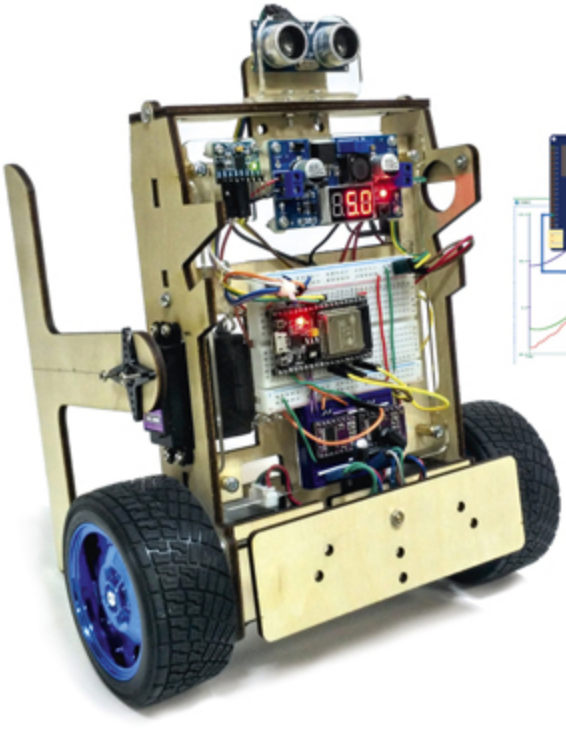
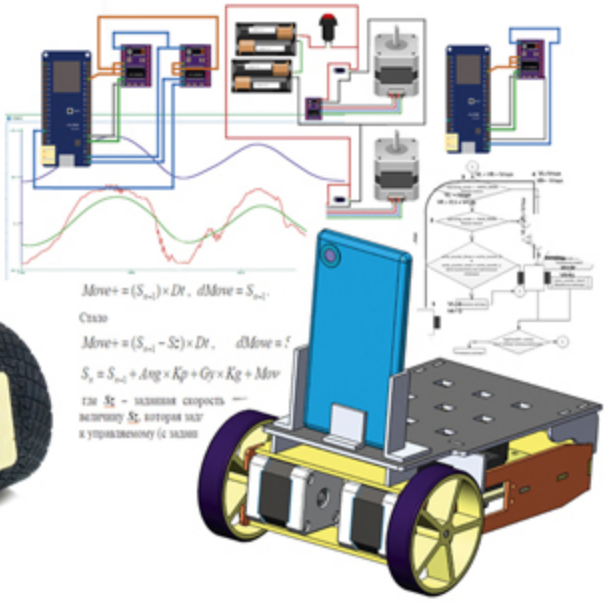
Электроника

Михаил Момот





Мобильные роботы на базе ESP32 в среде Arduino IDE

+ Материалы на www.bhv.ru

bhv®

УДК 007.52
ББК 32.816
М76

Момот М. В.

М76 Мобильные роботы на базе ESP32 в среде Arduino IDE. — СПб.: БХВ-Петербург, 2020. — 272 с.: ил.

ISBN 978-5-9775-6647-6

Руководство для начинающих конструкторов написано в форме практических проектов по построению мобильных роботов на новых высокоскоростных контроллерах ESP32. Использована единая базовая двухколесная конструкция на популярных высокоточных шаговых моторах. Все детали вырезаны из фанеры, их также можно напечатать на 3D-принтере. Описаны наиболее распространенные компоненты. Доступно изложено проектирование механики, приводов, элементов питания и стабилизации напряжения, электронных схем, программирование в среде Arduino IDE на примерах конструирования роботов различной функциональности. Особое внимание уделено созданию двухколесного балансирующего робота. Описано взаимодействие с датчиками нажатия (кнопка) и расстояния, гироскопом и акселерометром.

Электронный архив, находящийся на сайте издательства, содержит чертежи деталей для печати на 3D-принтере и листинги.

Для читателей, интересующихся электроникой, робототехникой и программированием микроконтроллеров

УДК 007.52
ББК 32.816

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Сависте*
Компьютерная верстка	*Людмилы Гауль*
Дизайн серии	*Марины Дамбиевой*
Оформление обложки	*Карины Соловьевой*

«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

ISBN 978-5-9775-6647-6

ОГЛАВЛЕНИЕ

ГЛАВА 1

КОНТРОЛЛЕРЫ НА БАЗЕ ESP32 И ИХ ПРИМЕНЕНИЕ В РОБОТОТЕХНИКЕ

От Ардуино к ESP32

Сложные «думающие» роботы управляются сложными и дорогими компьютерами. Простые мобильные роботы, о которых пойдет речь в книге, управляются более простыми и недорогими микроконтроллерами и контроллерами. Микроконтроллеры также могут успешно управлять несложными промышленными роботами.

