  }
  else
  { //use potentiometer input
//read analog output from IR sensors
//left_IR_sensor_value = analogRead(left_IR_sensor);       //pin 4 A0
    front_IR_sensor_value = analogRead(front_IR_sensor);   //pin 5 A1
    right_IR_sensor_value = analogRead(right_IR_sensor);   //pin 6 A2
//remap from 1023 to 255 scale
    input1 = map(right_IR_sensor_value, 0, 1023, 0, 255);
    input2 = map(front_IR_sensor_value, 0, 1023, 0, 255);
  }
  Serial.print("R sensor: ");
  Serial.print(input1);
  Serial.print("\t, F sensor: ");
  Serial.println(input2);
//Serial.println(" ");
  fuzzy->setInput(1, input1);
  fuzzy->setInput(2, input2);
  fuzzy->fuzzify();
//right sensor
//Serial.println("R Input: ");
  Serial.print("R sensor: close-> ");
  Serial.print(r_close->getPertinence());
  Serial.print(", near-> ");
  Serial.print(r_near->getPertinence());
  Serial.print(", far-> ");
  Serial.println(r_far->getPertinence());
```

```
//Serial.println(" ");
//front sensor
//Serial.println("F Input: ");
  Serial.print("F sensor: close-> ");
  Serial.print(f_close->getPertinence());
  Serial.print(", near-> ");
  Serial.print(f_near->getPertinence());
  Serial.print(", far-> ");
  Serial.println(f_far->getPertinence());
//Serial.println(" ");
  float output1 = fuzzy->defuzzify(1);      //left motor
  float output2 = fuzzy->defuzzify(2);      //right motor
//left motor
//Serial.println("L motor output: ");
  Serial.print("L motor: slow-> ");
  Serial.print(l_slow->getPertinence());
  Serial.print(", Medium-> ");
  Serial.print(l_medium->getPertinence());
  Serial.print(", Fast-> ");
  Serial.println(l_fast->getPertinence());
//Serial.println(" ");
//right motor
//Serial.println("R motor output: ");
  Serial.print("R motor: slow-> ");
  Serial.print(r_slow->getPertinence());
  Serial.print(", Medium-> ");
  Serial.print(r_medium->getPertinence());
  Serial.print(", Fast-> ");
  Serial.println(r_fast->getPertinence());
//Serial.println(" ");
  if(simulator)
  {
//motor control - left motor
    int output1_int = (int)(output1);       //LED on pin 9
    analogWrite(left_motor, output1_int);
    Serial.println("L motor output: ");
    Serial.print(output1_int);
    Serial.println(" ");
//motor control - right motor
    int output2_int = (int)(output2);       //LED on pin 10
    analogWrite(right_motor, output2_int);
    Serial.println("R motor output: ");
    Serial.print(output2_int);
    Serial.println(" ");
    Serial.println(" ");
  }
//wait 2 seconds
  delay(2000);
}
//*************************************************************
```

Чтобы протестировать нечеткий контроллер робота, используйте симулятор для проверки всех комбинаций входных сигналов датчиков, представленных в таблице правил на рис. 5.3, *г*. Если базовый контроллер будет работать правильно, рассмотрите возможность добавления левого датчика для обнаружения второй стены лабиринта.

5.5. Выводы

В этой главе описано, как управлять процессом с помощью методов нечеткой логики. Мы познакомились с нечеткой логикой, допускающей несколько уровней истины между логической единицей и нулем, и обнаружили, что многие реальные проблемы управления поддаются реализации с помощью нечеткой логики. Затем более подробно исследуется алгоритм действий контроллеров нечеткой логики. Мы начали с краткого обзора ключевых концепций проектирования контроллеров нечеткой логики. Затем познакомились со встраиваемой библиотекой нечеткой логики Arduino (eFLL) и примерами, разработанными командой исследовательской группы робототехники Государственного университета Пиауи в Терсини, Пиауи, Бразилия. Примеры рассмотрены довольно подробно. А также мы оснастили ранее рассмотренного робота Dagu Magician системой управления с нечеткой логикой на основе библиотеки eFLL.

5.6. Задания

1. Своими словами сравните и противопоставьте традиционный подход к проектированию управляющего контроллера и метод нечеткой логики.
2. Представьте блок-схему процесса проектирования нечеткого регулятора. Кратко опишите, какие действия необходимы на каждом этапе.
3. Проиллюстрируйте процесс проектирования нечеткого управления на собственном примере.
4. Как входные функции принадлежности получаются из четких выходных сигналов датчиков?

5. Опишите различные типы функций-трапеций, доступные для определения входных и выходных функций принадлежности.
6. Как определяются выходные функции для конкретного приложения?
7. Опишите использование операторов AND (И) и OR (ИЛИ) при разработке правил.
8. Какова базовая конфигурация правила нечеткой логики?
9. Что такое условия и следствия? Как они используются при разработке правил нечеткой логики?
10. Опишите своими словами, как нечеткие выходные данные преобразуются в четкие системные выходные данные.
11. В разделе «Применение» этой главы мы представили конструкцию робота, следующего по лабиринту с двумя датчиками. Пересмотрите конструкцию, включив в нее третий левый датчик.

Источники

1. A. J. Alves, R. Lira, M. Lemos, D.S. Kridi, and K. Leal, Arduino Embedded Fuzzy Logic Library (eFLL), Robotic Research Group at the State University of Piaui, Tersini, Piaui, Brazil.
2. A. J. Alves, eFLL – A Fuzzy Library for Arduino and Embedded Systems, https://www.blog.zerokol.com.
3. T. Jiang and Y. Li, Generalized Defuzzification Strategies and Their Parameter Learning Procedures, IEEE Transactions on Fuzzy Systems, Vol. 4, No. 1, February 1996, 64–71.
4. A. D. Kulkarni, Computer Vision and Fuzzy – Neural Systems, Prentice Hall, 2001.
5. C. C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 2, March/April 1990, pp. 404–418.
6. C. C. Lee, Fuzzy Logic in Control Systems: Fuzzy Logic Controller – Part I, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 20, No. 2, March/April 1990, pp. 419–435.
7. E. H. Mamdani and S. Assilian, An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, Int. J. Man – Machine Studies, (1975), 1–13.
8. D. J. Pack and S. F. Barrett, 68HC12Microcontroller Theory and Application, PrenticeHall, 2002.

9. D. T. Pham and M. Castellani, Action aggregation and defuzzification in Mamdani – type fuzzy systems, Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 2002, 747–759.
10. T. A. Runkler, Selection of Appropriate DefuzzificationMethods Using Application Specific Properties, IEEE Transactions on Fuzzy Systems, Vol. 5, No. 1, February 1997, 72–79.
11. L. A. Zadeh, Fuzzy Sets, Information and Control 8, 338–353, 1965.

ГЛАВА

6

Нейронные сети

Прочитав эту главу, читатель должен уметь:

- описать и зарисовать биологический нейрон;
- определить и предоставить вспомогательные уравнения для персептронной модели нейрона;
- смоделировать работу одного персептрона;
- использовать общую модель персептрона для линейной классификации объектов на две разные категории;
- смоделировать работу сети с несколькими персептронами;
- использовать модель нескольких персептронов для линейной классификации объектов по различным категориям;
- определить и предоставить вспомогательные уравнения для модели одного нейрона;
- использовать модель одного нейрона для сборки многослойной искусственной нейронной сети;
- описать концепцию обратного распространения ошибки;
- использовать трехслойную сеть прямого распространения с обратным распространением ошибки для классификации объектов по различным категориям;
- перечислить улучшения искусственной нейронной сети для улучшения сходимости моделей;
- описать передовые программные инструменты для разработки и внедрения глубоких нейронных сетей.

6.1. Обзор

В этой главе мы исследуем концепцию нейронов и нейронных моделей для решения реальных задач. Начнем с краткого описания биологического нейрона и исследуем модель нейрона, названную персептроном, разработанную Фрэнком Розенблаттом в 1959 году. Мы используем модель одиночного персептрона для разделения объектов на две категории и расширим модель, включив в нее дополнительные персептроны для разделения объектов на несколько категорий. После этого познакомимся с единичным нейроном, а затем изучим концепцию обратного распространения ошибки (backpropagation) и разработаем трехслойную сеть прямого распространения с обратным распространением ошибки. Попутно разработаем скетчи Arduino для этих случаев.

6.2. Биологический нейрон

Наш мозг состоит из множества нейронов. Нейроны работают вместе, чтобы помочь нам учиться, запоминать, связывать сложные идеи, выполнять сложные задачи и многое другое. Целью ученых и инженеров было понять работу и взаимодействие нейронов, смоделировать их поведение и использовать модели для решения сложных задач на компьютерах. Некоторым моделям требуется огромная вычислительная мощность, значительно превышающая возможности микроконтроллеров[1]. Однако есть много задач, которые можно легко выполнить на Arduino Nano 33 BLE Sense или даже классическом Arduino UNO R3 с небольшим энергопотреблением. В этой главе мы сосредоточимся на этих приложениях.

Схема одиночного биологического нейрона представлена на рис. 6.1. Основная вычислительная единица нейрона находится внутри тела клетки (сомы). Нейрон собирает информацию от соседних нейронов через сеть входных датчиков, называемых дендритами. Входная информация собирается и обрабатывается

[1] Успех первых версий известного ИИ-продукта ChartGPT был достигнут при обучении с помощью облачного суперкомпьютера Azure AI, содержавшего 285 тысяч обычных и 10 тысяч ядер графических процессоров. – *Прим. перев.*

сомой. Если достигается определенный уровень накопленной входной информации, нейрон срабатывает и передает электрический сигнал по своему аксону. Аксон покрыт амиелиновой оболочкой, способствующей передаче сигнала. Когда электрический сигнал достигает концевых волокон аксона, высвобождается химический передатчик, который передает информацию другим близлежащим нейронам [7][1].

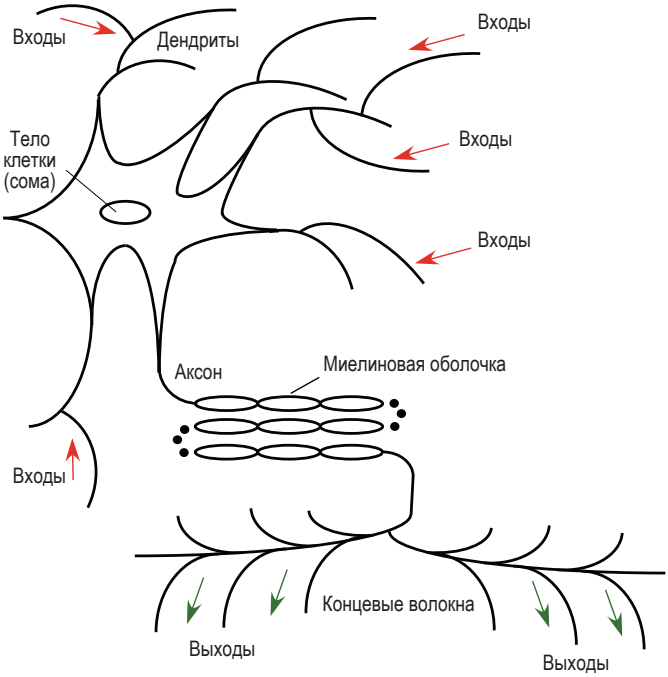


Рис. 6.1. Биологический нейрон [7]

6.3. Персептрон

В 1959 году Фрэнк Розенблатт на основе биологического нейрона разработал модель одиночного персептрона, показанную на

[1] См. подробное описание работы нейронов на русском языке в публикации Алексея Редозубова из серии статей «Логика мышления» https://habr.com/ru/articles/214109. – *Прим. перев.*

рис. 6.2. Математическое описание здесь основано на превосходной разработке, представленной Кулкарни (A. D. Kulkarni) [4].

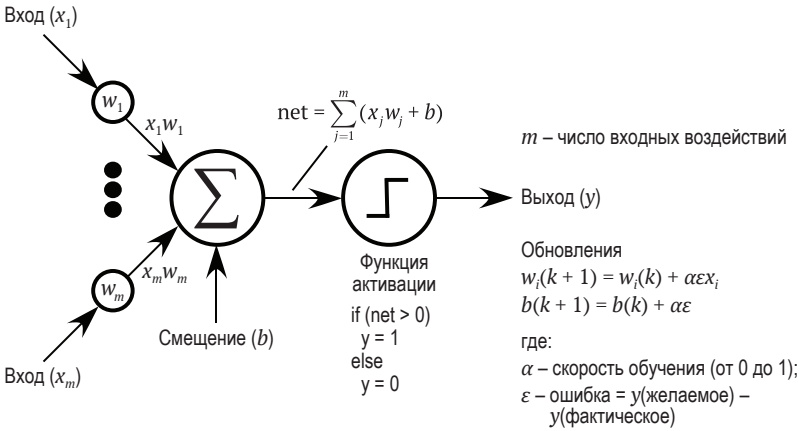


Рис. 6.2. Модель одиночного персептрона [4]

Модель персептрона предусматривает входные данные (x_i), которые умножаются на веса (w_i), специфичные для каждого входного сигнала. Взвешенные входные данные вместе со смещением (*bias*) суммируются, чтобы получить чистый отклик (*net*), а затем передаются в функцию активации.

Если значение отклика *net* превышает заданное значение порога активации, персептрон срабатывает и выдает выходное значение (y); в противном случае выходное значение равно 0. В показанном примере пороговое значение активации равно 0. Если значение *net* больше 0, персептрон обеспечивает выходной сигнал 1; в противном случае выходной сигнал равен 0. Можно использовать другие функции активации.

Одиночный персептрон можно научить сортировать объекты на две линейные категории. Что это значит? Если бы вам нужно было нанести объекты на двухмерный график, вы могли бы провести прямую линию, чтобы разделить две разные категории объектов. Существует множество реальных технических задач, где это было бы полезно. Например, в следующем примере мы рассмотрим систему сортировки помидоров, позволяющую выделять самые большие и красные плоды. Кроме того, важно досконально понимать работу персептрона, поскольку она формирует основу для более сложных моделей на основе нейронов.

6.3.1. Обучение модели персептрона

Чтобы научить персептрон распределять объекты по двум различным категориям, используется обучающий набор данных. Обучающий набор содержит взаимосвязь между набором заданных значений входных переменных и желаемой выходной категорией для этого набора. Обучающий набор может содержать несколько записей желаемых пар входа/выхода.

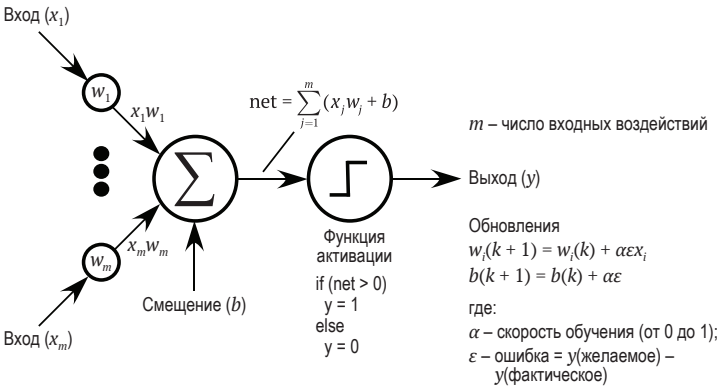
Чтобы инициировать последовательность обучения, веса *w* модели персептрона и смещение *b* изначально устанавливаются равными нулю или некоторым небольшим случайным значениям. Затем входные данные x_i умножаются на входные веса w_i. Взвешенные входные данные суммируются со смещением для определения значения *net*, а потом передаются в функцию активации. Функция активации генерирует соответствующий выходной сигнал (1 или 0), если значение *net* превысило установленный порог (в данном случае равный 0).

Выходные данные персептрона *y* сравниваются с желаемыми выходными данными, предоставляемыми обучающим набором. Затем вычисляется ошибка ε. Ошибка представляет собой разницу между желаемым выходным сигналом обучающего набора и фактическим выходным сигналом, полученным от персептрона. Затем веса и смещение обновляются с использованием уравнений, представленных на рис. 6.2. В дополнение к величине ошибки ε уравнения обновления также предоставляют величину скорости обучения (α). Для скорости обучения установлено значение от 0 до 1. Чем больше значение, тем более резкие изменения весов *w*. Меньшее значение α может потребовать дополнительного времени вычислений, но потенциально дает лучший результат сходимости модели.

Персептрон теперь обрабатывает вторую пару входных/выходных данных из обучающего набора, используя обновленные значения весов и смещений. Этот процесс продолжается на протяжении всего обучающего набора, пока модель не сойдется. Это произойдет тогда, когда ошибка для каждой записи в обучающем наборе станет равна нулю или достигнет желаемого целевого значения. Цикл прохождения всего обучающего набора называется эпохой (epoch). Процесс может потребовать нескольких итераций применения обучающего набора к модели персептрона. В первом примере далее для сходимости потребовалось два последовательных применения обучающего набора. Второй пример потребовал 1500 итеративных применений обучающего набора.

Как только модель достигнет целевого значения ошибки для всего обучающего набора, полученные значения веса и смещения

можно использовать для построения линии разделения двух категорий объектов. В целом процесс обучения проиллюстрирован на рис. 6.3. После обучения модели ее можно использовать для классификации новых входных данных, отсутствующих в исходном обучающем наборе.



Обучение:

1. Задаем обучающий набор в виде желаемых пар входа/выхода.
2. Выбираем скорость обучения α (`learning_rate`) от 0 до 1. Чем больше значение, тем более резкие изменения веса. Меньшее значение требует дополнительного времени вычислений, но может дать лучшие результаты.
3. Инициализируйте веса (`w`) и смещение (`bias`) нулевыми или небольшими случайными значениями.
4. Рассчитайте значение `net` для пары вход/выход обучающего набора.
5. Примените это значение к функции активации и определите фактический результат для пары вход/выход обучающего набора.
6. Рассчитайте погрешность ε (`sq_error`).
7. Обновите значения весов и смещений.
8. Если закончился обучающий набор (эпоха),
   вычислите среднеквадратичную ошибку (`MSE`) обучающего набора;
   иначе
   продолжить обучение с шага 4.
9. Если `MSE` обучающего набора равен нулю или достиг желаемой величины,
   сохраните значения веса и смещения;
   иначе
   продолжить обучение с шага 4.

Выполнение (классификация):

1. Получите сохраненные веса и смещение в результате обучения.
2. Примените новую пару входа/выхода сверх обучающего набора.
3. Рассчитайте величины `net` и выхода.
4. Поместите входную пару в соответствующую категорию.

Рис. 6.3. Обучение модели одиночного персептрона [4]

Пример **В первом примере персептрона мы воспроизводим результаты, предоставленные Дэном (Dan) в статье «Объяснение однослойного персептрона» [3]. Персептрон состоит из двух входов, смещения и одного выхода. Веса и смещение изначально установлены на ноль. После двух эпох среднеквадратическая ошибка (MSE) выходной величины равна нулю. Полученные веса и смещение можно использовать для разделения пар входа/выхода на две разные категории, как показано на рис. 6.4.**

X1	X2	W1	W2	Bias	Desired Out	NET	Actual Out	Error	W1 update	W2 update	Bias Update	Error^2	MSE
2.0	4.0	0	0	0	1	0	0	1	0.2	0.4	0.1	1	
1.0	3.0	0.2	0.4	0.1	1	1.5	1	0	0.2	0.4	0.1	0	
2.0	1.0	0.2	0.4	0.1	0	0.9	1	-1	0	0.3	0	1	
1.0	0.6	0	0.3	0	0	0.18	1	-1	-0.1	0.24	-0.1	1	0.75
2.0	4.0	-0.1	0.24	-0.1	1	0.06	1	0	-0.1	0.24	-0.1	0	
1.0	3.0	-0.1	0.24	-0.1	1	0.52	1	0	-0.1	0.24	-0.1	0	
2.0	1.0	-0.1	0.24	-0.1	0	-0.06	0	0	-0.1	0.24	-0.1	0	
1.0	0.6	-0.1	0.24	-0.1	0	-0.066	0	0	-0.1	0.24	-0.1	0	0

Alpha	0.1

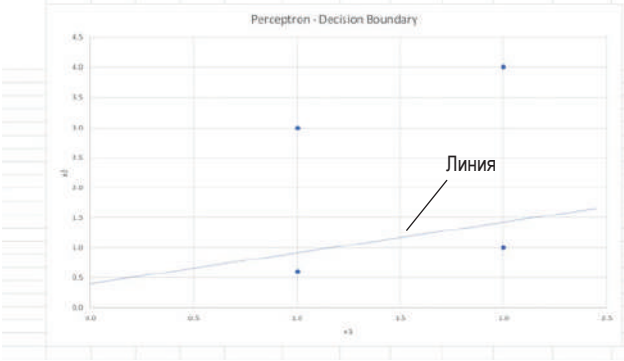


Рис. 6.4. Таблица Excel обучения модели одиночного персептрона [3]

Линию, разделяющую объекты категории 1 и категории 2, можно построить на основе линейного уравнения, полученного из следующих вычислений:

$$x_1 w_1 + x_2 w_2 + b = 0;$$

$$x_1(-0{,}1) + x_2(0{,}24) + (-0{,}1) = 0;$$

$$x_2 = (0{,}1)x_1/(0{,}24) + (0{,}1)/(0{,}24);$$

$$y = mx + b;$$

$$y = 0{,}42x + 0{,}42.$$

Следующий скетч Arduino выполняет эту обучающую задачу. UML-диаграмма действий для скетча представлена на рис. 6.5.

В этом примере скетч запускается на Arduino UNO R3. Итоговый результат выполнения скетча представлен на рис. 6.6. После обучения персептрона его можно использовать для классификации входных данных за пределами исходного набора данных. Мы назовем такое применение модели Run mode (режимом выполнения) и обсудим этот режим в следующем разделе.

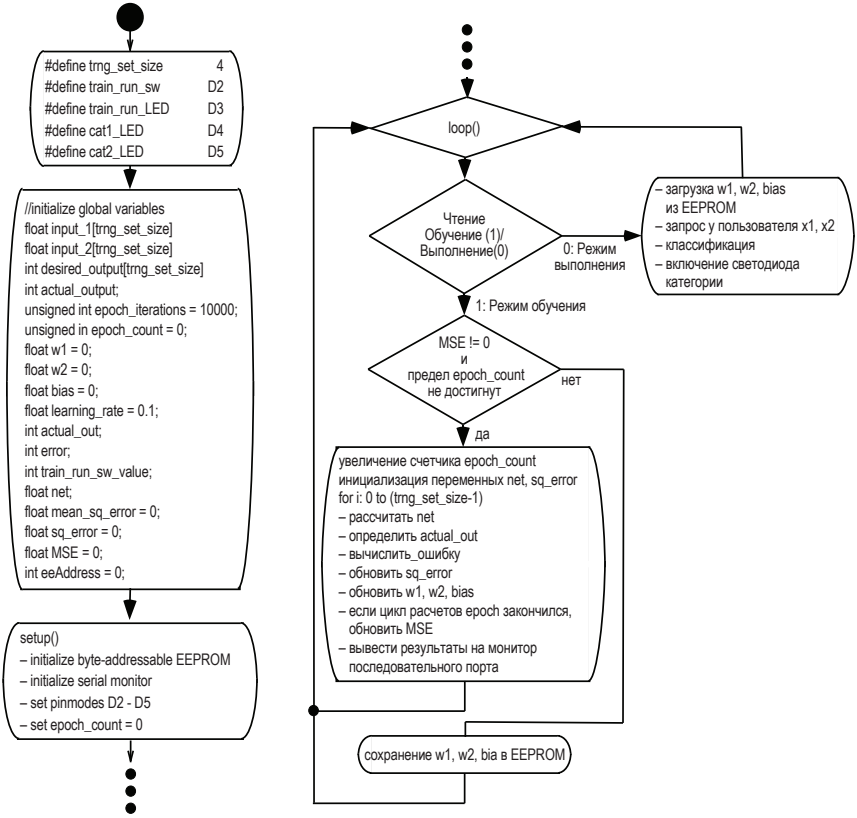


Рис. 6.5. UML-диаграмма обучения/выполнения персептрона (см. также раздел 6.3.2)

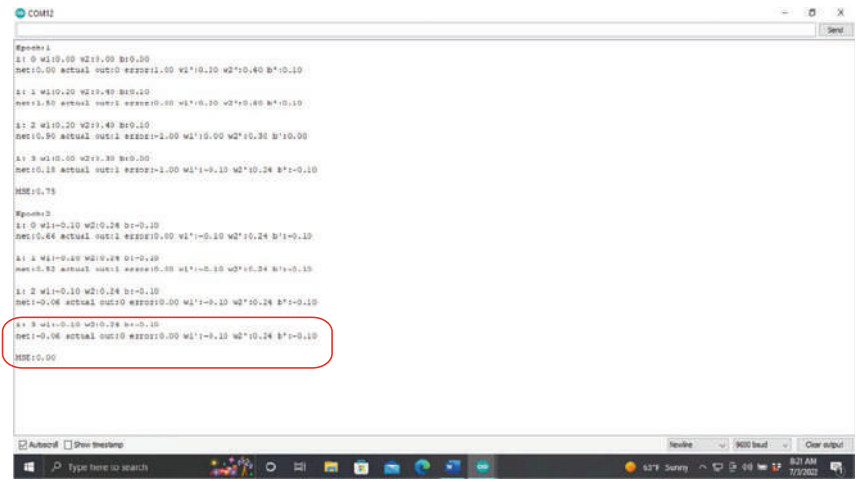


Рис. 6.6. Вывод скетча обучения модели одиночного персептрона

