Оглавление

Об авторе

Нил Кэмерон – опытный аналитик и программист с глубоким пониманием работы электронных устройств. Нил – автор книги *Arduino Applied: Comprehensive Projects for Everyday Electronics* (изд-во «Апресс»). Работал научным сотрудником и преподавал в Эдинбургском и Корнелльском университетах.

О техническом рецензенте

Майк МакРобертс – автор книги *Beginning Arduino* (изд-во «Апресс»). Лауреат Pi Wars 2018 и член Medway Makers. Энтузиаст Arduino и Raspberry Pi.

C/C++, Arduino, Python, Processing, JS, Node-Red, NodeJS, Lua

Предисловие

Никогда еще не было так просто получать доступ к информации через интернет: разрабатывать веб-страницы для обновления информации с датчиков, создавать мобильные приложения для удаленного управления устройствами с распознаванием речи или интегрировать Google Maps в приложение для GPS-трекинга. Сочетание беспроводного доступа по Wi-Fi, высокой вычислительной мощности и низкой стоимости микроконтроллеров ESP8266 и ESP32 расширяет спектр возможностей. Связь с устройствами и доступ к информации через интернет с помощью микроконтроллеров ESP8266 и ESP32 в центре внимания книги «Электронные проекты на основе ESP8266 и ESP32».

Первый раздел книги (главы с 1 по 6) демонстрирует мощь и легкость использования микроконтроллеров ESP8266 и ESP32 для получения и отображения интернет-информации. Представленные проекты включают в себя создание интернет-радио, интернет-часов и международной погодной станции, а также проект с камерой ESP32-CAM для загрузки фотографий на веб-страницы.

Второй раздел книги (главы с 7 по 9) посвящен проектам дизайна и обновления в режиме реального времени веб-страниц с информацией от датчиков, представленной в графической форме, или проектам управления удаленным устройством через веб-страницу. Вы узнаете об AJAX (асинхронный JavaScript и XML), который сочетает в себе расширяемый язык разметки XML и протокол передачи гипертекста HTTP, о запросах на обновление веб-страницы с помощью JavaScript, запросах JSON[1] для объединения информации, передаваемой сервером клиенту, о двухсторонней быстрой связи через протокол WebSocket, MQTT-брокеры и IFTTT[2] для связи между устройствами в разных сетях. Практические проекты включают в себя загрузку информации в интернет и управление устройствами из любой точки мира с помощью микроконтроллеров ESP8266 и ESP32.

Мобильные приложения теперь повсеместны, что делает представленные в третьем разделе книги (главы с 10 по 13) проекты очень актуальными. Приложение для управления дистанционно расположенными сервоприводами, подключенными к плате с ESP8266 или ESP32, имитирует робототехнику, используемую в автомобильной промышленности; приложение с распознаванием речи управляет устройствами; приложение для GPS-трекинга, включающее Google Maps, отображает текущую позицию и информацию о маршруте. Каждый проект с микроконтроллерами ESP8266 и ESP32 полностью описан, причем от читателя не требуется предыдущего опыта проектирования и сборки мобильных приложений.

---

[1] JSON (JavaScript Object Notation) – текстовый формат обмена данными, основанный на JavaScript. Позволяет сократить объем кода и выполнять запросы быстрее, чем обычные XML HTTP-запросы. – *Прим. перев.*

[2] Расшифровывается как If This, Then That – *если это, то то-то*; подробнее см. главу 8. – *Прим. перев.*

Связь между микроконтроллерами ESP8266 и ESP32 описана в четвертом разделе книги (главы с 14 по 18). Встроенная система связи ESP-NOW, связь LoRa (дальнего радиуса действия) и радиочастотная RF-связь применяются для управления удаленно расположенными устройствами с помощью информации, обновляемой на веб-странице микроконтроллерами ESP8266 и ESP32. коммуникационные протоколы расширены для приложений генерации сигналов с помощью ESP8266 и ESP32, передающих теперь не только буквенно-цифровой текст, но и звуковые музыкальные сигналы. Генерация сигналов без микроконтроллеров показана на примерах проектов электронного пианино, управления сервоприводом и системы сигнализации, включающей MP3-плеер и детектор движения. Четвертый раздел книги охватывает встроенный протокол связи микроконтроллеров ESP8266 и ESP32 с применением базовой электроники. Глава об электрических измерениях с помощью микроконтроллеров ESP8266 или ESP32, применяемых в проекте солнечной панели, распространяет электронную тему на измерения, чтобы понять методологию, лежащую в основе построения различных датчиков.