Контроллер по структуре очень похож на компьютер (рис. 1.1), но, как правило, имеет больше ограничений, и специфика его применения не столь универсальна. Отличается контроллер от микроконтроллера тем, что все составляющие микроконтроллера расположены на одной микросхеме. Например, ATmega328 — микроконтроллер, потому что его микропроцессор, оперативная память и постоянная память (аналог жесткого диска компьютера) скомпонованы в пределах одной микросхемы.

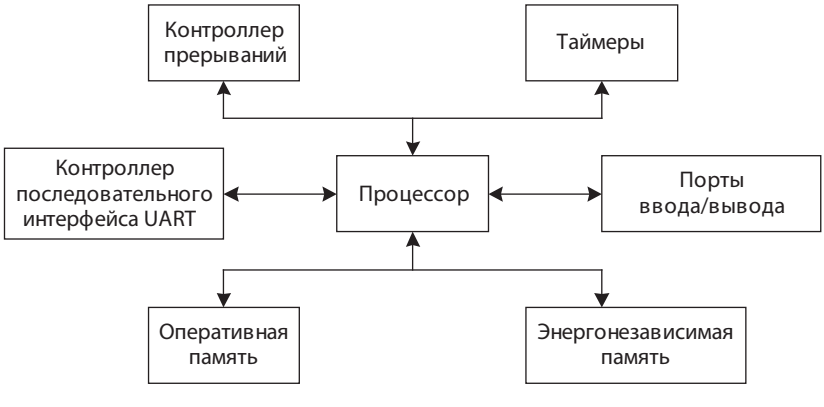


Рис. 1.1. Упрощенная структура контроллера

На микроконтроллерах ATmega основана популярная аппаратная вычислительная платформа контроллеров Arduino. Контроллеры Arduino представляют собой платы с распаянным микроконтроллером ATmega и всей необходимой для него обвязкой, регулятором напряжения и USB-UART-мостом. Все контакты платформы выведены на края платы и, как правило, уже оборудованы разъемами.

ESP32 — современный микроконтроллер, обладающий расширенным функционалом и многообещающими возможностями. В состав ESP32 входят модули Wi-Fi и Bluetooth, два процессорных ядра и богатый набор периферии. Именно поэтому применение его в робототехнике вполне обоснованно.

Замечу, что мы не призываем отказаться от использования Arduino Nano и Uno, но требования, предъявляемые к проектам, рассмотренным в этой книге, упираются в нехватку ресурсов ATmega328 и даже ATmega2560 (Arduino Mega) по скорости, по разрядности и по количеству встроенных интерфейсов. Arduino Nano и Uno применяют для решении задач, которые им по плечу, а при помощи ESP32 можно существенно расширить арсенал простых в монтаже, использовании и программировании контроллеров, «заточенных» на решение современных относительно скоростных задач. Вот несколько параметров, по которым ESP32 превосходит распространенные недорогие контроллеры Arduino: скорость опроса датчиков, скорость и качество управления шаговыми двигателями, возможность создания веб-сервера с поддержкой сетевого интерфейса Wi-Fi. Впрочем, рассматривать возможности лучше всего в сравнении с аналогами.

Сравнение характеристик контроллеров Arduino Nano, Arduino Mega2560 и микроконтроллера ESP32

Первая версия микроконтроллера ESP32 была выпущена компанией Espressif Systems (Китай, головной офис находится в Шанхае) в 2015 году. Микроконтроллер несколько раз незначительно модернизировался и к настоящему моменту представляет собой двухъядерный модуль, способный работать на частоте 240 МГц, имеющий 512 Кбайт встроенной оперативной памяти (той памяти, в которой располагаются код программы и данные при выполнении), а также большое количество поддерживаемых на аппаратном уровне интерфейсов. В то же время ESP32 рассчитан на работу с внешней энергонезависимой памятью.

ESP32 — микроконтроллер 32-разрядный, он может обработать за один такт ($^{1}/_{240000000}$ доля секунды) число длиной 4 байта (32 двоичных разряда), тогда как микроконтроллеры ATmega328, стоящие на платах Arduino Nano и Uno,

8-разрядные и обрабатывают за один такт 1 байт (8 двоичных разрядов). Это означает, что скорость обработки информации контроллером ESP32 превышает таковую в ATmega328 (Arduino Nano и Uno) приблизительно в 60–100 раз. В отличие от большинства недорогих контроллеров, предназначенных для проектов Arduino, микроконтроллер ESP32 обеспечивает поддержку питания 3,3 вольта.