```
//***************************************************************
//perceptron_2in_1out_train_run
//Sketch has two modes of operation selected via a DIP switch.
//Sketch should be run first in Train Mode(1) before Run Mode(0).
//attached to input D2:
// - D2 = HIGH, selects training mode
// - D2 = LOW, selects run mode
//
//Other pins used:
// - D3: LED to indicate mode Train(1)/Run(0)
// - D4: LED to indicate inputs placed in Category 1
// - D5: LED to indicate inputs placed in Category 2
//***************************************************************
#include<EEPROM.h>
//define
#define trng_set_size  4    //entries in training set
#define train_run_sw   2    //switch input train(1)/run(0)
#define train_run_LED  3    //LED for train(1)/run(0)
#define cat1_LED       4    //LED for category 1
#define cat2_LED       5    //LED for category 2
//global
float input_1[trng_set_size] = {2.0, 1.0, 2.0, 1.0};
float input_2[trng_set_size] = {4.0, 3.0, 1.0, 0.6};
int desired_output[trng_set_size] = {1, 1, 0, 0};
int actual_output;
unsigned int epoch_iterations = 10000;
unsigned int epoch_count = 0;
float w1 = 0;                 //set initial weight of w1
float w2 = 0;                 //set initial weight of w2
```

```
float bias = 0;               //set inital bias value
float learning_rate = 0.1;    //set 0 to 1
float x1, x2;
int actual_out;               //output from perceptron
float error;                  //desired - actual output
int train_run_sw_value;       //switch input train(1)/run(0)
float net;                    //numerical out from perceptron
float sq_error = 0;           //squared error
float MSE = 1;                //mean squared error
int eeAddress = 0;
void setup()
{
  Serial.begin(9600);         //Set the Serial output
//set pin modes
  pinMode(train_run_sw, INPUT);
pinMode(train_run_LED, OUTPUT);
pinMode(cat1_LED, OUTPUT);
pinMode(cat2_LED, OUTPUT);
epoch_count = 0;              //reset epoch count
}
void loop()
{
  int i;
//Reset LEDs
  digitalWrite(train_run_LED, LOW);
  digitalWrite(cat1_LED, LOW);
  digitalWrite(cat2_LED, LOW);
//Read Train(1)/Run Switch (0)
//Serial.print("Read Train(1)/Run Switch(0):");
  train_run_sw_value = digitalRead(train_run_sw);
  if(train_run_sw_value == 1)          //training mode
  {
    digitalWrite(train_run_LED, HIGH);
//calculate response until MSE = 0
//or max count reached
    if((MSE > 0)&&(epoch_count <=epoch_iterations))
    {
      epoch_count = epoch_count + 1;   //epoch count
      Serial.print("Epoch:");
      Serial.println(epoch_count);
      net = 0;                         //reset net
      sq_error = 0;                    //reset sq error
      for(i=0; i < trng_set_size; i++)
      {
        Serial.print("i: ");
        Serial.print(i);
        net = (input_1[i]*w1) + (input_2[i]*w2) + bias;
        if(net > 0)
        {
          actual_out = 1;
        }
```

```
      else
      {
        actual_out = 0;
      }
      Serial.print(" w1:");
      Serial.print(w1);
      Serial.print(" w2:");
      Serial.print(w2);
      Serial.print(" b:");
      Serial.println(bias);
      error = desired_output[i] - actual_out;
      sq_error = sq_error + (error * error);
      w1 = w1 + (learning_rate * error * input_1[i]);
      w2 = w2 + (learning_rate * error * input_2[i]);
      bias = bias + (learning_rate * error);
      Serial.print("net:");
      Serial.print(net);
      Serial.print(" actual out:");
      Serial.print(actual_out);
      Serial.print(" error:");
      Serial.print(error);
      Serial.print(" w1':");
      Serial.print(w1);
      Serial.print(" w2':");
      Serial.print(w2);
      Serial.print(" b':");
      Serial.print(bias);
      Serial.println(" ");
      if(i == (trng_set_size-1))      //end of Epoch
      {
        MSE = sq_error/trng_set_size;
        Serial.println(" ");
        Serial.print("MSE:");
        Serial.print(MSE);
        Serial.println(" ");
      }
      Serial.println(" ");
    }//end for
  }//if (MSE)
  if(MSE == 0)                        //store weights and bias to EEPROM
  {
    eeAddress = 0;
    EEPROM.put(eeAddress, w1);        //store weight 1
    eeAddress = eeAddress + sizeof(float);
    EEPROM.put(eeAddress, w2);        //store weight 2
    eeAddress = eeAddress + sizeof(float);
    EEPROM.put(eeAddress, bias);      //store bias
  }
}
else                                  //run mode
{
```