Более производительный, чем ESP8266, микроконтроллер ESP32 среди прочего включает связь Bluetooth и ее специальный вариант с низким энергопотреблением Bluetooth Low Energy (BLE). Главы о практических различиях между микроконтроллерами ESP8266 и ESP32 и отдельных особенностях ESP32 составляют последний раздел книги (главы 21 и 22).

На протяжении всей книги описаны все различия в библиотеках или инструкциях для микроконтроллеров ESP8266 и ESP32, так что каждый проект совместим с обоими микроконтроллерами.

Все разделы книги являются самостоятельными, поэтому вы можете углубиться в любой раздел, а не начинать с начала. Несколько глав основываются на информации из предыдущих глав. Например, глава 12 («Приложение для GPS-трекинга с использованием Google Maps») включает в себя дизайн мобильного приложения, связь Bluetooth, получение информации из интернета и обновление страницы в интернете. Предполагается некоторый опыт программирования в среде Arduino IDE, хотя все скетчи описаны полностью и исчерпывающе прокомментированы. Для знакомства с микроконтроллерами, начиная от мигания светодиода и заканчивая созданием автомобиля-робота, рекомендуется книга *Arduino Applied: Comprehensive Projects for Everyday Electronics*[3]. *Принципиальные схемы*[4] *были созданы с помощью программного обеспечения Fritzing (www.fritzing.org)* с акцентом на максимальной наглядности размещения компонентов и минимизации пересекающихся соединений. Авторы библиотек, использованных в книге, указаны в каждой главе, а сведения о библиотеках включены в *приложение*. Все скетчи Arduino IDE и исходный код MIT App Inventor для приложений доступны для загрузки на GitHub (github.com/Apress/ESP8266-and-ESP32). Среда программирования Arduino и библиотеки постоянно обновляются, поэтому информация о последних обновлениях также доступна на сайте GitHub.

---

[3] Для русскоязычных читателей можно порекомендовать следующую литературу: *Ревич Ю.* Азбука электроники. Изучаем Arduino. М.: АСТ, 2017; *Петин В. В., Биняковский А. А.* Практическая энциклопедия Arduino. 2-е изд. М.: ДМК Пресс, 2019. – *Прим. перев.*

[4] Строго говоря, представленные в книге схемы соединений компонентов принципиальными схемами не являются; принципиальные схемы представлены только в отдельных случаях для лучшей иллюстрации отдельных положений. – *Прим. перев.*

Глава 1

Интернет-радио

Интернет-радио – это непрерывная потоковая передача цифрового аудио через интернет. Цифровой звук в формате MP3 принимается микроконтроллером ESP8266 или ESP32 через соединение Wi-Fi. Микроконтроллер ESP8266 или ESP32 взаимодействует с аудиодекодером VS1053 с помощью SPI[5], принятые данные декодируются 18-разрядным цифроаналоговым преобразователем (ЦАП) и превращаются опять в аудиосигнал, который усиливается для вывода на динамик. Микроконтроллеры ESP8266 и ESP32 оборудованы интерфейсом Wi-Fi и имеют достаточную скорость работы процессора для интернет-радио. Для подключения к беспроводной локальной сети (WLAN) требуется SSID[6] сети Wi-Fi и пароль.

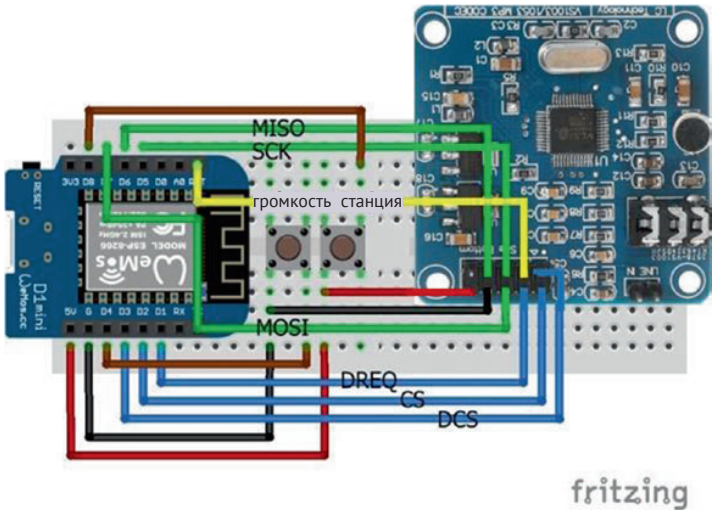

Рисунок 1-1. Интернет-радио с переключателями громкости и станций на основе платы LOLIN (WeMos) D1 mini

[5] SPI (Serial Peripheral Interface, последовательный периферийный интерфейс) – стандарт последовательной синхронной передачи данных для высокоскоростного соединения микроконтроллеров и периферии в масштабах устройства. – *Прим. перев.*