Важной особенностью контроллеров, основанных на ESP32, являются встроенные беспроводные интерфейсы, благодаря чему весьма легко создать на основе такого контроллера простой веб-сервер, обращаться к нему через стандартный браузер смартфона или компьютера по IP-адресу во внутренней сети вашего дома (при подключении к Wi-Fi) и получать полезную информацию от подключенных к нему датчиков. Можно также организовать прием и передачу информации по каналу Bluetooth.

Ремарка от автора

Мое знакомство с контроллерами фирмы Espressif, которые поддерживают создание проектов Arduino, началось с другого контроллера — ESP8266, но работа с ним в среде Arduino IDE при создании мобильного робота не заладилась, поскольку для него оказалось недостаточно свободных выводов GPIO[1], — вернее, они были, но параллельно использовались для внутреннего потребления. Кроме того, программный код стандартных библиотек содержал ошибки и пробелы. Я пытался создать робота-футболиста, он должен был гонять мигающий в инфракрасном свете мяч. После пары дней экспериментов мне пришлось вернуться к использованию Arduino Nano, и я достаточно быстро реализовал задуманное, используя прямой доступ к регистрам ввода/вывода и повысив скорость аналого-цифрового преобразования информации от ИК фототранзисторов, выступающих в роли датчиков мяча, уменьшив его точность. Да, оптимизация программного кода нередко способна увеличить производительность программы чуть ли не в 10 раз.

Когда же была представлена поддержка платформы Arduino для чипа ESP32, я как раз занимался разработкой балансирующего робота на шаговых моторах. Для раздельного управления работой пары «шаговиков» потребовался контроллер с наличием двух 16-разрядных таймеров — по таймеру для управления каждым шаговым мотором. Arduino Nano и Uno подобными ресурсами не располагали, следовало сделать выбор между Arduino Mega2560 и ESP32: первый имел два 16-разрядных таймера, а второй — четыре 64-разрядных. Покопавшись в примерах кода, я счел работу с таймерами в ESP32 более простой и удобной, все остальные параметры мне также понравились, а наличия огромного количества выводов GPIO, как в Arduino Mega, в проекте балансирующего робота не требовалось. Робот был довольно быстро сконструирован и отлично показал себя на испытаниях, что и послужило предпосылкой к созданию других интересных проектов.

Представим сравнение ESP32 с двумя наиболее распространенными контроллерами, которые используются в проектах Arduino, в виде таблицы (табл. 1.1).

[1] GPIO — интерфейс ввода/вывода общего назначения (от англ. General-Purpose Input/Output). В общих случаях это информационные контакты, способные к приему или передаче данных. В проектах Arduino обычно используется наименование PIN (пин).

Таблица 1.1. *Сравнение характеристик контроллеров Arduino Nano, Arduino Mega2560, ESP32*

	Arduino Nano (ATmega328)	**Arduino Mega (Mega2560)**	**ESP32**
Разрядность данных	8	8	32
Тактовая частота, МГц	16	16	240
Размер ОЗУ, Кбайт	2	8	512
Размер энергонезависимой памяти: данные/код, Кбайт	1/32	4/256	4000–16 000 (внешняя память)
Количество контактов GPIO общее	14	54	до 25 (зависит от реализации)
16-разрядные таймеры общего назначения	1	4	—
8-разрядные таймеры общего назначения	2	2	—
64-разрядные таймеры общего назначения	—	—	4
Порт UART (аппаратный)	1 (D0, D1)	4 (D0–D1, D19–D18, D17–D16, D15–D14)	3 (можно выбирать GPIO)
Разрядность АЦП, бит	10	10	12
Время одного аналого-цифрового преобразования, мкс	~100	~100	~10
Количество контактов GPIO для АЦП	8	16	16–18
ЦАП (аналоговый сигнал на выходе)	нет	нет	2 канала
Количество каналов ШИМ (PWM)	6	14	16
Разрядность ШИМ (PWM), бит	8	8	1–15
Частота ШИМ (PWM), Гц	~500	~500	1–78 125 (при разрядности ШИМ 10)
Кол-во интерфейсов I^2C	1 (A4,A5)	1 (D20,D21)	2 (можно выбирать GPIO)
Поддержка Bluetooth	—	—	1
Поддержка Wi-Fi	—	—	1
Рабочее напряжение, вольт	5	5	3,3
Напряжение питания, подаваемое на плату контроллера, вольт	7–12	7–12	5

Рассмотрим теперь некоторые данные табл. 1.1 подробнее:

- *оперативная память* (эта быстрая память с прямым доступом служит для хранения данных во время исполнения программы) — регламентирует размер исполняемой программы и размер используемых данных переменных и массивов. Из табл. 1.1 можно сделать вывод, что размер ОЗУ ESP32 в 512 Кбайт позволяет разместиться весьма объемной программе (даже есть примеры создания операционной системы Lua-RTOS для ESP32). Для наших будущих проектов такой объем оперативной памяти можно считать достаточным и даже избыточным;
- *энергонезависимая память* (аналог жесткого диска компьютера) — этот тип памяти в контроллерах ATmega разделен на сектор данных и сектор исполняемого кода. В ESP32 такого разделения нет, как нет и вообще внутренней энергонезависимой памяти, — чип Flash-памяти здесь используется внешний, это отдельная микросхема на плате, не имеющая аппаратного разграничения на сектор данных и кода. По опыту, при использовании версии модуля ESP32 с 4 Мбайт Flash-памяти для данных выделяется примерно 1 Мбайт — здесь можно сохранять данные, которые будут доступны после перезагрузки и/или выключения питания. Например, информацию о пройденном роботом пути. Микроконтроллер ESP32 имеет 48 выводов, но не все из них можно использовать в программе, — например, контакты GPIO 6–11 могут быть выведены на некоторых платах, но заняты для доступа к данным распаянной на плате Flash-памяти;
- *количество контактов GPIO* — сравнимо с таковым в Arduino Nano, поэтому практически все проекты, реализованные на Nano, могут быть легко перенесены на ESP32. При этом следует иметь в виду, что не все из 48 выводов микроконтроллера ESP32 удается использовать в программе, — например, контакты GPIO 6–11 могут быть выведены на некоторых платах, но заняты для доступа к данным распаянной на плате Flash-памяти;
- *таймеры* — здесь все прекрасно. Отметим только, что 64-разрядные таймеры общего назначения ESP32 тактируются не частотой контроллера, а частотой 80 МГц. Их использование очень удобно и не мешает другим ресурсам, — например, чтобы генерировать сигналы ШИМ, на Arduino Nano задействуются системные таймеры. В ESP32 есть отдельный модуль генерации ШИМ (PWM). Таймеры можно применять для запуска небольших участков программного кода (функций) через равные промежутки времени в фоновом режиме. Основная программа на это время приостанавливается (если используется одно и то же ядро), а после выполнения кода, привязанного к таймеру, продолжается. Эти программы называются *обработчиками прерываний*, они должны быть максимально быстрыми, чтобы их постоянные вызовы не замедляли основную программу;
- *аппаратный UART* — это последовательный интерфейс (в ПК эту функцию реализуют COM-порты) для связи с внешними устройствами. Он позволяет

осуществлять обмен информацией между различными контроллерами на заданной скорости и задействуется при разработке систем, нуждающихся в обмене информацией. Преимущество аппаратной реализации UART заключается в освобождении ресурсов ядра контроллера, поскольку используется отдельный аппаратный модуль;

- *аналого-цифровой преобразователь, АЦП* — позволяет измерять величину напряжения на некоторых выводах GPIO и представлять его в числовом виде. Для Nano и Mega — это числа от 0 до 1000 (измеряемое напряжение 0–5 В), для ESP32 — числа от 0 до 4000 (измеряемое напряжение 0–3,3 В). По умолчанию в качестве опорного напряжения берется напряжение питания микроконтроллера. Чем выше разрядность, тем выше точность преобразования. В некоторых проектах также важна скорость преобразования;
- *широтно-импульсная модуляция, ШИМ* (от англ. Pulse-Width Modulation, PWM) — используется в управлении мощностью моторов, сервомашинок, бесколлекторных двигателей (например, в квадрокоптерах). Изменять мощность, подаваемую на моторы, она позволяет за счет изменения длины импульсов, генерируемых на заданной частоте. А то, что частота ШИМ может быть изменена в широких пределах, позволяет очень гибко подходить к настройкам оборудования;
- *I^2C-интерфейс* (последовательная общая шина) — при использовании ESP32 с Arduino IDE по умолчанию для шины I^2C (библиотека Wire) задействуются стандартные выходы ESP32: GPIO 21 (SDA) и GPIO 22 (SCL). При этом номера GPIO можно менять программно. I^2C-интерфейс применяется различными «умными» датчиками и дисплеями, наличие двух подобных интерфейсов не лишнее — на один интерфейс можно подключить одновременно до четырех устройств, если в проекте их больше. Если же есть пара устройств с одинаковыми I^2C-адресами, то использование ESP32 также оправданно;
- *цифроаналоговое преобразование, ЦАП* — напряжение на входе измеряется и представляется в числовой форме. В ESP32 есть два 8-разрядных канала преобразования цифровых сигналов в аналоговые: DAC1 (GPIO25) и DAC2 (GPIO26). ЦАП можно применять при генерации звука или сигналов управления другими аналоговыми устройствами;
- *поддержка Bluetooth* — весьма удобна в мобильных роботах и других домашних несложных автоматических устройствах. Часто применяется для управления роботами со смартфона. Через Bluetooth можно получать от робота статистическую информацию и другие данные. Для решения аналогичных задач с Arduino Nano/Mega потребуется дополнительно использовать внешний приемопередатчик, задействовать аппаратный UART-порт или создавать программный UART на свободных выводах GPIO, что, конечно, менее удобно, чем наличие встроенного модуля;