```
    Serial.println("else...");
    digitalWrite(train_run_LED, LOW);
//get values from EEPROM
    eeAddress = 0;
    EEPROM.get(eeAddress, w1);           //store weight 1
    eeAddress = eeAddress + sizeof(float);
    EEPROM.get(eeAddress, w2);           //store weight 2
    eeAddress = eeAddress + sizeof(float);
    EEPROM.get(eeAddress, bias);         //store bias
//print results
    Serial.println(" ");
    Serial.print(" w1':");
    Serial.print(w1);
    Serial.print(" w2':");
    Serial.print(w2);
    Serial.print(" b':");
    Serial.print(bias);
    Serial.println(" ");
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x1 input value from user via serial monitor
    Serial.println("Insert new value of x1: [send]");
    while(Serial.available()==0){}       //wait for user input data
    x1 = Serial.parseFloat();
    Serial.println(" ");
    Serial.print("x1:");
    Serial.println(x1);
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x input value from user via serial monitor
    Serial.print("Insert new value of x2: [send]");
    delay(5000);
    while(Serial.available()==0){}       //wait for user input data
    x2 = Serial.parseFloat();
    Serial.println(" ");
    Serial.print("x2:");
    Serial.println(x2);
//reset LEDs
    digitalWrite(cat1_LED, LOW);
    digitalWrite(cat2_LED, LOW);
//process new input and assign to appropriate category
//illuminate appropriate LED
    net = (x1*w1) + (x2*w2) + bias;
    if(net > 0)
    {
```

```
//category 1
      Serial.println("Category 1");
      digitalWrite(cat1_LED, HIGH);
      digitalWrite(cat2_LED, LOW);
    }
    else
    {
//category 2
      Serial.println("Category 2");
      digitalWrite(cat1_LED, LOW);
      digitalWrite(cat2_LED, HIGH);
    }
    delay(5000);
  }
}
//*************************************************************
```

6.3.2. Режим выполнения одиночного персептрона

В предыдущем скетче режим выполнения (Run mode) начинается с оператора `else`. Для реализации режима выполнения отдельно от обучения в скетч потребуется внести следующие дополнения (см. также UML-диаграмму на рис. 6.5):

- оснастите Arduino UNO R3 внешним переключателем для выбора режима «Обучение» (1) / «Выполнение» (0);
- обеспечьте отдельный светодиод для индикации текущего режима: «Обучение» (вкл) / «Выполнение» (выкл);
- в режиме обучения предоставьте скетчу возможность записывать значения весов и смещений в энергонезависимую память EEPROM;
- в режиме выполнения предоставьте скетчу возможность считывать значения весов и смещений из EEPROM;
- в режиме выполнения скетч должен запрашивать у пользователя новую пару входных данных вне (или внутри) исходного обучающего набора;
- введите код для отнесения предоставленной новой пары входных/выходных данных к определенной категории на основе весов и смещения обучающего набора;
- обеспечьте внешние светодиоды для индикации двух разных категорий объектов.

Мы используем встроенный переключатель (DIP) для выбора режима скетча «Обучение»/«Выполнение». Кроме того, три светодиода используются для индикации режима «Обучение» (вкл) / «Выполнение» (выкл) и категории 1 или 2, к которой отнесена новая входная пара, как показано на рис. 6.7.

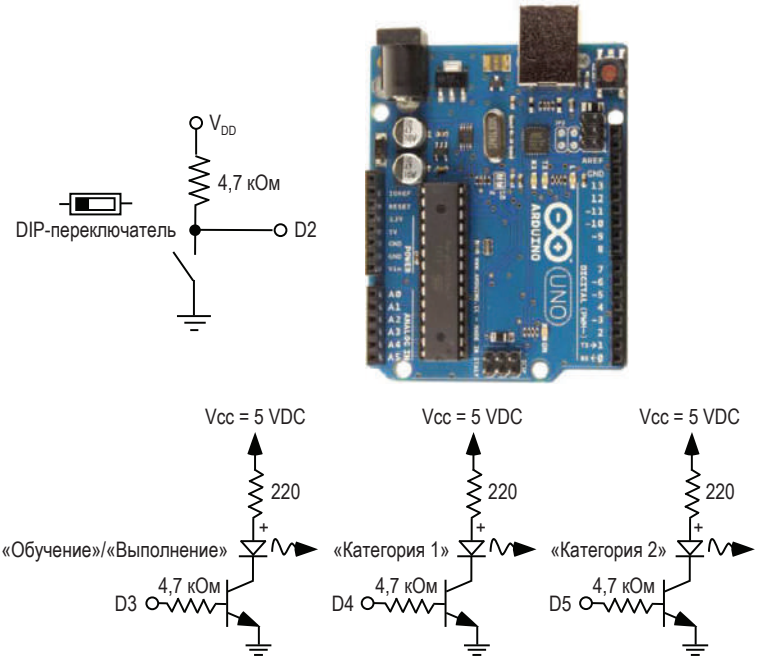


Рис. 6.7. Схема тестирования персептрона

Чтобы протестировать скетч, начните с установки DIP-переключателя режима «Обучение»/«Выполнение» в положение «Обучение». Когда скетч завершит обучение (MSE = 0 или желаемая величина MSE достигнута), измените положение переключателя на «Выполнение». Пользователю будет предложено ввести значения входных объектов x_1 и x_2. Чтобы протестировать персептрон, используйте входные значения как из исходного обучающего набора, так и за его пределами. Убедитесь, что алгоритм помещает данные в правильную категорию.

6.3.3. Сортировка помидоров

Помидоры доступны в широком диапазоне цветов и размеров. Помидоры сортируют по красноте и размеру. Это так называемые входные функции. Помидоры красные и большие по размеру получают лучшие цены. В этом примере мы разрабатываем и обучаем персептрон отделять самые красные и крупные помидоры от остальных. Набор данных для обучения персептрона представлен на рис. 6.8. Для классификации плодов мы используем два разных признака: покраснение (от 0 до 255) и диаметр (до 200 мм).

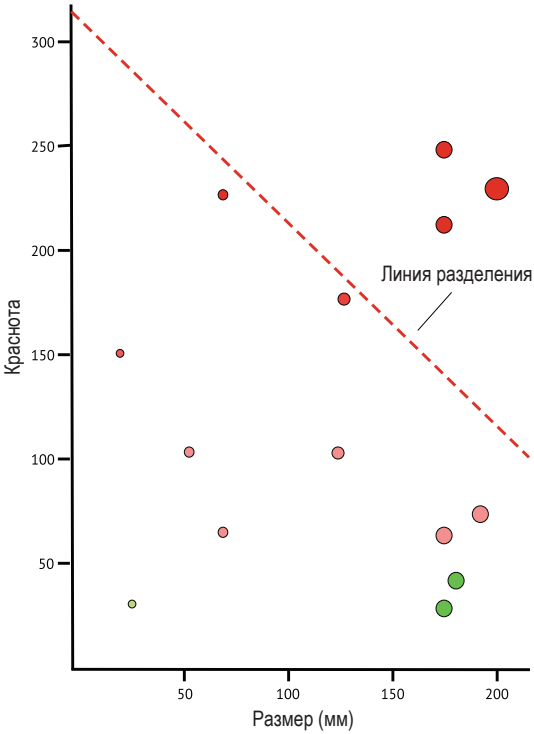


Рис. 6.8. Обучающий набор для сортировки помидоров

Для обучения модели персептрона мы используем тот же процесс, что и в предыдущем примере для получения значений весов

и смещения. Скетч немного изменен, чтобы результаты выводились на последовательный монитор только по завершении каждого цикла-эпохи. В скетче предусмотрен режим обучения. После обучения персептрона его можно использовать для классификации входных данных за пределами исходного набора. Мы называем это режимом выполнения (Run Mode).

```
//*************************************************************
//tomato_sorter
//Sketch has two modes of operation selected via a DIP switch
//attached to input D0:
// - D0 = HIGH, selects training mode
// - D0 = LOW, selects run mode
//
//Other pins used:
// - D1: LED to indicate mode Train(1)/Run(0)
// - D2: LED to indicate inputs placed in Category 1
// - D3: LED to indicate inputs placed in Category 2
//*************************************************************
//define
#define trng_set_size 12     //entries in training set
#define train_run_sw 0       //switch input train(1)/run(0)
#define train_run_LED 1      //LED for train(1)/run(0)
#define cat1_LED 2           //LED for category 1
#define cat2_LED 3           //LED for category 2
#define print_modulo 100     //print iteration
//global
float input_1[trng_set_size] = {250, 225, 175, 250, 220, 225, 100, 100, 75, 25, 60, 35};
float input_2[trng_set_size] = {25, 75, 125, 170, 170, 200, 50, 125, 190, 25, 75, 175};
int desired_output[trng_set_size] = {0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0};
int actual_output;
unsigned int epoch_iterations = 10000;
unsigned int epoch_count = 0;
float w1 = 0;                //set initial weight of w1
float w2 = 0;                //set initial weight of w2
float bias = 0;              //set initial bias value
float learning_rate = 0.1;   //set 0 to 1
int actual_out;              //output from perceptron
float error;                 //desired - actual output
int train_run_sw_value;      //switch input train(1)/run(0)
float net;                   //numerical out from perceptron
float sq_error = 0;          //squared error
float MSE = 1;               //mean squared error
void setup()
{
//Initialize byte-addressable EEPROM
//Set the Serial output
  Serial.begin(9600);
  pinMode(train_run_sw, INPUT);
  pinMode(train_run_LED, OUTPUT);
```

```
pinMode(cat1_LED, OUTPUT);
pinMode(cat2_LED, OUTPUT);
epoch_count = 0;
}
void loop()
{
  int i;
//Read Train (1) /Run Switch (0)
//Serial.print("Read Train(1)/Run Switch (0):");
//train_run_sw_value = digitalRead(train_run_sw);
  train_run_sw_value = HIGH;
//Serial.println(train_run_sw_value);
  if(train_run_sw_value == 1)             //training mode
  {
    if((MSE > 0)&&(epoch_count <=epoch_iterations))
    {
      epoch_count = epoch_count + 1;      //epoch count
      net = 0;                            //reset net
      sq_error = 0;                       //reset sq error
      for(i=0; i < trng_set_size; i++)
      {
        net = (input_1[i]*w1) + (input_2[i]*w2) + bias;
        if(net > 0)
        {
          actual_out = 1;
        }
        else
        {
          actual_out = 0;
        }
        if((i==(trng_set_size-1))&&(epoch_count%print_modulo==0))
        {
          Serial.println(" ");
          Serial.print("Epoch:");
          Serial.println(epoch_count);
          Serial.print("i: ");
          Serial.print(i);
          Serial.print(" w1:");
          Serial.print(w1);
          Serial.print(" w2:");
          Serial.print(w2);
          Serial.print(" b:");
          Serial.println(bias);
        }
        error = desired_output[i] - actual_out;
        sq_error = sq_error + (error * error);
        w1 = w1 + (learning_rate * error * input_1[i]);
        w2 = w2 + (learning_rate * error * input_2[i]);
        bias = bias + (learning_rate * error);
        if((i==(trng_set_size-1))&&(epoch_count%print_modulo==0))
        {
          Serial.print("net:");
```

```
          Serial.print(net);
          Serial.print(" actual out:");
          Serial.print(actual_out);
          Serial.print(" error:");
          Serial.print(error);
          Serial.print(" w1':");
          Serial.print(w1);
          Serial.print(" w2':");
          Serial.print(w2);
          Serial.print(" b':");
          Serial.print(bias);
        }
        if((i==(trng_set_size-1))&&(epoch_count%print_modulo==0))
//calculate response until MSE = 0
        {
          Serial.println(" ");
          MSE = sq_error/trng_set_size;
          Serial.print("MSE:");
          Serial.print(MSE);
          Serial.println(" ");
        }
      }//end for
    }//if (MSE)
//store results to EEPROM
  }
  else //Run Mode
  {
    Serial.println("else...");
//get values from EEPROM
//request input values from user via serial monitor
//process new input
//illuminate appropriate LED
  }
}
//*************************************************************
```

До достижения сходимости обучающий набор последовательно применялся к модели 1500 раз. Результаты обучения модели представлены на рис. 6.9.

Как и раньше, для разделения категорий используются значения весов и смещения. Линию, разделяющую объекты категории 1 и категории 2, можно построить с помощью линейного уравнения:

$$x_1 w_1 + x_2 w_2 + b = 0;$$

$$x_1(0{,}5) + x_2(0{,}5) + (-162{,}70) = 0;$$

$$y = mx + b;$$

$$y = -x + 325{,}4.$$

Полученная линия разделения показана на рис. 6.8.

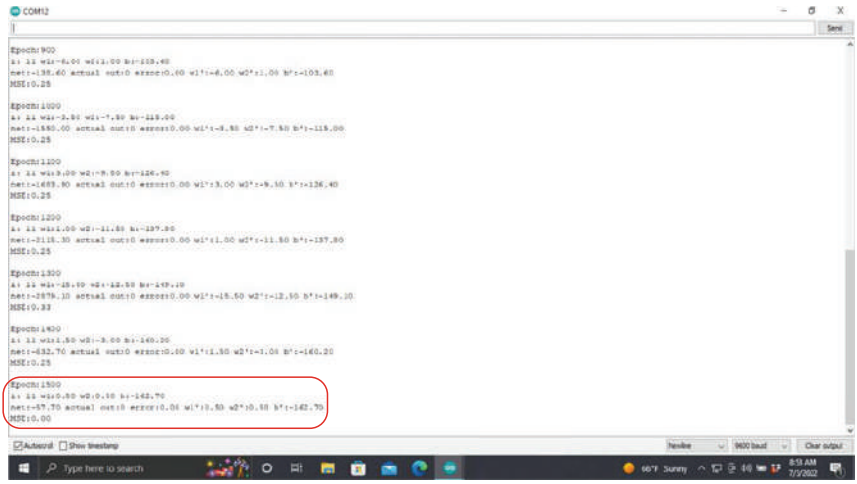


Рис. 6.9. Вывод скетча обученной модели одиночного персептрона – сортировка помидоров

После обучения персептрона его можно использовать для классификации входных данных за пределами исходного набора. Мы называем это режимом выполнения. В качестве следующего упражнения измените скетч сортировки помидоров, включив в него режим «Выполнить».

6.4. Модель группы персептронов

Модель одиночного персептрона можно использовать для линейного разделения объектов на две разные категории. Для создания дополнительных категорий можно использовать несколько персептронов. Сеть из нескольких персептронов показана на рис. 6.10. Важно отметить, что три персептрона действуют независимо друг от друга, то есть не обмениваются информацией между собой. Персептроны действуют независимо, но параллельно. Персептроны можно использовать для помещения входных данных в отдельные категории, если эти категории линейно разделимы.

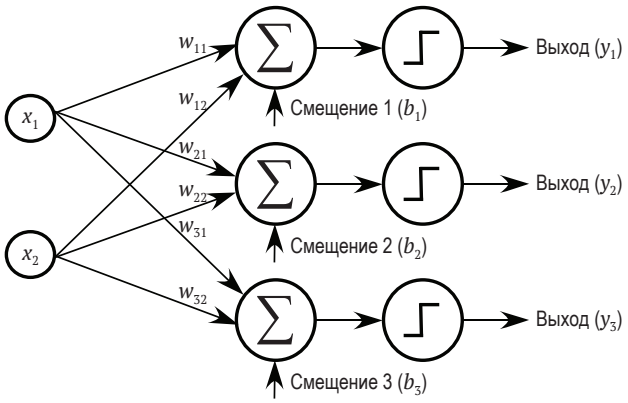


Рис. 6.10. Модель многих персептронов [4]

На рис. 6.11 представлен набор данных пар входа/выхода. Набор данных классифицируется по одному из трех выходных данных (y_1, y_2 или y_3). Модель трех персептронов обучается путем применения нескольких итераций набора данных к сети персептронов. Как показано на рис. 6.11, потребовалось более 900 итераций (эпох), чтобы все три персептрона сходились к нулевой среднеквадратической ошибке, используя схему, представленную ниже. Кроме того, трем персептронам потребовалось разное количество эпох для сходимости.

По завершении обучающей части скетча для каждого персептрона имеются необходимые веса и смещения. Как показано в предыдущем примере, веса и смещения используются для формирования линейных уравнений, отделяющих данную категорию пар входных/выходных данных от тех, которые находятся за пределами категории. Полученные линейные уравнения представлены ниже и нанесены на график (рис. 6.11). Обратите внимание, как каждая линия отделяет данную группу (кластер) пар входа/выхода от других кластеров.

Линия 1:

$$x_1 w_1 + x_2 w_2 + b = 0;$$

$$x_1(-12{,}8) + x_2(6{,}2) + (0{,}20) = 0;$$

$$y = mx + b;$$

$$y = 2{,}06x + -0{,}03.$$

Линия 2:

$$x_1 w_1 + x_2 w_2 + b = 0;$$

$$x_1(0{,}60) + x_2(0{,}90) + (-120{,}8) = 0;$$

$$y = mx + b;$$

$$y = -0{,}67x + 134{,}2.$$

X1	X2	Y1	Y2	Y3
25	62	1	0	0
12	75	1	0	0
20	80	1	0	0
1	79	1	0	0
80	87	0	1	0
87	80	0	1	0
95	95	0	1	0
76	94	0	1	0
40	17	0	0	1
57	20	0	0	1
35	15	0	0	1
26	4	0	0	1

Y1 X1 < 50, X2 > 50
Y2 X1 > 50, X2 > 50
Y3 ((X1 > 25)&&(X1<75)), X2 < 25

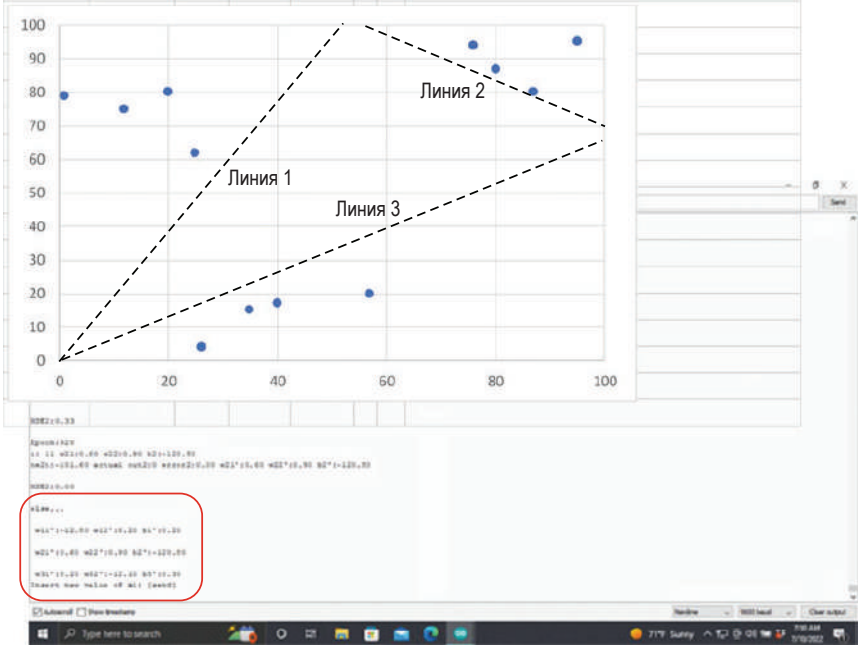


Рис. 6.11. Результаты обучения модели нескольких персептронов [4]

Линия 3:

$$x_1 w_1 + x_2 w_2 + b = 0;$$

$$x_1(8{,}20) + x_2(-12{,}1) + (0{,}30) = 0;$$

$$y = mx + b;$$

$$y = 0{,}67x + 0{,}02.$$

После обучения персептрона его можно использовать для классификации входных данных за пределами исходного набора данных (мы называем это режимом выполнения). Этот режим мы рассмотрим в следующем разделе главы.