[6] SSID (Service Set Identifier, идентификатор набора услуг) – символьное наименование сети Wi-Fi, связанной с конкретной точкой доступа. Обычно точки доступа непрерывно передают в эфир наименование своей SSID, благодаря чему воспринимающее устройство «видит» все доступные сети в данном месте и может выбрать какую-либо из них для подключения. Существует и скрытый режим, при котором воспринимающее устройство обязано заранее «знать» SSID сети, чтобы подключиться к ней. – *Прим. перев.*

Соединения платы ESP8266 и аудиодекодера VS1053 показаны на рис. 1-1, с подробным описанием на рис. 1-2 и перечислены в табл. 1-1. Соединения для связи по интерфейсу SPI обозначены зеленым цветом, а соединения для передачи данных – синим. Две кнопки (работающие на замыкание), подключенные к прерываниям, управляют громкостью (*volume*) и выбором интернет-радиостанции (*station*). Для платы ESP8266 кнопки переключения громкости и выбора станции на выводах D4 и D8 подключены к GND (черный провод) и 5V (красный провод), так как контакты D4 и D8 подключены к внутренним подтягивающему и заземляющему резисторам соответственно[7]. Соединения для платы на основе ESP32 (а не ESP8266) также приведены в табл. 1-1. При использовании платы ESP32 кнопки громкости и станции обе подключены к GND.

Усилитель и динамик, или мини-колонки, подобные используемым с мобильным телефоном, подключаются к аудиодекодеру VS1053 через обычный аудиоразъем типа «джек»[8].

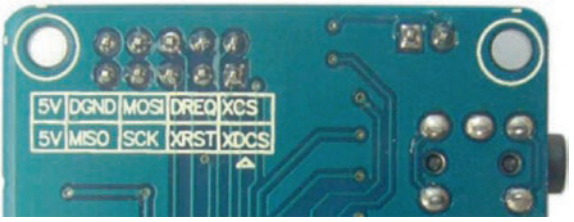

Рисунок 1-2. Соединения аудиоплаты VS1053 (плата перевернута в сравнении с рис. 1-1)

Таблица 1-1. Соединения интернет-радио и кнопок

Компонент	Подключено к ESP8266	Подключено к ESP32
VS1053 5V	5V	VIN или V5
VS1053 DGND	GND	GND
VS1053 MOSI	(MOSI) D7	(MOSI) GPIO 23
VS1053 DREQ (запрос данных)	D1	GPIO 4
VS1053 XCS (*chip select*, выбор кристалла)	D2	GPIO 0
VS1053 MISO	(MISO) D6	(MISO) GPIO 19
VS1053 SCK	(SCK) D5	(CLK) GPIO 18

[7] Контакт D4 платы LOLIN D1 mini соответствует выводу GPIO 2 контроллера ESP8266, а D8 – выводу GPIO16 (см. рис. 21-1 на стр. 621), единственному, имеющему в ESP8266 заземляющий, а не подтягивающий встроенный резистор. Отсюда такой разнобой при установке внешних прерываний для контроллеров ESP8266 и ESP32 (см. далее). – *Прим. перев.*

[8] Разъем типа «джек» (audio jack socket) – круглый разъем с тремя или четырьмя контактами на одной оси. В настоящее время обычно «джеком» называют разъем диаметром 3,5 мм, хотя изначально он имел диаметр 1/4" (6,35 мм), а уменьшенный вариант 3,5 мм именовался «мини-джек». На рис. 1-1 гнездо для «джека» видно над правым нижним крепежным отверстием платы аудиодекодера. – *Прим. перев.*

Окончание табл. 1-1

Компонент	Подключено к ESP8266	Подключено к ESP32
VS1053 XRST (*reset*, сброс)	RST	GPIO EN
VS1053 XDCS (*data chip select*, выбор кристалла данных)	D3	GPIO 2
Левый вывод кнопки громкости (*volume*)	GND	GND
Правый вывод кнопки громкости (*volume*)	D4	GPIO 26
Левый вывод кнопки выбора станций (*station*)	5V	GND
Правый вывод кнопки выбора станций (*station*)	D8	GPIO 27

URL-адрес или веб-адрес интернет-радиостанции можно получить с веб-сайта www.radio.de. Найдите нужную станцию, нажмите кнопку воспроизведения и выберите в меню браузера *View Page Source* (*Исходный код страницы*)[9]. В отображаемом HTML-коде веб-страницы выполните поиск потока (*stream*), который содержится в URL-адресе радиостанции[10]. URL-адрес отформатирован как *хост:порт/путь*. Например, радиостанция Великобритании 1940-х годов имеет URL-адрес *1940sradio1.co.uk:8100/stream/1/*; хост здесь представлен текстом перед первой обратной косой чертой или двоеточием (*1940sradio1.co.uk*), а путь равен оставшемуся тексту (*stream/1/*). Если порт не равен значению 80, которое является портом для просмотра веб-страниц по умолчанию, то его значение следует за двоеточием после хоста, в данном случае, например, 8100.