- *поддержка Wi-Fi* — встроенный модуль Wi-Fi позволяет легко подключать устройства на ESP32 к Интернету. В частности, можно получить очень удобный интерфейс для удаленного контроля за мобильными роботами;
- *напряжение питания* — современная архитектура в совокупности с высокой скоростью работы на частоте 240 МГц привели к тому, что напряжение питания ESP32 было уменьшено до 3,3 В. Для совместимости устройств с различными величинами логических уровней используются конвертеры уровней или более простые компоненты вроде резисторного делителя или опторазвязки. Отметим, что потребление тока ESP32 при работе с беспроводными интерфейсами возрастает, что накладывает дополнительные требования на стабильность электропитания. В ESP32 присутствует поддержка ядра низкого энергопотребления. В частности, выводы GPIO, перенаправленные в подсистему с низким энергопотреблением RTC, могут использоваться, когда ESP32 находится в глубоком сне.

Мы рассмотрели основные возможности и особенности ESP32, на которые будем опираться в книге, но следует иметь также в виду, что реальные возможности этого микроконтроллера еще шире. ESP32 по функционалу ближе к одноплатным компьютерам (Raspberry Pi), чем к контроллерам, но его низкая стоимость и возможность программирования в простой среде Arduino IDE позволяют широко применять ESP32 в Arduino-проектах.

Познакомимся теперь с основными доступными на рынке модулями с микроконтроллером ESP32.

Модули на основе ESP32 для создания прототипов

ESP32 с 4 Мбайт Flash-памяти

Это наиболее распространенный контроллер (рис. 1.2). Однако с подобным модулем не всегда удобно работать — у него нет встроенных стабилизатора питания и конвертера USB-UART, благодаря которому контроллеры Arduino удобно программируются через USB-интерфейс компьютера. Поэтому далее мы рассмотрим модули с уже распаянными на плате стабилизатором питания и USB-UART.

ESP32 Lolin Lite

Типичный пример модуля ESP32, пригодного для установки на плате для макетирования (рис. 1.3). Защитный стальной экран на контроллере отсутствует,

контроллер распаян непосредственно на макетной плате, GPIO 21 не разведен. Предполагалось, что подобные модули будут дешевле, но на деле разницы в цене с другими модулями нет. Расстояние между рядами выводов Lolin Lite 23,5 мм, что меньше, чем у других модулей, где это расстояние равно 26,5 мм, поэтому его можно установить в стандартную плату для прототипирования (рис. 1.4).

ESP32 LOLIN D32

Этот модуль (рис. 1.5) отличается от предыдущего более прочным исполнением: текстолит толщиной 1,7 мм против 1,1 мм у Lolin Lite, микроконтроллер защищен металлическим кожухом, но его уже не установить в стандартную «макетку».

ESP32 AI-Thinker с видеокамерой

Интересные проекты можно придумать, используя модуль ESP32 с камерой (рис. 1.6). Скорости ESP32 не всегда хватает для передачи картинки по Wi-Fi, в то же время возможностей упрощенного внутреннего анализа изображения должно быть вполне достаточно, чтобы, например, двигаться по нарисованным на полу меткам.