```
//***************************************************************
//3perceptron_2in_1out_train_run_final
//Sketch implements 3 perceptron model
//Sketch has two modes of operation: Train and Run
//Model first performs training and then proceeds to run.
//***************************************************************
//define
#define trng_set_size 12     //entries in training set
//global
float input_1[trng_set_size] = {25.0, 12.0, 20.0, 1.0, 80.0, 87.0,  95.0, 76.0, 40.0, 57.0,
35.0, 26.0};
float input_2[trng_set_size] = {62.0, 75.0, 80.0, 79.0, 87.0, 80.0, 95.0, 94.0, 17.0, 20.0,
15.0, 4.0};
int desired_output1[trng_set_size] = {1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0};
int desired_output2[trng_set_size] = {0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0};
int desired_output3[trng_set_size] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1};
int actual_output1, actual_output2, actual_output3;
unsigned int epoch_iterations = 15000;
unsigned int epoch_count = 0;
float w11 = 0;               //set init weight of perceptron 1, x1
float w12 = 0;               //set init weight of perceptron 1, x2
float w21 = 0;               //set init weight of perceptron 2, x1
float w22 = 0;               //set init weight of perceptron 2, x2
float w31 = 0;               //set init weight of perceptron 3, x1
float w32 = 0;               //set init weight of perceptron 3, x2
float bias1 = 0, bias2 = 0, bias3 = 0;             //set init bias values
float learning_rate = 0.1;                         //set 0 to 1
float x1, x2;
int actual_out1, actual_out2, actual_out3;         //output from perceptron
float error1, error2, error3;                      //desired - actual output
float net1, net2, net3;                            //numerical out from perceptron
float sq_error1 = 0, sq_error2 = 0, sq_error3 =0;  //squared error
float MSE1 = 1, MSE2 = 1, MSE3 = 1;                //mean squared error
void setup()
{
```

```
  Serial.begin(9600);                          //Set the Serial output
  epoch_count = 0;                             //reset epoch count
}
void loop()
{
  int i;
//Training Mode
  if((epoch_count <= epoch_iterations)&&((MSE1 !=0)||(MSE2 !=0)||(MSE3 !=0)))
  {
//calculate response until MSE = 0
//or max count reached
    epoch_count = epoch_count + 1;             //epoch count
    while(!Serial);
    Serial.print("Epoch:");
    Serial.println(epoch_count);
    if((MSE1 > 0)&&(epoch_count <=epoch_iterations))
    {
      net1 = 0;                                //reset net
        sq_error1 = 0;                         //reset sq error
        for(i=0; i < trng_set_size; i++)
        {
          if(i == (trng_set_size-1))           //end of Epoch
        {
          Serial.print("i: ");
          Serial.print(i);
        }
        net1 = (input_1[i]*w11) + (input_2[i]*w12) + bias1;
        if(net1 > 0)
        {
          actual_out1 = 1;
        }
        else
        {
          actual_out1 = 0;
        }
        if(i == (trng_set_size-1))             //end of Epoch
        {
          Serial.print(" w11:");
          Serial.print(w11);
          Serial.print(" w12:");
          Serial.print(w12);
          Serial.print(" b1:");
          Serial.println(bias1);
        }
        error1 = desired_output1[i] - actual_out1;
        sq_error1 = sq_error1 + (error1 * error1);
        w11 = w11 + (learning_rate * error1 * input_1[i]);
        w12 = w12 + (learning_rate * error1 * input_2[i]);
        bias1 = bias1 + (learning_rate * error1);
        if(i == (trng_set_size-1))             //end of Epoch
        {
```

```
      Serial.print("net1:");
      Serial.print(net1);
      Serial.print(" actual ou1t:");
      Serial.print(actual_out1);
      Serial.print(" error1:");
      Serial.print(error1);
      Serial.print(" w11':");
      Serial.print(w11);
      Serial.print(" w12':");
      Serial.print(w12);
      Serial.print(" b1':");
      Serial.print(bias1);
      Serial.println(" ");
    }
    if(i == (trng_set_size-1))           //end of Epoch
    {
      MSE1 = sq_error1/trng_set_size;
      Serial.println(" ");
      Serial.print("MSE1:");
      Serial.print(MSE1);
      Serial.println(" ");
      Serial.println(" ");
    }
  }//end for
}//if (MSE1)
if((MSE2 > 0)&&(epoch_count <=epoch_iterations))
{
  net2 = 0; //reset net
  sq_error2 = 0; //reset sq error
  for(i=0; i < trng_set_size; i++)
  {
    if(i == (trng_set_size-1))           //end of Epoch
    {
      Serial.print("i: ");
      Serial.print(i);
    }
    net2 = (input_1[i]*w21) + (input_2[i]*w22) + bias2;
    if(net2 > 0)
    {
      actual_out2 = 1;
    }
    else
    {
      actual_out2 = 0;
    }
    if(i == (trng_set_size-1))           //end of Epoch
    {
      Serial.print(" w21:");
      Serial.print(w21);
      Serial.print(" w22:");
      Serial.print(w22);
```

```
      Serial.print(" b2:");
      Serial.println(bias2);
    }
    error2 = desired_output2[i] - actual_out2;
    sq_error2 = sq_error2 + (error2 * error2);
    w21 = w21 + (learning_rate * error2 * input_1[i]);
    w22 = w22 + (learning_rate * error2 * input_2[i]);
    bias2 = bias2 + (learning_rate * error2);
    if(i == (trng_set_size-1))           //end of Epoch
    {
      Serial.print("ne2t:");
      Serial.print(net2);
      Serial.print(" actual out2:");
      Serial.print(actual_out2);
      Serial.print(" error2:");
      Serial.print(error2);
      Serial.print(" w21':");
      Serial.print(w21);
      Serial.print(" w22':");
      Serial.print(w22);
      Serial.print(" b2':");
      Serial.print(bias2);
      Serial.println(" ");
    }
    if(i == (trng_set_size-1))           //end of Epoch
    {
      MSE2 = sq_error2/trng_set_size;
      Serial.println(" ");
      Serial.print("MSE2:");
      Serial.print(MSE2);
      Serial.println(" ");
      Serial.println(" ");
    }
  }//end for
}//if (MSE2)
if((MSE3 > 0)&&(epoch_count <=epoch_iterations))
{
  net3 = 0;                              //reset net
  sq_error3 = 0;                         //reset sq error
  for(i=0; i < trng_set_size; i++)
  {
    if(i == (trng_set_size-1))           //end of Epoch
    {
      Serial.print("i: ");
      Serial.print(i);
    }
    net3 = (input_1[i]*w31) + (input_2[i]*w32) + bias3;
    if(net3 > 0)
    {
      actual_out3 = 1;
    }
```

```
        else
        {
          actual_out3 = 0;
        }
        if(i == (trng_set_size-1))           //end of Epoch
        {
          Serial.print(" w31:");
          Serial.print(w31);
          Serial.print(" w32:");
          Serial.print(w32);
          Serial.print(" b3:");
          Serial.println(bias3);
        }
        error3 = desired_output3[i] - actual_out3;
        sq_error3 = sq_error3 + (error3 * error3);
        w31 = w31 + (learning_rate * error3 * input_1[i]);
        w32 = w32 + (learning_rate * error3 * input_2[i]);
        bias3 = bias3 + (learning_rate * error3);
        if(i == (trng_set_size-1))           //end of Epoch
        {
          Serial.print("net3:");
          Serial.print(net3);
          Serial.print(" actual out3:");
          Serial.print(actual_out3);
          Serial.print(" error3:");
          Serial.print(error3);
          Serial.print(" w31':");
          Serial.print(w31);
          Serial.print(" w32':");
          Serial.print(w32);
          Serial.print(" b3':");
          Serial.print(bias3);
          Serial.println(" ");
        }
        if(i == (trng_set_size-1))           //end of Epoch
        {
          MSE3 = sq_error3/trng_set_size;
          Serial.println(" ");
          Serial.print("MSE3:");
          Serial.print(MSE3);
          Serial.println(" ");
        }
        Serial.println(" ");
      }//end for
    }//if (MSE3)
  }
  else                                       //All MSEs = 0
  {                                          //Run Mode
    Serial.println("else...Run Mode");
//print results
    Serial.println(" ");
```

```
    Serial.print(" w11':");
    Serial.print(w11);
    Serial.print(" w12':");
    Serial.print(w12);
    Serial.print(" b1':");
    Serial.print(bias1);
    Serial.println(" ");
    //print results
    Serial.println(" ");
    Serial.print(" w21':");
    Serial.print(w21);
    Serial.print(" w22':");
    Serial.print(w22);
    Serial.print(" b2':");
    Serial.print(bias2);
    Serial.println(" ");
//print results
    Serial.println(" ");
    Serial.print(" w31':");
    Serial.print(w31);
    Serial.print(" w32':");
    Serial.print(w32);
    Serial.print(" b3':");
    Serial.print(bias3);
    Serial.println(" ");
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x1 input value from user via serial monitor
    Serial.println("Insert new value of x1: [send]");
    while(Serial.available()==0){}      //wait for user input data
    x1 = Serial.parseFloat();
    Serial.println(" ");
    Serial.print("x1:");
    Serial.println(x1);
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x input value from user via serial monitor
    Serial.print("Insert new value of x2: [send]");
    delay(5000);
    while(Serial.available()==0){}      //wait for user input data
    x2 = Serial.parseFloat();
    Serial.println(" ");
    Serial.print("x2:");
    Serial.println(x2);
//process new input and assign to appropriate category
```