Рис. 1.2. Модуль ESP32 со встроенной антенной и экранирующей крышкой

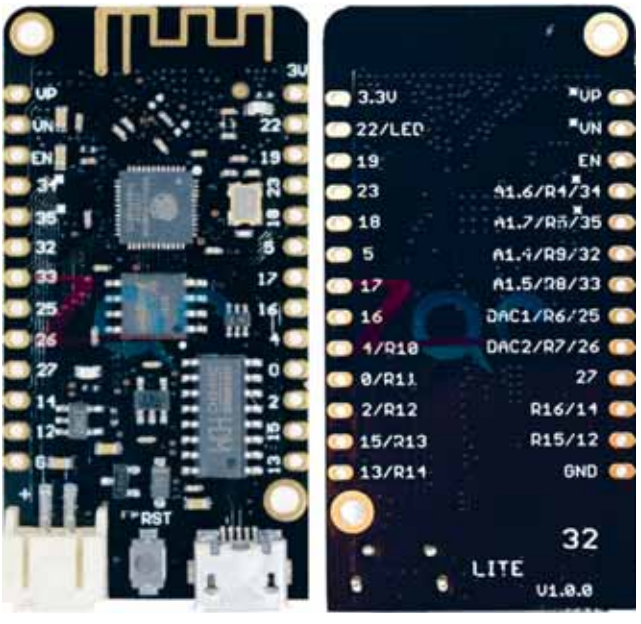


Рис. 1.3. Модуль ESP32 Wemos Lolin Lite

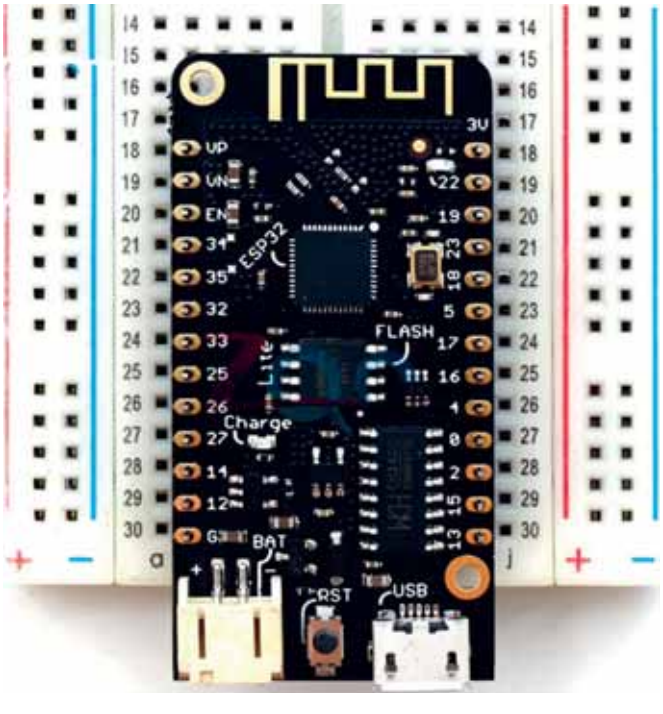


Рис. 1.4. Модуль Wemos Lolin Lite установлен в плату для прототипирования (с обеих сторон есть по одному контактному отверстию для подключения проводов)

Рис. 1.5. Модуль ESP32 LOLIN D32

Рис. 1.6. Модуль ESP32 AI-Thinker с видеокамерой

Функциональная схема ESP32 и его распиновка

Рассмотрим функциональную схему микроконтроллера ESP32 (рис. 1.7). В левом верхнем углу схемы притаился блок **Radio** — ESP32 можно использовать в качестве радиоприемника. Блок управления питанием содержит энергоэффективный сопроцессор, управляющий «спящим режимом». Блок периферии включает IR-интерфейс, температурный сенсор, сенсоры касания. Конвертер digital-to-analog применяется для воспроизведения звука. Остальные блоки мы уже обсудили ранее.

На рис. 1.8 приведена типовая распиновка модуля ESP32 на плате. Контакты GPIO, на которых возможна генерация ШИМ (PWM), обозначены знаком волны ~ , ADC — это контакты GPIO с возможностью аналогового ввода, VSPI и HSPI — интерфейсы для подключения устройств, поддерживающих эти протоколы обмена, SDL и SDA — I^2C-интерфейс, TX0 и RX0 — интерфейс UART0. Еще раз отметим, что некоторые контакты GPIO для стандартных интерфейсов, закрепленные по умолчанию за одними номерами GPIO, можно изменять в программе.

Среды и средства программирования ESP32

Arduino IDE

Программный код управления проектами, представленными в этой книге, основан на языке программирования Wiring (упрощенная и оптимизированная версия C++ для работы с портами ввода/вывода), более того, само программирование осуществляется в среде Arduino IDE, знакомой большинству *мейкеров* (назовем так творческих людей, умеющих воплощать свои идеи) и описанной во многих изданиях.

Однако серия контроллеров ESP32 достаточно универсальна и допускает использование иных сред и средств программирования. Кратко рассмотрим основные из них.

Экосистема Espruino

Espruino — экосистема программирования микроконтроллеров на языке JavaScript. Прошивка Espruino может исполнять JavaScript-код, подготовленный в качестве прикладной программы.