```
//illuminate appropriate LED
    net1 = (x1*w11) + (x2*w12) + bias1;
    net2 = (x1*w21) + (x2*w22) + bias2;
    net3 = (x1*w31) + (x2*w32) + bias3;
    if(net1 > 0)
    {
//category 1
      Serial.println("Category 1");
      Serial.print("net1: ");
      Serial.println(net1);
    }
    else if (net2 > 0)
    {
//category 2
      Serial.println("Category 2");
      Serial.print("net2: ");
      Serial.println(net2);
    }
    else if (net3 > 0)
    {
//category 3
      Serial.println("Category 3");
      Serial.print("net3: ");
      Serial.println(net3);
    }
    else
    {
//Error
      Serial.println("Category Error");
    }
    delay(5000);
  }
}
//*************************************************************
```

6.4.1. Режим выполнения трех персептронов

В этом разделе представлен режим выполнения (Run Mode) для модели трех персептронов, рассмотренной в предыдущем разделе. Мы упростили скетч, удалив хранилище в EEPROM, не используя внешний DIP-переключатель режимов «Обучение»/«Выполнение» и светодиодные индикаторы состояния. Кроме того, мы протестировали скетч как на Arduino UNO R3, так и на Arduino Nano 33 BLE sense. Режим выполнения начинается с оператора `else` в вышеприведенном скетче.

Чтобы протестировать скетч, скомпилируйте и загрузите его в Arduino Uno R3 или Arduino Nano BLE 33 Sense. Когда обучающая

часть скетча будет завершена, он автоматически перейдет в режим выполнения. Вам будет предложено ввести значения x_1 и x_2. Используйте значения как из исходного обучающего набора, так и за его пределами. Убедитесь, что алгоритм помещает данные в правильную категорию.

6.5. Проблемы персептрона

Персептроны полезны для разделения пар входных/выходных данных, которые являются линейно разделимыми. Для наборов данных, не отвечающих этим требованиям, требуются более продвинутые методы [5]. Оставшуюся часть главы мы исследуем искусственные нейронные сети (ANN).

6.6. Искусственная нейронная сеть (ANN)

В этом разделе мы исследуем искусственную нейронную сеть (Artificial Neural Network, ANN). Начнем с обсуждения фундаментальной модели нейрона, разделив его на структурные элементы. Затем используем нейрон как структурный элемент для сборки ANN, содержащей несколько слоев взаимодействующих нейронов. Мы исследуем ANN прямого распространения, а затем обсудим концепцию обратного распространения ошибки (backpropagation) и модифицируем ANN в соответствии с этой концепцией. Вместе с этим представим скетчи Arduino для реализации нейрона, ANN прямого распространения и ANN с функциями обратного распространения ошибки. Математическое описание здесь основано на разработках, представленных в [4, 6, 8].

6.6.1. Модель одиночного нейрона

Модель одиночного нейрона представлена на рис. 6.12. Обратите внимание на поразительное сходство между моделями нейрона и персептрона (рис. 6.2). Мы модифицируем модель персептрона,

заменив функцию активации на сигмовидную функцию[1], а также переписав уравнения для обновления весов и смещения, как показано на рисунке.

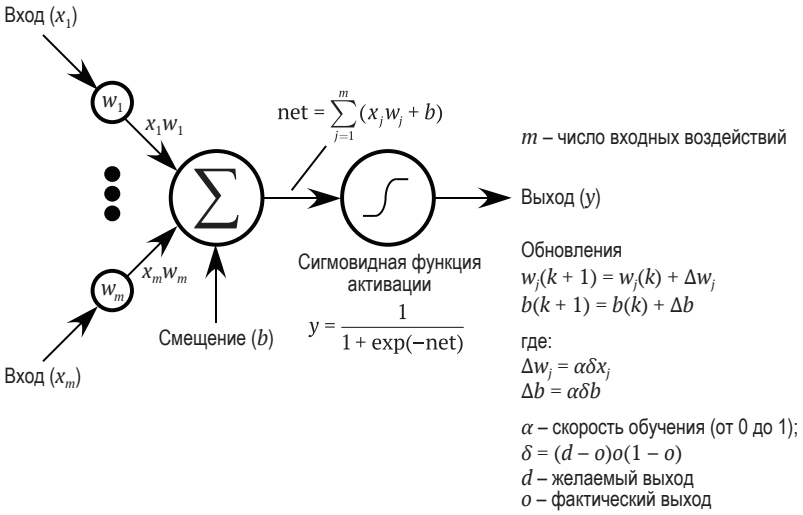


Рис. 6.12. Модель одиночного нейрона для входного слоя ([4])

Важно отметить, что модель сходится, а среднеквадратическая ошибка (mean squared error, MSE) уменьшается с каждой итерацией (эпохой). Однако MSE может не достичь нулевого значения после нескольких эпох. Таким образом, в скетче далее есть возможность установить максимальное количество итераций или минимальное значение желаемой MSE.

Как и прежде, мы сверяем уравнения с моделью в Excel, а затем приступаем к тестированию скетча Arduino. Результаты модели Excel и тестовый запуск скетча Arduino представлены на рис. 6.13.

Чтобы определить линию разделения между данными категорий 1 и 2 (см. рис. 6.13), скетч был запущен на Arduino Uno R3, с условием пока MSE не станет меньше 0,005. Для этого потребовалась 41 итерация, и в результате получились следующие значения: w_1 = –0,7232, w_2 = 0,6706 и смещение (bias) = 0,0568.

---

[1] Сигмовидная функция (сигмоида) – гладкая монотонная возрастающая функция, имеющая форму буквы «S». Подробнее о функции и ее разновидностях см. статью «Сигмоида» в Википедии. – *Прим. перев.*

$$x_1 w_1 + x_2 w_2 + b = 0.$$

$$x_1(-0{,}7232) + x_2(0{,}6706) + (0{,}0568) = 0.$$

$$x_2 = 1{,}08x_1 - 0{,}08.$$

Эта разделительная линия показана на рис. 6.13.

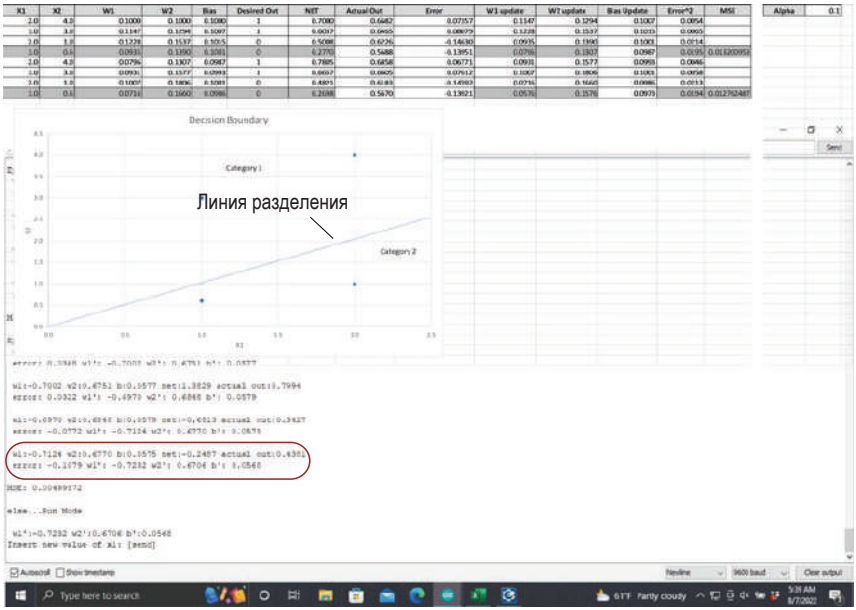


Рис. 6.13. Результаты выполнения модели нейрона

```
//*****************************************************************
//neuron_2in_1out_train_run
//Sketch has two modes of operation: Train and Run
//Model first performs training and then proceeds to run.
//*****************************************************************
//define
#define trng_set_size 4          //entries in training set
//global
double input_1[trng_set_size] = {2.0, 1.0, 2.0, 1.0};
double input_2[trng_set_size] = {4.0, 3.0, 1.0, 0.6};
double desired_output[trng_set_size] = {1.0, 1.0, 0, 0};
double actual_output;
unsigned int epoch_iterations = 10000;
unsigned int epoch_count = 0;
double w1 = 0.10;                //set initial weight of w1
```

```
double w2 = 0.10;               //set initial weight of w2
double bias = 0.10;             //set inital bias value
double learning_rate = 0.1;     //set 0 to 1
double x1, x2;                  //neuron inputs
double error;                   //desired vs actual output
double net;                     //numerical out from neuron
double sq_error = 0;            //squared error
double MSE = 1;                 //mean squared error
double MSE_old=1, MSE_new=2;    //detect change in MSE
double MSE_goal = 0.005;        //MSE goal
void setup()
{
  Serial.begin(9600);           //Set the Serial output
  epoch_count = 0;              //reset epoch count
}
void loop()
{
  int i;
  while(!Serial);
//train until condition not met
  if((MSE > 0)&&(epoch_count <=epoch_iterations)&&(MSE_old != MSE_new)&&(MSE_new > MSE_goal))
  {
    epoch_count = epoch_count + 1;     //epoch count
    Serial.print("Epoch:");
    Serial.println(epoch_count);
    net = 0;                           //reset net
    sq_error = 0;                      //reset sq error
    for(i=0; i < trng_set_size; i++)
    {
      net = (input_1[i]*w1) + (input_2[i]*w2) + bias;
      actual_output = (double) (1.0/(1.0 + exp(-net)));
      Serial.print(" w1:");
      Serial.print(w1, 4);
      Serial.print(" w2:");
      Serial.print(w2, 4);
      Serial.print(" b:");
      Serial.print(bias, 4);
      Serial.print(" net:");
      Serial.print(net, 4);
      Serial.print(" actual out:");
      Serial.println(actual_output, 4);
      error = (desired_output[i] - actual_output) * (actual_output) * (1 - actual_output);
      sq_error = sq_error + (error * error);
      w1 = w1 + (learning_rate * error * input_1[i]);
      w2 = w2 + (learning_rate * error * input_2[i]);
      bias = bias + (learning_rate * error * bias);
      Serial.print(" error: ");
      Serial.print(error, 4);
      Serial.print(" w1': ");
      Serial.print(w1, 4);
      Serial.print(" w2': ");
```

```
      Serial.print(w2, 4);
      Serial.print(" b': ");
      Serial.print(bias, 4);
      Serial.println(" ");
      if(i == (trng_set_size-1))       //end of Epoch
      {
        MSE_old = MSE;
        MSE = sq_error/trng_set_size;
        Serial.println(" ");
        Serial.print("MSE: ");
        Serial.print(MSE, 8);
        Serial.println(" ");
        MSE_new = MSE;
      }
      Serial.println(" ");
    }//end for
  }//if (MSE)
  else //run mode
  {
    Serial.println("else...Run Mode");
//print results
    Serial.println(" ");
    Serial.print(" w1':");
    Serial.print(w1, 4);
    Serial.print(" w2':");
    Serial.print(w2, 4);
    Serial.print(" b':");
    Serial.print(bias, 4);
    Serial.println(" ");
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x1 input value from user via serial monitor
    Serial.println("Insert new value of x1: [send]");
    while(Serial.available()==0){}     //wait for user input data
    x1 = Serial.parseFloat();
    Serial.println(" ");
    Serial.print("x1:");
    Serial.println(x1, 4);
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x input value from user via serial monitor
    Serial.print("Insert new value of x2: [send]");
    delay(5000);
    while(Serial.available()==0){}     //wait for user input data
    x2 = Serial.parseFloat();
```

```
    Serial.println(" ");
    Serial.print("x2:");
    Serial.println(x2, 4);
//process new input and assign to appropriate category
//illuminate appropriate LED
    net = (x1*w1) + (x2*w2) + bias;
    actual_output = (double) (1.0/(1.0 + exp(-net)));
    Serial.print("Actual out: ");
    Serial.println(actual_output);
    if(actual_output >= 0.5)
      Serial.println("Category 1");
    else
      Serial.println("Category 2");
    delay(5000);
  }//end else
}
//***************************************************************
```

Важно тщательно выбирать обучающий набор для ANN. После завершения обучения нейрон можно использовать для обработки пар входных/выходных данных за пределами обучающего набора. Мы называем это режимом выполнения (Run Mode) и рассмотрим данный режим в следующем разделе.

6.6.2. Режим выполнения одиночного нейрона

Режим выполнения для модели одного нейрона, рассмотренной в предыдущем разделе, начинается с оператора `else`. Чтобы протестировать скетч, скомпилируйте и загрузите его в Arduino UNO R3 или Arduino Nano BLE 33 Sense. Когда обучающая часть скетча завершена, он переходит в режим выполнения. Пользователю предлагается ввести значения входных характеристик x_1 и x_2. Используйте значения как из исходного обучающего набора, так и за его пределами. Убедитесь, что алгоритм помещает данные в правильную категорию.

6.6.3. Искусственные нейронные сети ANN

Отдельный нейрон служит основным структурным элементом искусственной нейронной сети (ANN), показанной на рис. 6.14, *а*. Эта конкретная ANN состоит из трех слоев: входного (input) слоя L1, скрытого (hidden) слоя L2 и выходного (output) слоя L3. Каждый нейрон показан на рисунке окружностью.

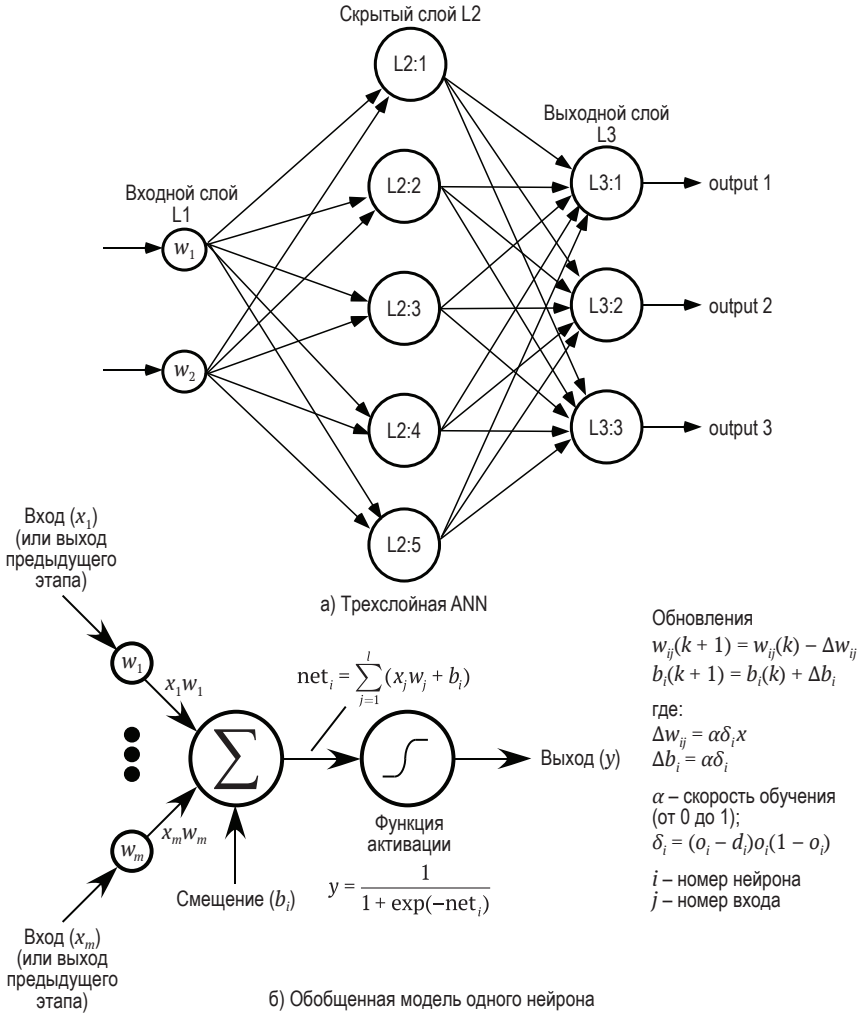


Рис. 6.14: *а* – модель трехслойной ANN, *б* – обобщенная модель одного нейрона ([4])

Взвешенные входные данные с выхода слоя L1 направляются к каждому узлу (нейрону) скрытого слоя L2. Его называют скрытым, поскольку он не имеет прямых внешних входов или выходов для всей сети. Выходные данные нейронов слоя 2 служат входными данными для нейронов слоя 3. Представленная ANN демонстрирует полностью связанную сеть узлов, т. е. каждый выход узла в данном слое обеспечивает вход для каждого узла следующего уровня.

Нейроны слоев 2 и 3 моделируются с использованием общей модели, показанной на рис. 6.14, *б*.

Чтобы отслеживать нумерацию, используемую для обозначения входов и выходов узла (нейрона), используется альтернативная схема, показанная на рис. 6.16. На этой упрощенной линейной диаграмме узлы (нейроны) показаны вертикальными линиями. Представлены соответствующие входы, смещение и выход для каждого узла.

ANN анализируется в прямом направлении слева направо с использованием модели нейрона, изображенной на рис. 6.14, *б*, и правил, представленных на рис. 6.15. Пошаговый подход к анализу ANN дан на рис. 6.16. Цифры внутри окружностей соответствуют шагам на рис. 6.15. Представленное здесь математическое описание основано на подробных разработках [4, 6, 8].

Элементы ошибок, возникающие на выходах ANN, распространяются обратно справа налево для обновления весов других слоев, как показано на рис. 6.17. Общая цель состоит в том, чтобы скорректировать веса и смещения сети так, дабы уменьшить ошибку, т. е. разницу между фактическими и желаемыми выходными данными. Как показано красным, ошибки на каждом выходе распространяются обратно через сеть ANN для обновления весов прямого потока.

Как и раньше, обучающий набор используется для итеративной установки весов и смещений до тех пор, пока не будут достигнуты желаемые значения ошибок на выходах ANN. Веса и смещения сети ANN могут обновляться после каждой выборки пары входных/выходных данных, после некоторого заданного количества выборок входных/выходных данных или после завершения всего набора выборок (эпохи). В следующем скетче мы обновляем веса сети ANN после каждой входной/выходной выборки [2].

Важно тщательно выбирать обучающий набор для ANN. После завершения обучения ANN можно использовать для обработки пар входных/выходных данных за пределами обучающего набора (мы называем это режимом выполнения). Если в ANN поступила новая пара входа/выхода, которая хорошо укладывается в обучающую выборку, ANN будет работать в режиме интерполяции. С другой стороны, если новая пара вход/выход находится за пределами области обучающей выборки, ANN работает в режиме экстраполяции и может дать неудовлетворительные результаты [10]. В этом конкретном примере используется приведенный обучающий набор, показанный на рис. 6.18[1]. UML-диаграмма высокого уровня для скетча представлена на рис. 6.19.

---

[1] О приведении набора данных к нормированному виду подробнее см. раздел 6.6.4. – *Прим. перев.*

Примечания

1. Перенумеруйте узлы (neuron) и слои Llayer: Llayer#:neuron#.
2. Обозначьте количество нейронов в слоях L1, L2 и L3 – n, m, l.
3. Обозначьте веса: w_{lij} $w_{\text{слой\#нейрон\#вход\#}}$.

Процесс обучения

1. Инициализируйте все веса и смещения во всех слоях небольшими случайными значениями.
2. Предоставьте входные данные обучающего набора для x_1 и x_2 на слое L1.
3. Расчет выходных данных выполняется послойно:
   – начните со слоя L2, рассчитайте значение net для каждого нейрона;
   – обработка значений net слоя L2 для определения функций активации.

   Выходы нейронов слоя L2:
   – выходы слоя 2 теперь служат входами слоя 3;
   – расчет значений net слоя L3 для каждого нейрона;
   – обработка значений net слоя L3 для определения функций активации.

   Выходы нейронов слоя L3:

$$\text{net}_i = \sum_{j=1}^{m} x_{ij} w_j + b \qquad O_i = \frac{1}{1 + \exp(-\text{net}_i)}$$

4. Рассчитайте изменение весов и смещений между уровнями 2 и 3:
   – сравните фактический выход уровня 3 (o) с желаемым выходом (d);
   – рассчитайте изменение веса: $\Delta w_{ij} = \alpha \delta o_j$, $\Delta b_i = \alpha \delta_i$, где:
   α – скорость обучения (от 0 до 1);
   $\delta_i = (o_i - d_i) o_i (1 - o_i)$;
   o_j – выходные данные нейрона j в слое L2.
5. Рассчитайте изменение весов и смещений между уровнями 1 и 2:

$$\Delta w_{ij} = \beta \delta_{HI} o_j, \quad \Delta b_i = \beta \delta_{HI},$$

   где

$$\delta_{HI} = o_i (1 - o_i) \sum_{k=1}^{m} \delta_k w_{ik};$$

   β – скорость обучения (от 0 до 1);
   o_j – выход нейрона j в слое L1;
   o_i – выход нейрона i в слое L2;
   δ_k – от нейронов слоя L3;
   w_{ik} – вес, соединяющий конкретный нейрон (k) в слое L2 с конкретным нейроном (i) в слое L3.
6. Обновите веса и смещения между уровнями L2 и L3 и уровнями L1 и L2:

$$w_{ij}(k+1) = w_{ij}(k) - \Delta w_{ij}, \qquad b_i(k+1) = b_i(k) - \Delta b_i.$$

7. Получите ошибку для нейронов в слое L3:

$$E = \frac{1}{2} \sum_{i=1}^{i} (o_i - d_i)^2.$$

8. Если величина ошибки достигла желаемой, прекратите обучение, иначе продолжайте.

Рис. 6.15. Процесс анализа ANN [4]

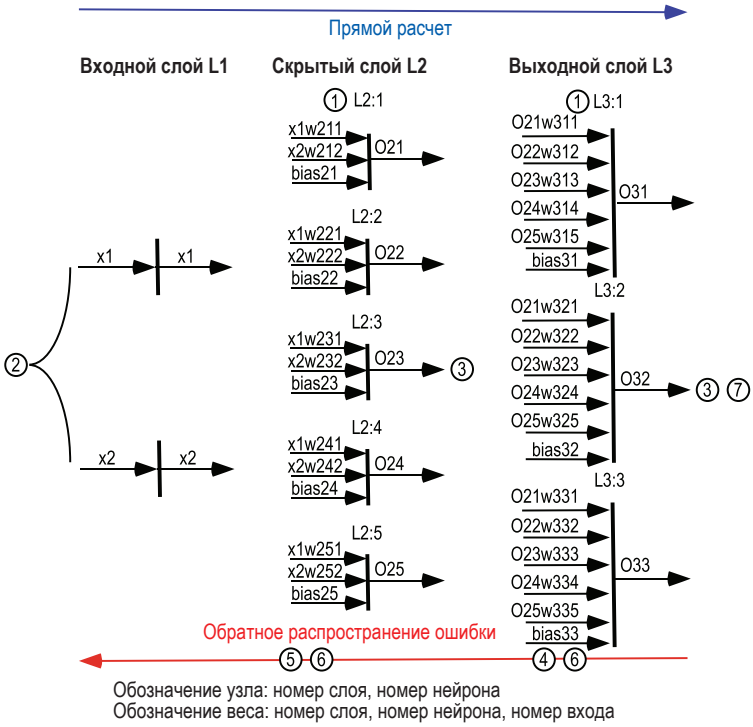


Рис. 6.16. Упрощенное представление ANN

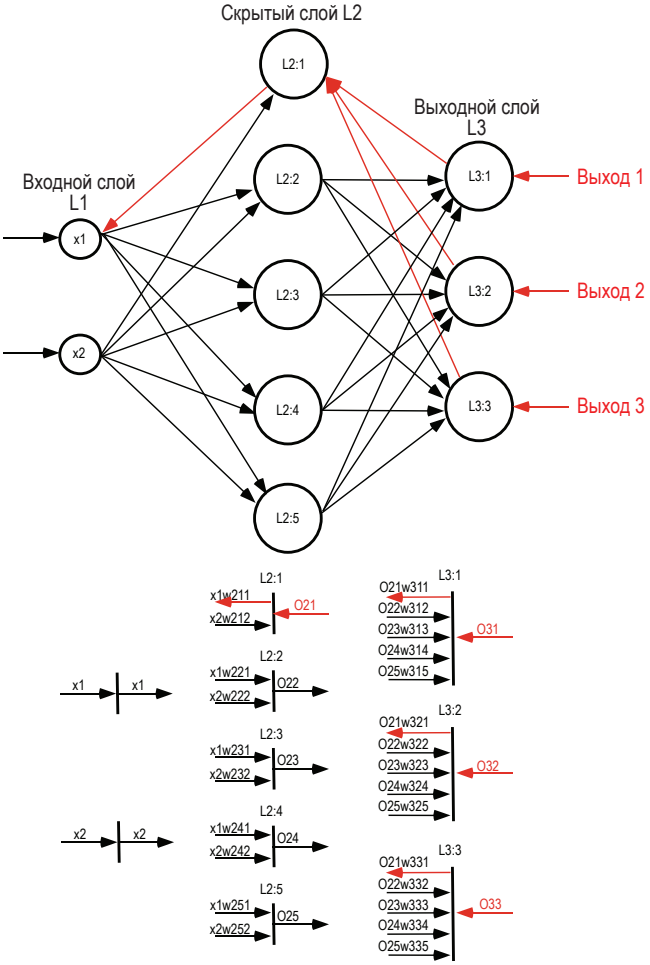
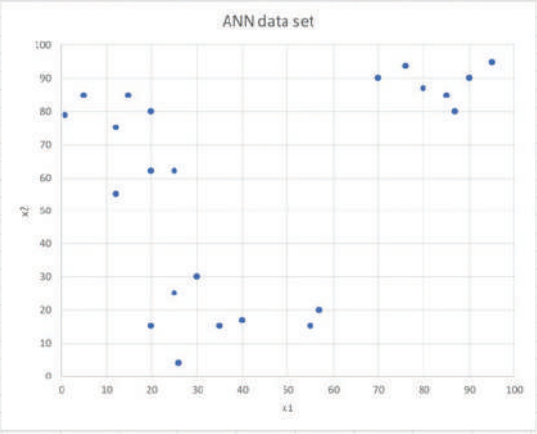


Рис. 6.17. Обратное распространение ошибки в сети ANN

x1	x2	y1	y2	y3
25	62	1	0	0
12	75	1	0	0
20	80	1	0	0
1	79	1	0	0
80	87	0	1	0
87	80	0	1	0
95	95	0	1	0
76	94	0	1	0
40	17	0	0	1
57	20	0	0	1
35	15	0	0	1
26	4	0	0	1
20	62	1	0	0
12	55	1	0	0
15	85	1	0	0
5	85	1	0	0
85	85	0	1	0
87	80	0	1	0
90	90	0	1	0
70	90	0	1	0
30	30	0	0	1
55	15	0	0	1
25	25	0	0	1
20	15	0	0	1
Input Average		51.9		
SD		32.2		

X1_Z	X2_Z
-0.84	0.31
-1.24	0.72
-0.99	0.87
-1.58	0.84
0.87	1.09
1.09	0.87
1.34	1.34
0.75	1.31
-0.37	-1.08
0.16	-0.99
-0.53	-1.15
-0.81	-1.49
-0.99	0.31
-1.24	0.10
-1.15	1.03
-1.46	1.03
1.03	1.03
1.09	0.87
1.18	1.18
0.56	1.18
-0.68	-0.68
0.10	-1.15
-0.84	-0.84
-0.99	-1.15



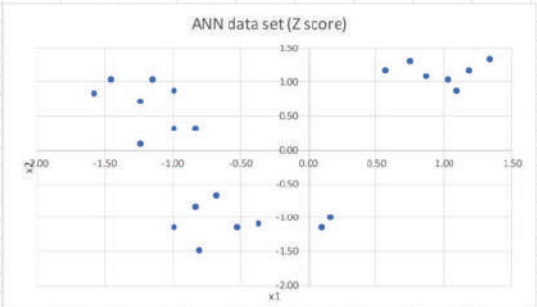


Рис. 6.18. Приведенные входные данные ANN

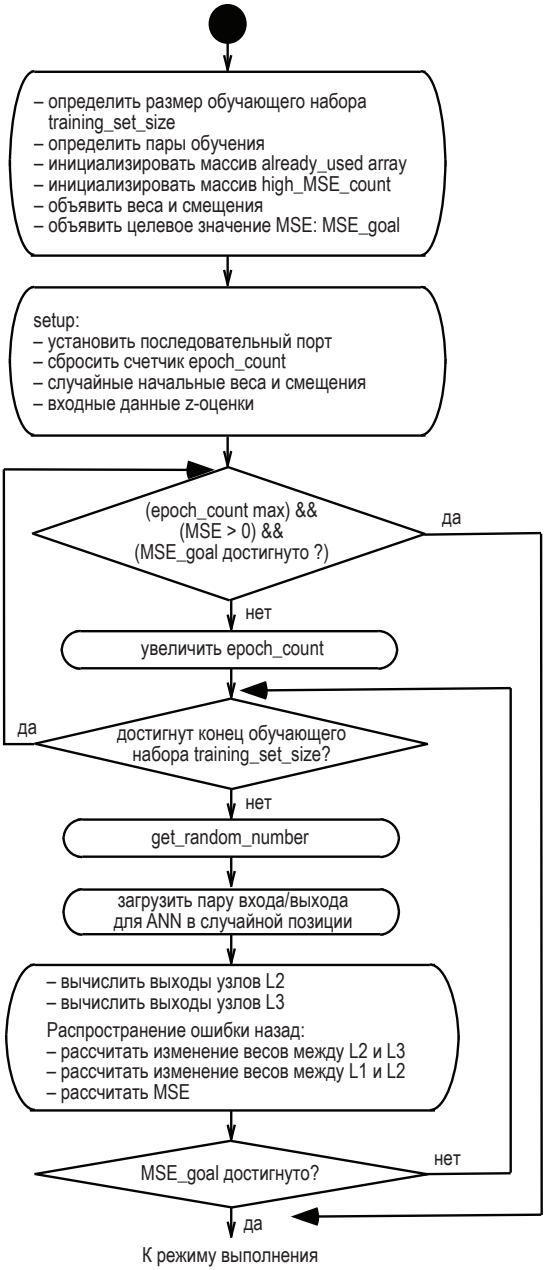


Рис. 6.19. UML-диаграмма высокого уровня для нейросети ANN

```
//*********************************************************************
//ANN_2in_3layer_3out_with_bias_max_mse_random_z_score
//Sketch has two modes of operation: Train and Run
//Model first performs training and then proceeds to run.
//ANN model has the following characteristics:
// - 2 inputs
// - 3 layers
// - 3 outputs
// - each node (neuron) has bias input
// - fully connected layers
//*********************************************************************
//define
#define trng_set_size 24   //entries in training set
#define troubleshoot 0     //flag to turn prints on(1)/off(0)
//data sets input_x1, input_x2/desired_output31, 32, 33
float input_x1[trng_set_size] = {25.0, 12.0, 20.0, 1.0, 20.0, 12.0, 15.0, 5.0, 80.0, 87.0,
95.0, 76.0, 85.0, 87.0, 90.0, 70.0, 40.0, 57.0, 35.0, 26.0, 30.0, 55.0, 25.0, 20.0};
float input_x2[trng_set_size] = {62.0, 75.0, 80.0, 79.0, 62.0, 55.0, 85.0, 85.0, 87.0, 80.0,
95.0, 94.0, 85.0, 80.0, 90.0, 90.0, 17.0, 20.0, 15.0, 4.0, 30.0, 15.0, 25.0, 15.0};

int desired_output31[trng_set_size] = {1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0};
int desired_output32[trng_set_size] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0, 0, 0, 0, 0, 0};
int desired_output33[trng_set_size] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
1, 1, 1, 1, 1, 1};
unsigned int already_used[trng_set_size] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
unsigned int high_MSE_count[trng_set_size] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0};
unsigned int epoch_iterations = 10000;  //maximum number of iterations
unsigned int epoch_count = 0;           //tracks number of interations
//layer 3 actual outputs
double actual_output31, actual_output32, actual_output33;
//layer 2 weights and biases
double w211, w212, w221, w222, w231;
double w232, w241, w242, w251, w252;
double bias21, bias22, bias23, bias24, bias25;
//layer 3 weights and biases
double w311, w312, w313, w314, w315;
double w321, w322, w323, w324, w325;
double w331, w332, w333, w334, w335;
double bias31, bias32, bias33;
double learning_rate = 0.05;                        //set from0 to 1
double x1, x2;                                      //neuron inputs
double delta21, delta22, delta23, delta24, delta25;
double delta31, delta32, delta33;                   //desired vs actual output
double net21, net22, net23, net24, net25;           //Layer 2 net output
double net31, net32, net33;                         //Layer 3 net output
double out21, out22, out23, out24, out25;           //Layer 2 output
double out31, out32, out33;                         //Layer 3 output
double sq_error31= 0, sq_error32= 0, sq_error33 = 0; //sq error
```