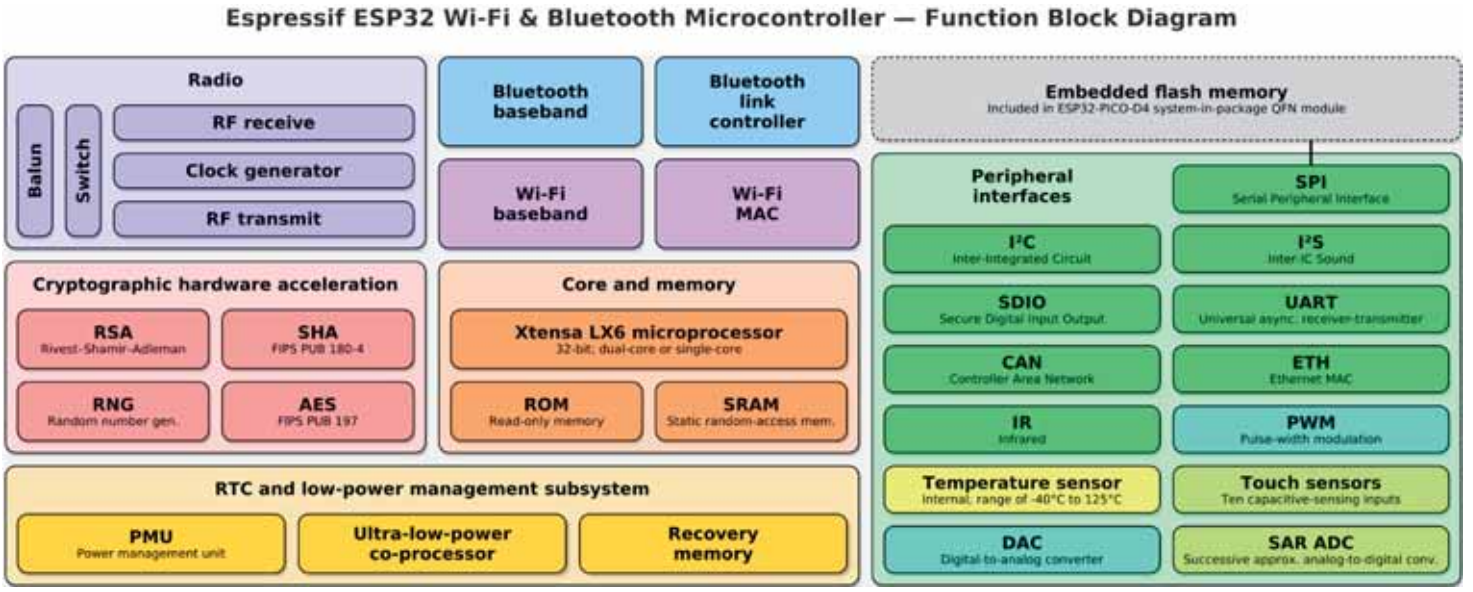


Рис. 1.7. Функциональная схема микроконтроллера ESP32

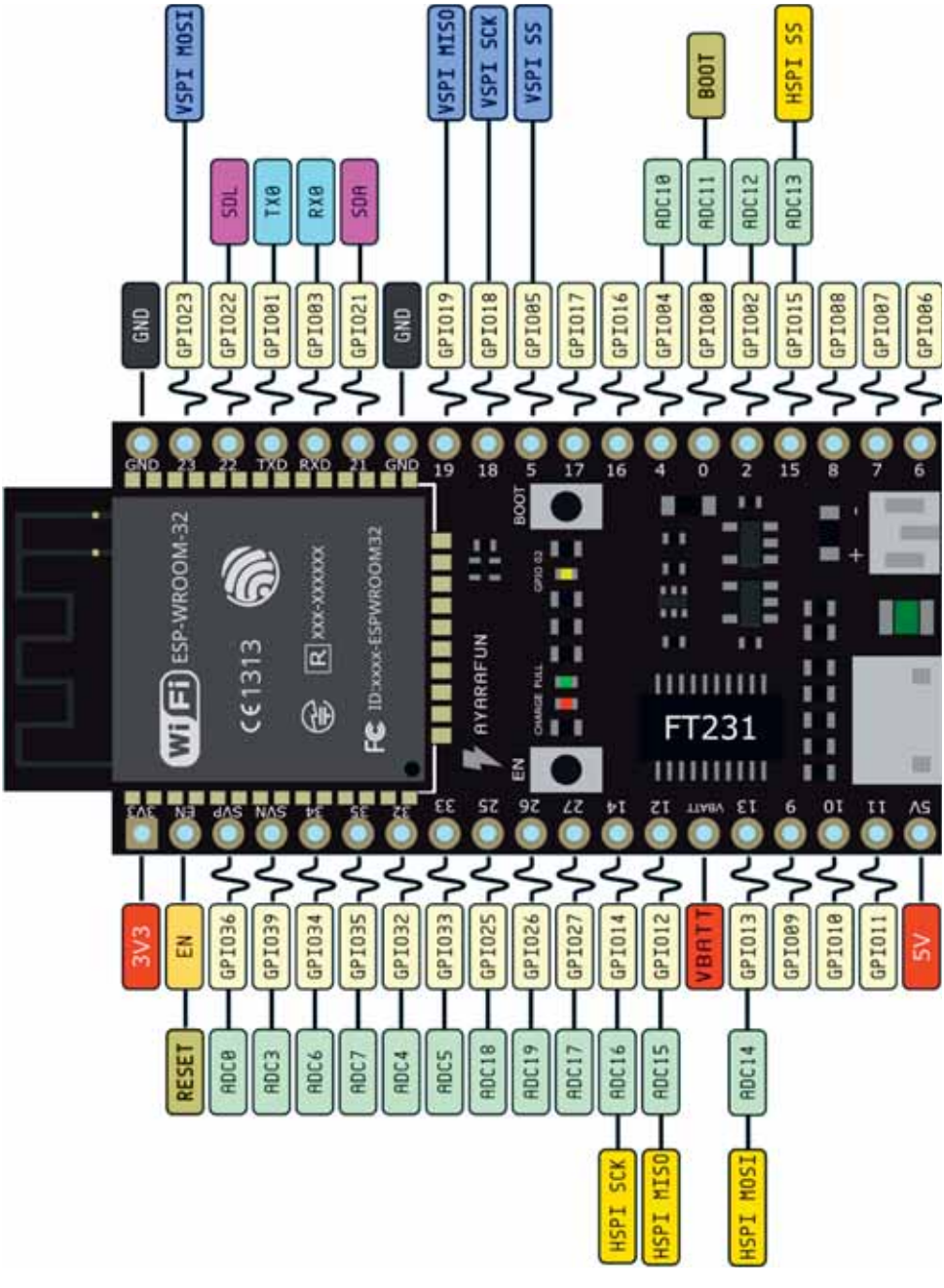


Рис. 1.8. Распиновка ESP32: ADC — аналоговые вводы, GPIO — стандартные контакты ввода/вывода

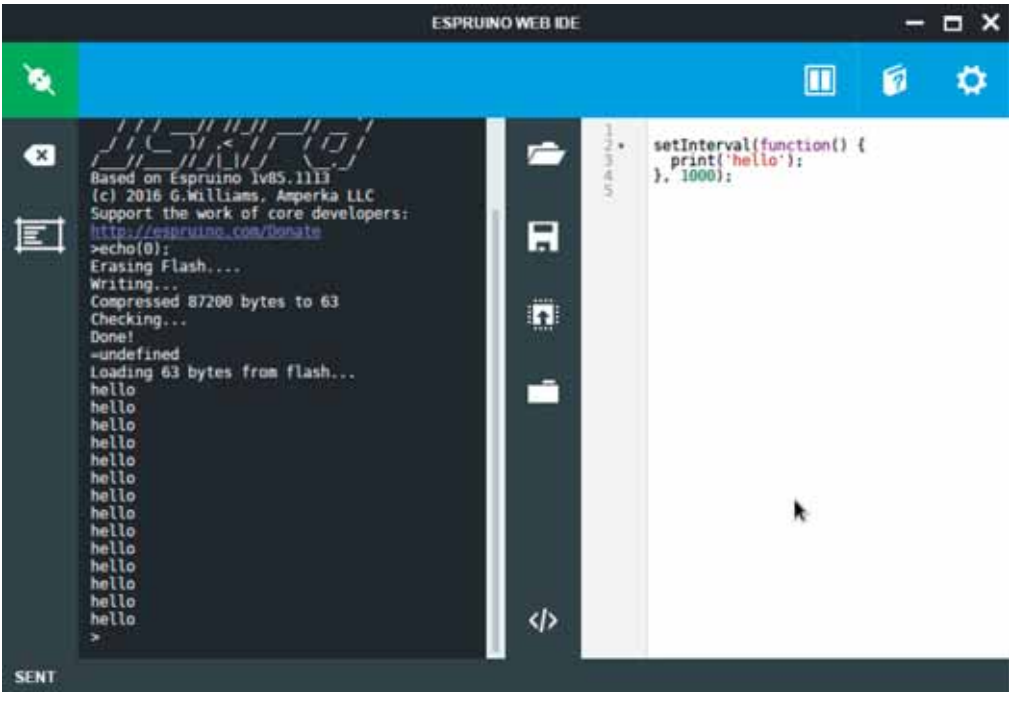


Рис. 1.9. Среда программирования Espruino

Эта экосистема состоит из следующих частей:

- Espruino Web IDE — среда программирования (рис. 1.9);
- Espruino Firmware — JavaScript-машина;
- платы, совместимые с Espruino;
- стандартная библиотека и внешние библиотеки.

Основатель и идеолог проекта Гордон Вильямс выпустил первую Espruino в 2013 году. Функции работы с GPIO в среде похожи на работу с Arduino, благодаря чему разобраться с новой платформой тем, кто знаком с Arduino, не составит труда.

Экосистема Whitecat

IDE Whitecat поддерживает две программные модели: блоки и язык программирования Lua.

Визуальная среда программирования Whitecat (рис. 1.10) схожа со средами Scratch и MIT App Inventor. В настоящее время подобные визуальные среды все чаще изучают в школах, поскольку процесс программирования становится наглядным.