```
double MSE = 1;                       //data point MSE
double MSE_goal = 0.005;              //MSE goal
double max_MSE_epoch = 1;             //tracks max MSE per data set run
float input_average, input_stddev;    //used for input z transform
void setup()
{
  Serial.begin(9600);                 //set the Serial output
  epoch_count = 0;                    //reset epoch count
  randomize_weights_bias();           //randomize weights and bias
  z_score_input();                    //z score inputs x1 and x2
}
void loop()
{
  int i, j, m;
  unsigned new_number_selected;
  while(!Serial);                     //wait for serial connection
                                      //train until condition not met
  if((epoch_count<=epoch_iterations)&&(MSE>0)&& (max_MSE_epoch>MSE_goal))
  {
    max_MSE_epoch = -1;               //reset value
    epoch_count = epoch_count + 1;    //increment epoch count
    if(troubleshoot)
    {
      Serial.print("Epoch: ");
      Serial.print(epoch_count);
    }
    initialize_array();               //initialize random number used
                                      //start training epoch
    for(j=0; j < trng_set_size; j++)
    {
      i = get_unused_random_number();
      if(troubleshoot)
      {
        Serial.print(" ");
        Serial.print(i);
      }
//with random number "i" selected, analyze ANN with i data set
//Calculate Layer 2 neuron outputs
      net21 = (input_x1[i]*w211)+(input_x2[i]*w212)+bias21;    //L2,N1
      out21 = (double) (1.0/(1.0 + exp(-net21)));
      net22 = (input_x1[i]*w221)+(input_x2[i]*w222)+bias22;    //L2,N2
      out22 = (double) (1.0/(1.0 + exp(-net22)));
      net23 = (input_x1[i]*w231)+(input_x2[i]*w232)+bias23;    //L2,N3
      out23 = (double) (1.0/(1.0 + exp(-net23)));
      net24 = (input_x1[i]*w241)+(input_x2[i]*w242)+bias24;    //L2,N4
      out24 = (double) (1.0/(1.0 + exp(-net24)));
      net25 = (input_x1[i]*w251)+(input_x2[i]*w252)+bias25;    //L2,N5
      out25 = (double) (1.0/(1.0 + exp(-net25)));
//Layer 2 outputs become inputs to Layer 3
//Calculate Layer 3 neuron outputs
      net31 = (out21*w311)+(out22*w312)+(out23*w313)+ (out24*w314)+(out25*w315) + bias31;
//L3,N1
```

```
      out31 = (double) (1.0/(1.0 + exp(-net31)));
      net32 = (out21*w321)+(out22*w322)+(out23*w323)+ (out24*w324)+(out25*w325) + bias32;
//L3,N2
      out32 = (double) (1.0/(1.0 + exp(-net32)));
      net33 = (out21*w331)+(out22*w332)+(out23*w333)+ (out24*w334)+(out25*w335) + bias33;
//L3,N3
      out33 = (double) (1.0/(1.0 + exp(-net33)));
//Calculate change in weights between Layers 2 and 3
//neuron Layer 3, Neuron 1 (L3:1)
     delta31=(out31 - desired_output31[i])*(out31)*(1-out31);
     w311 = w311 - (learning_rate * delta31 * out21);
     w312 = w312 - (learning_rate * delta31 * out22);
     w313 = w313 - (learning_rate * delta31 * out23);
     w314 = w314 - (learning_rate * delta31 * out24);
     w315 = w315 - (learning_rate * delta31 * out25);
     bias31 = bias31 - (learning_rate * delta31);
//neuron Layer 3, Neuron 2 (L3:2)
     delta32=(out32 - desired_output32[i])*(out32)*(1-out32);
     w321 = w321 - (learning_rate * delta32 * out21);
     w322 = w322 - (learning_rate * delta32 * out22);
     w323 = w323 - (learning_rate * delta32 * out23);
     w324 = w324 - (learning_rate * delta32 * out24);
     w325 = w325 - (learning_rate * delta32 * out25);
     bias32 = bias32 - (learning_rate * delta32);
//neuron Layer 3, Neuron 3 (L3:3)
     delta33=(out33 - desired_output33[i])*(out33)*(1-out33);
     w331 = w331 - (learning_rate * delta33 * out21);
     w332 = w332 - (learning_rate * delta33 * out22);
     w333 = w333 - (learning_rate * delta33 * out23);
     w334 = w334 - (learning_rate * delta33 * out24);
     w335 = w335 - (learning_rate * delta33 * out25);
     bias33 = bias33 - (learning_rate * delta33);
//Calculate change in weights between Layers 1 and 2
//neuron Layer 2, Neuron 1 (L2:1)
     delta21 = out21 * (1 - out21) *
((w311*delta31)+(w321*delta32)+(w331*delta33));
     w211 = w211 - (learning_rate * input_x1[i] * delta21);
     w212 = w212 - (learning_rate * input_x2[i] * delta21);
     bias21 = bias21 - (learning_rate * delta21);
//neuron Layer 2, Neuron 2 (L2:2)
     delta22 = out22 * (1 - out22)* ((w312*delta31)+(w322*delta32)+(w332*delta33));
     w221 = w221 - (learning_rate * input_x1[i] * delta22);
     w222 = w222 - (learning_rate * input_x2[i] * delta22);
     bias22 = bias22 - (learning_rate * delta22);
//neuron Layer 2, Neuron 3 (L2:3)
     delta23 = out23 * (1 - out23) * ((w313*delta31)+(w323*delta32)+(w333*delta33));
     w231 = w231 - (learning_rate * input_x1[i] * delta23);
     w232 = w232 - (learning_rate * input_x2[i] * delta23);
     bias23 = bias23 - (learning_rate * delta23);
//neuron Layer 2, Neuron 4 (L2:4)
     delta24 = out24 * (1 - out24) * ((w314*delta31)+(w324*delta32)+(w334*delta33));
```

```
    w241 = w241 - (learning_rate * input_x1[i] * delta24);
    w242 = w242 - (learning_rate * input_x2[i] * delta24);
    bias24 = bias24 - (learning_rate * delta24);
//neuron Layer 2, Neuron 5 (L2:5)
    delta25 = out25 * (1 - out25) * ((w315*delta31)+(w325*delta32)+(w335*delta33));
    w251 = w251 - (learning_rate * input_x1[i] * delta25);
    w252 = w252 - (learning_rate * input_x2[i] * delta25);
    bias25 = bias25 - (learning_rate * delta25);
//Calculate error at the end of each sample
    MSE=0.5*(((desired_output31[i]-out31)*(desired_output31[i]-out31))+((desired_output32[i]-
out32)*(desired_output32[i]-out32))+((desired_output33[i]-out33)*(desired_output33[i]-
out33)));
    if(troubleshoot)
    {
      Serial.print("MSE: ");
      Serial.println(MSE, 4);
    }
//Print max MSE at the end of an epoch
    if(MSE > max_MSE_epoch)
    {
      max_MSE_epoch = MSE;
      m = i;
      if(troubleshoot)
      {
        Serial.print("max_MSE_epoch updated");
        Serial.println(max_MSE_epoch, 4);
      }
    }
    if(j == (trng_set_size-1))       //end of Epoch
    {
      Serial.print("Epoch: ");
      Serial.print(epoch_count);
      Serial.print(" max i : ");
      Serial.print(m);
      Serial.print(" max_MSE_epoch: ");
      Serial.println(max_MSE_epoch, 4);
      high_MSE_count[m] = high_MSE_count[m] + 1;
    }
  }//end for
 }//if (MSE)
 else                                //run mode
 {
   if(troubleshoot)
   {
     Serial.print("Epoch: ");
     Serial.print(epoch_count);
     for(j=0; j < trng_set_size; j++)
     {
       Serial.print("Training set: ");
       Serial.print(j);
       Serial.print(" Training set count: ");
```

```
        Serial.println(high_MSE_count[j]);
      }
    }
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x1 input value from user via serial monitor
    Serial.println("Insert new value of x1: [send]");
    while(Serial.available()==0){}            //wait for user input data
    x1 = Serial.parseFloat();
    x1 = (x1 - input_average)/input_stddev;   //z: input x1
    Serial.println(" ");
    Serial.print("x1:");
    Serial.println(x1, 4);
//flush input buffer
    while(Serial.available() >0)
    {
      Serial.read();
    }
//request x input value from user via serial monitor
    Serial.print("Insert new value of x2: [send]");
    delay(5000);
    while(Serial.available()==0){}            //wait for user input data
    x2 = Serial.parseFloat();
    x2 = (x2 - input_average)/input_stddev;   //z: input x2
    Serial.println(" ");
    Serial.print("x2:");
    Serial.println(x2, 4);
//process new input and assign to appropriate category
//Calculate Layer 2 neuron outputs
    net21 = (x1*w211) + (x2*w212) + bias21;   //layer 2, node1
    out21 = (double) (1.0/(1.0 + exp(-net21)));
    net22 = (x1*w221) + (x2*w222) + bias22;   //layer 2, node2
    out22 = (double) (1.0/(1.0 + exp(-net22)));
    net23 = (x1*w231) + (x2*w232) + bias23;   //layer 2, node3
    out23 = (double) (1.0/(1.0 + exp(-net23)));
    net24 = (x1*w241) + (x2*w242) + bias24;   //layer 2, node4
    out24 = (double) (1.0/(1.0 + exp(-net24)));
    net25 = (x1*w251) + (x2*w252)+ bias25;    //layer 2, node5
    out25 = (double) (1.0/(1.0 + exp(-net25)));
//Layer 2 outputs become inputs to Layer 3
//Calculate Layer 3 neuron outputs
    net31 = (out21*w311)+(out22*w312)+(out23*w313)+ (out24*w314)+(out25*w315) + bias31;
    out31 = (double) (1.0/(1.0 + exp(-net31)));
    net32 = (out21*w321)+(out22*w322)+(out23*w323)+ (out24*w324)+(out25*w325) + bias32;
    out32 = (double) (1.0/(1.0 + exp(-net32)));
    net33 = (out21*w331)+(out22*w332)+(out23*w333)+ (out24*w334)+(out25*w335) + bias33;
    out33 = (double) (1.0/(1.0 + exp(-net33)));
    Serial.print("out31: ");
```

```
    Serial.print(out31);
    Serial.print(" out32: ");
    Serial.print(out32);
    Serial.print(" out33: ");
    Serial.println(out33);
    delay(5000);
  }//end else
}
//*********************************************************************
void randomize_weights_bias()
{
//layer 2 weights and biases
  w211 = get_random_weight(); w212 = get_random_weight();
  w221 = get_random_weight(); w222 = get_random_weight();
  w231 = get_random_weight(); w232 = get_random_weight();
  w241 = get_random_weight(); w242 = get_random_weight();
  w251 = get_random_weight(); w252 = get_random_weight();
  bias21 = get_random_weight(); bias22 = get_random_weight();
  bias23 = get_random_weight(); bias24 = get_random_weight();
  bias25 = get_random_weight();
//layer 3 weights and biases
  w311 = get_random_weight(); w312 = get_random_weight();
  w313 = get_random_weight(); w314 = get_random_weight();
  w315 = get_random_weight(); w321 = get_random_weight();
  w322 = get_random_weight(); w323 = get_random_weight();
  w324 = get_random_weight(); w325 = get_random_weight();
  w331 = get_random_weight(); w332 = get_random_weight();
  w333 = get_random_weight(); w334 = get_random_weight();
  w335 = get_random_weight(); bias31 = get_random_weight();
  bias32 = get_random_weight(); bias33 = get_random_weight();
}
//*********************************************************************
double get_random_weight()
{
  double random_input;
  unsigned int random_min = 1, random_max = 21;
  unsigned int scale_factor = 200.0;
  randomSeed(analogRead(A0));
  random_input = ((double)(random(random_min,   random_max)))/scale_factor;
  if(random(0,101) > 50)              //randommize sign
       random_input = random_input * -1.0;
  return random_input;
}
//*********************************************************************
void initialize_array(void)
{
  unsigned int k;
  for(k=0; k < trng_set_size; k++)    //initialize array
  already_used[k] = 0;
}
//*********************************************************************
```

```
unsigned int get_unused_random_number(void)
{
  unsigned rn;
  unsigned new_number_selected;
  new_number_selected = 0;
  while(new_number_selected != 1)
  {
    rn = (unsigned int) (random(0, trng_set_size));
    if(already_used[rn] == 0)            //number not used yet
    {
      already_used[rn] = 1;              //number has now been used
      new_number_selected = 1;
    }
    else
    {
      new_number_selected = 0;
    }
  }//end while
  return rn;
}
//*****************************************************************
void z_score_input()
{
  int i;
  input_average = 0;
  for(i=0; i< trng_set_size; i++)
  {
    input_average = input_average + input_x1[i];
  }
  for(i=0; i< trng_set_size; i++)
  {
    input_average = input_average + input_x2[i];
  }
  input_average = input_average/(2.0 * trng_set_size);
  Serial.print("Average: ");
  Serial.println(input_average);
  input_stddev = 0;
  for(i=0; i< trng_set_size; i++)
  {
    input_stddev = input_stddev + ((input_x1[i] - input_average) *
(input_x1[i] - input_average));
  }
  for(i=0; i< trng_set_size; i++)
  {
    input_stddev = input_stddev + ((input_x2[i] - input_average) * (input_x2[i] - input_
average));
  }
  input_stddev = input_stddev/((2.0 * trng_set_size) - 1);
  input_stddev = sqrt(input_stddev);
/*Serial.print("Input stddev: ");
  Serial.println(input_stddev); */
```

```
//Generate x1 Z scores
  for(i=0; i< trng_set_size; i++)
  {
    input_x1[i] = (input_x1[i] - input_average)/input_stddev;
    if(troubleshoot)
    {
      Serial.print("x1: ");
      Serial.print(i);
      Serial.print(" Input x1 z score: ");
      Serial.println(input_x1[i]);
    }
  }
//Generate x2 Z scores
  for(i=0; i< trng_set_size; i++)
  {
    input_x2[i] = (input_x2[i] - input_average)/input_stddev;
    if(troubleshoot)
    {
      Serial.print("x2: ");
      Serial.print(i);
      Serial.print(" Input x2 z score: ");
      Serial.println(input_x2[i]);
    }
  }
}
//*********************************************************************
```

6.6.4. Сходимость ANN

Во время обучения ANN среднеквадратическая ошибка должна уменьшаться и в конечном итоге достичь желаемого целевого значения. В зависимости от конкретной конфигурации ANN и содержания набора данных этого может не произойти. Если ANN не сходится к целевому значению MSE, можно предпринять следующие действия:

- рандомизировать первоначальные веса и смещения в диапазоне небольших чисел. В представленном скетче это достигается с помощью функции `randomize_weights_bias()`. Внутри функции каждому весу и смещению ANN изначально присваивается небольшое случайное значение в диапазоне от ±0,005 до 0,100;
- рандомизировать порядок, в котором пары входных/выходных данных предоставляются ANN в данную эпоху. В представленном скетче это достигается с помощью функции `get_unused_random_number()`. Эта функция предоставляет случайное

число от 0 до `trng_set_size` – 1, чтобы при обучении выбрать случайный порядок пар входа/выхода из набора данных для ANN;

- предоставить бóльший обучающий набор ANN. Рекомендуется иметь не менее десяти разных записей для каждого входного объекта. В приведенном скетче мы использовали 24 пары входных/выходных данных для двух входных функций x1 и x2;
- привести набор данных к нормированному виду [1]. В представленном скетче это достигается с помощью функции `z_score_input()`. Нормированный набор данных заменяет каждое входное значение исходного набора его эквивалентом с помощью z-оценки[1] [1]. Показатель z данного входного значения рассчитывается путем вычитания входного среднего значения набора данных из входного значения и последующего деления на стандартное отклонение набора входных данных. В целом это дает эквивалентный набор данных со средним значением, равным нулю, и стандартным отклонением, равным единице, как показано на рис. 6.18;
- настроить архитектуру ANN (например, изменить количество узлов в скрытом слое). Некоторые авторы указывают, что это делается методом проб и ошибок.

6.7. Глубокие нейронные сети и глубокое обучение. Введение в программные инструменты

В этой главе мы изучали основы персептронов и ANN. Эти базовые концепции могут быть расширены в случае разработки многослойных нейронных сетей, или, как их еще называют, глубоких нейронных сетей (deep neural network, DNN). Для изучения этих концеп-

[1] Z-оценка, стандартизированная оценка (z score) – прием получения нормированной случайной величины, не зависящей от масштаба и единиц измерения. Множество данных x_i преобразуется так, чтобы среднее значение (математическое ожидание) равнялось нулю, а стандартное отклонение – единице (см. далее по тексту). – *Прим. перев.*

ций потребуются передовые инструменты машинного обучения на основе программного пакета TensorFlow. Этот набор инструментов изначально был разработан Google и является программным обеспечением с открытым исходным кодом (open source software).

В этом разделе мы даем краткое введение в набор программных инструментов TensorFlow. Это логически следующий шаг в изучении концепций искусственного интеллекта и машинного обучения. Читателю настоятельно рекомендуется приобрести книгу [9], чтобы сделать следующий шаг[1]. Книга представляет собой подробное поэтапное знакомство с передовыми инструментами машинного обучения, в том числе с разработкой многослойных сетей для глубокого обучения.

Рабочий процесс глубокого обучения, используемый TensorFlow, показан на рис. 6.20 [9]. Это похоже на последовательность действий, примененную при разработке моделей персептрона и ANN ранее в этой главе. Программные инструменты, используемые в сети глубокого обучения, включают:

- Python: язык программирования, используемый предлагаемыми программными инструментами для разработки систем машинного обучения;
- TensorFlow: набор инструментов для обучения, внедрения и тестирования сети глубокого обучения;
- TensorFlow Lite: вариант TensorFlow для разработки мобильных приложений;
- TensorFlow Lite for microcontrollers: вариант TensorFlow для разработки приложений на базе микроконтроллеров[2];
- Jupyter Notebooks: среда разработки обеспечивает хорошо документированное приложение машинного обучения с поддержкой комментариев, кода и визуализаций выполнения;
- Google Colab: онлайн-среда для запуска и совместного использования проектов машинного обучения на базе Jupyter Notebook.

Обычно при разработке приложения глубокого обучения для микроконтроллера шаги с 1 по 5 (см. рис. 6.20) выполняются глав-

---

[1] На русском языке рекомендуется книга: *Джан Марко Йодиче.* TinyML. Книга рецептов. М.: ДМК Пресс, 2023. – *Прим. перев.*

[2] Для работы библиотеки требуется 32-битный микроконтроллер, например Arduino Nano 33 BLE Sense. – *Прим. авт.*

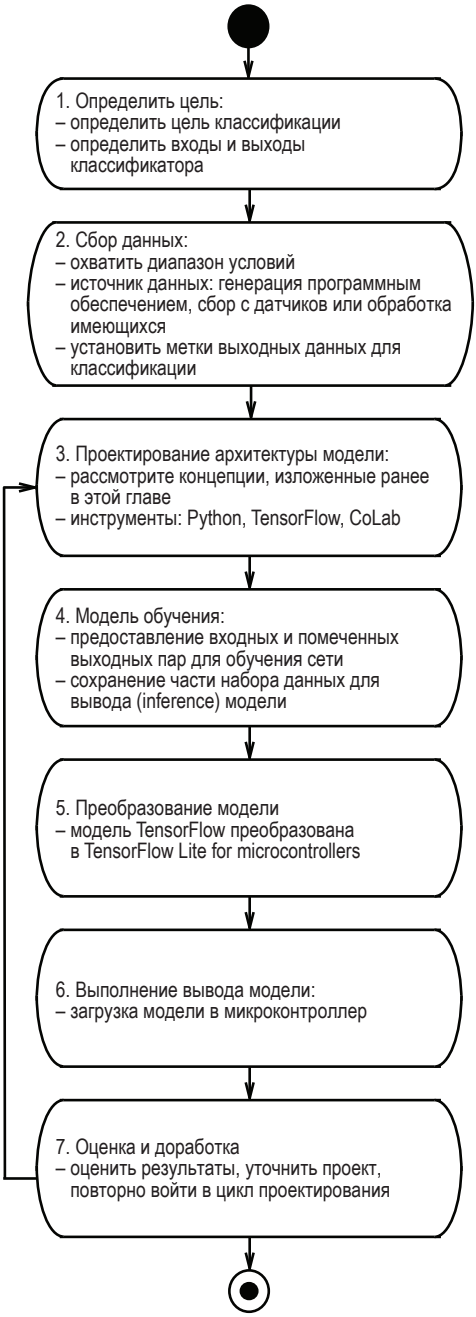


Рис. 6.20. Рабочий процесс глубокого обучения TensorFlow [9]

ным компьютером. После обучения и завершения модели она загружается в микроконтроллер для тестирования и уточнения. Книга [9] предоставляет четыре обученные модели. Библиотека TensorFlow обновляется ежедневно, и ее можно загрузить по ссылке, приведенной в этой книге[1]. После загрузки библиотеку можно импортировать в Arduino IDE обычным методом. К уже обученным примерам моделей относятся:

- hello_world: алгоритм принимает в качестве входных данных значение x и прогнозирует выходное значение y = sin(x);
- magic_wand: обученный алгоритм использует встроенный акселерометр для обнаружения жестов волшебной палочки в виде взмахов, вращения или наклонов и обеспечивает вывод на светодиоды и монитор последовательного порта;
- micro_speech: с помощью обученного приложения для распознавания речи загорается зеленый светодиод при произнесении слова Yes (да) или красный светодиод при произнесении слова No (нет);
- person_detection: предоставляет обученный алгоритм, который принимает изображение в качестве входных данных и определяет, отображается на нем лицо или нет. При обнаружении лица загорается светодиод.

Как видите, эти обученные алгоритмы можно легко адаптировать для выполнения других полезных и интересных действий. Со временем вам захочется разработать и обучить собственное приложение TensorFlow с нуля. Заинтересованному читателю предлагается изучить этот следующий шаг в книге [9].

6.8. Приложение: управление роботом с помощью ANN

В предыдущих главах мы разработали алгоритм управления роботом, перемещающимся по лабиринту. Разработайте сеть ANN для управления роботом с тремя датчиками (левым, центральным

[1] Необходимые ссылки также приводятся в упомянутой в предыдущей сноске книге «TinyML. Книга рецептов». – *Прим. перев.*

и правым) и четырьмя возможными вариантами движения (поворот налево, поворот направо, вперед и назад) в лабиринте.

Практические результаты разработки:

- схема сети ANN;
- набор данных, используемый для обучения сети;
- сходящийся алгоритм, реализованный в виде скетча Arduino, моделирующий данную сеть;
- план тестирования, демонстрирующий работу управления роботом.

6.9. Выводы

В этой главе мы исследовали концепцию моделей нейронов для решения проблем реального мира. Начали с краткого описания биологического нейрона и исследовали модель нейрона, названную персептроном, разработанную Фрэнком Розенблаттом в 1959 году. Мы использовали модель одиночного персептрона, чтобы разделить объекты на две категории. Затем расширили модель, включив в нее дополнительные персептроны для разделения объектов на несколько категорий. После этого мы исследовали одиночный нейрон, а затем изучили концепцию обратного распространения ошибки и разработали трехслойную нейронную сеть. Для всех этих разных моделей мы разработали скетчи для Arduino. Главу завершает краткое введение в программные инструменты для создания и реализации глубоких нейронных сетей.

6.10. Задания

1. Опишите возможности и ограничения обобщенной модели персептрона.
2. Опишите возможности и ограничения модели нескольких персептронов.
3. Опишите разницу между режимами обучения и выполнения в ANN.
4. Реализуйте режим выполнения для примера сортировщика помидоров с использованием модели одиночного персептрона.

5. Разработайте набор данных, который модель нескольких персептронов не сможет классифицировать[1]. Что ограничивает модель в классификации набора данных?
6. Используйте набор данных, разработанный в предыдущем задании, чтобы продемонстрировать классификацию с помощью искусственной нейронной сети.
7. Вы разработали искусственную нейронную сеть, которая не сходится. Каковы возможные пути решения проблемы?
8. Как измеряется сходимость в искусственной нейронной сети? Объясните.
9. В предыдущих главах мы разработали алгоритм управления роботом, перемещающимся по лабиринту. Наша цель – разработать сеть ANN для управления роботом с тремя датчиками (левым, центральным и правым) и четырьмя возможными вариантами движения (поворот налево, поворот направо, вперед и назад) в лабиринте. Приведите схему сети ANN. Сколько слоев будет содержать ANN? Сколько узлов будет в каждом слое?
10. Разработайте скетч Arduino, реализующий сходящийся алгоритм системы управления, описанной в предыдущем вопросе.
11. Сравните и сопоставьте нечеткую логику с механизмами ANN для управления роботом.
12. Ранее в этой главе мы исследовали приложение для сортировки помидоров. Внедрите сортировщик помидоров, чтобы распределить плоды по следующим группам: маленькие зеленые, маленькие красные, большие зеленые и большие красные. Будете ли вы использовать модель множественного персептрона или ANN? Объясните.
13. Реализуйте сортировщик, описанный в предыдущем вопросе. Продемонстрируйте правильную работу сортировщика.
14. Каковы различные методы нормировки входных данных в сети ANN?
15. Завершите режим выполнения ANN для UML-диаграммы действий.
16. Адаптируйте алгоритм примера «micro_speech» из «Arduino_TensorFlowLibrary», чтобы включать и выключать небольшой моторчик с помощью голосовых команд.

---

[1] Наглядные примеры таких наборов данных показаны в конце статьи https://skine.ru/articles/254647.

Источники

1. Baeldung, Normalizing Inputs for an Artificial Neural Network, www.baeldung.com.
2. J. Brownlee, Difference Between a Batch and an Epoch in a Neural Network, www.machinelearningmatery.com, July 2018.
3. Dan, Single Layer Perceptron Explained, ML Corner, October 13, 2020.
4. A. D. Kulkarni, Computer Vision and Fuzzy – Neural Systems, Prentice Hall, 2001.
5. M. L. Minsky and S. A. Papert, Perceptrons, expanded edition, The MIT Press, 1988.
6. T. Rashid, Make Your Own Neural Network, Middletown, DE, 2017.
7. C. F. Stevens, The Neuron, Scientific American, pp. 55–65, 1979.
8. M. Taylor, Neural Networks A Visual Introduction for Beginners, Blue Windmill Media, 2017.
9. P. Warden and D. Situnayake, TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, O'Reilly, 2020.
10. B. J. Wythoff, Backpropagation neural networks. A tutorial. Chemometrics and Intelligent Laboratory Systems, 18 (1993), 115–155.

Предметный указатель

A

ANN
  модель, 216
  нормирование входа, 232
  обратное распространение ошибки, 218
  режим интерполяции, 218
  режим экстраполяции, 218
  сходимость, 231
ANN (искусственная нейронная сеть), 211
APDS-9960, 71
Arduino IDE, 24
Arduino Nano 33 BLE Sense, 21, 35
Arduino UNO R3, 173
ASCII, 44

B

Bluetooth Low Energy (BLE), 56

E

EEPROM, 195

K

KNN
  ColorClassifer, 131
  метод K ближайших соседей, 129

L

LED. См. *Светодиод*

N

NINA B306 (модуль), 36
nRF52840 (процессор), 36

S

SRAM, 40

А

Аксон, 185
Аналогово-цифровой преобразователь, ADC, 53

Б

Барометр, 65, 68

В

Выходная функция
принадлежности, 162
Выходное устройство, 98

Д

Дарлингтона транзистор, 104
Датчик
  относительной влажности, 69
  расстояния, 76
Двигатель постоянного
тока, 100, 102
  управление, 106
  характеристики, 102
Дерево решений, 134
  обход, 150
Диапазон частот ISM, 56
Дисплей LCD, 46
Дребезг контактов, 96

Ж

ЖК-дисплей, 46

И

ИК-датчик, 110
Инерциальный измерительный
блок (IMU), 65
Интерфейс
  I2C, 52
  SPI, 47
  USART, 43

К

Кнопка, 94
  подключение, 95
Коэффициент заполнения, 41

Л

Лингвистическая
переменная, 157
Линейный привод, 101

М

Микрофон, 77
Модель персептрона
  группы, 201
  одиночного, 185

Н

Нейрон, 184
Нечеткая логика, 155
  контроллер, 173
  набор правил, 162

О

Относительная влажность, 69

П

Переключатель, 94
  подключение, 95
Персептрон, 185
  обучение, 187
Последовательная связь, 43

Р

Распознавание жестов, 71
Режим
  выполнения, 195, 198, 216
  обучения, 198
Робот Dagu Magician, 107

С

Светодиод, 98
  прямое напряжение, 98
  прямой ток, 98
Светодиодная лента, 28
Серводвигатель, 100
Симулятор, 80
Скачкообразная перестройка
частоты, 56

Скетч, 20, 26
Среднеквадратическая ошибка (MSE), 212
Стабилизатор напряжения, 91

Ф

Флеш-память, 40
Функция принадлежности, 159

Ш

Шаговый двигатель, 101
Широтно-импульсная модуляция (PWM), 41

Э

Энтропия, 139
Эпоха, 212

Стивен Ф. Барретт

Arduino: искусственный интеллект и машинное обучение

Стивен Ф. Барретт

ARDUINO:
ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И МАШИННОЕ ОБУЧЕНИЕ

Те, кто знаком с концепциями искусственого интеллекта (ИИ) и машинного обучения (МО), могут возразить, что они наиболее подходят для более мощных вычислительных платформ. Однако недавние разработки позволили использовать определенные приложения на микроконтроллерах после их обучения.

В этой книге обсуждаются методы ИИ и МО специально для микроконтроллеров. Цель состоит в том, чтобы познакомить вас с концепциями и позволить вам попрактиковаться на недорогом, доступном оборудовании и программном обеспечении Arduino.

Вначале рассматриваются Arduino IDE, Arduino Nano 33 BLE Sense, а также методы работы с сенсорами и периферийными интерфейсами. Далее на базе этих микроконтроллеров реализуются методы K ближайших соседей (KNN), деревья решений, нечеткая логика, персептроны и искусственные нейронные сети (ANN).

Издание предназначено широкому кругу радиолюбителей, инженеров, разработчиков, а также может быть полезно студентам и преподавателям.