В скетче интернет-радио с платой ESP8266 (см. листинг 1-1) используется библиотека *VS1053* Эда Смолленбурга (Ed Smallenburg) и Джеймса Колиза (James Coliz), которую можно загрузить в виде zip-файла по адресу *github.com/baldram/ESP_VS1053_Library*. Первый раздел скетча определяет количество интернет-радиостанций и их URL-адреса, инициализирует аудиодекодер, устанавливает соединение Wi-Fi и определяет прерывания. Переменные `newStation` и `newVolume` представлены типом `volatile`, поскольку к ним обращается как основной код программы, так и прерывания. На плате ESP32 вывод кнопки смены станции устанавливается в высокий уровень (*HIGH*) с помощью внутреннего подтягивающего резистора (инструкция `pinMode(statPin, INPUT_PULLUP;)`, а прерывание, подключенное к кнопке станции, устанавливается на срабатывание по падающему фронту (`FALLING`), а не возрастающему (`RISING`), как для платы ESP8266. Микроконтроллеры ESP8266 и ESP32 будут хранить скомпилированный код во внутренней оперативной памяти (IRAM), а не в более медленной флеш-памяти, если добавить к коду атрибут `IRAM_ATTR`. Поэтому обработчик прерывания (ISR[11]) определяется как `IRAM_ATTR void ISR()`, а не просто `void ISR()`.

---

[9] В большинстве браузеров этот пункт размещается в главном меню (клавиша **F10** или три точки в правом верхнем углу) примерно по следующему пути: *Инструменты (Дополнительные инструменты)* > *Инструменты веб-разработки* > *Исходный код страницы*. – *Прим. перев.*

[10] URL (Uniform Resource Locator, единый локатор ресурсов) – утвержденная форма представления интернет-адреса в виде буквенной строки, понятной для человека. – *Прим. перев.*

[11] Interrupt Service Routine, букв. *процедура обслуживания прерываний*. – *Прим перев.*

В функции `loop()` устанавливается соединение с веб-сайтом интернет-радиостанции, и аудиодекодер VS1053 обрабатывает данные в 32-байтовых пакетах. Две процедуры обслуживания прерываний, `chan` и `vol`, осуществляют переход к следующей радиостанции и увеличение громкости соответственно. Шкала громкости составляет от 0 до 100 %. Библиотека *VS1053* ссылается на библиотеку SPI, и инструкция `#include <SPI.h>` не требуется.

Подключение к серверу интернет-радиостанции с помощью команды `connect (host[station], port[station])` сопровождается HTTP-запросом. Библиотека VS1053 использует протокол HTTP для связи между клиентом и сервером интернет-радиостанции. Клиент отправляет HTTP-запрос на сервер для получения аудиоданных, и сервер отправляет клиенту ответ с требуемыми данными. За инструкциями HTTP-запроса "`GET pathname HTTP/1.1`" и "`Host: hostname`" следует инструкция по закрытию подключения "`Connection: close`". На примере радиостанции Великобритании 1940-х годов инструкции запроса будут следующими:

```
GET stream/1/HTTP/1.1
Host: 1940sradio1.co.uk
Connection: close
<\r\n>
```

Обратите внимание, что требуется четвертая команда – возврата каретки `\r`, и новой строки `\n`, что эквивалентно инструкции `println()`.

Листинг 1-1. Интернет-радио с переключателями громкости и станций для платы ESP8266

```
#include <VS1053.h>                          // include VS1053 library
#include <ESP8266WiFi.h>                     // include ESP8266WiFi library
int CS = D2;
int DCS = D3;                                // define VS1053 decoder pins
int DREQ = D1;
VS1053 decoder(CS, DCS, DREQ);               // associate decoder with VS1053
int statPin = D8;                            // define switch pins for
int volPin = D4;                             // station and volume
WiFiClient client;                           // associate client and library
char ssid[] = "xxxx";                        // change xxxx to Wi-Fi ssid
char password[] = "xxxx";                    // change xxxx to Wi-Fi password
const int maxStat = 4;                       // number of radio stations
String stationName[] = {"1940 UK", "Bayern3", "ClassicFM", "BBC4"};
char * host[maxStat] = {"1940sradio1.co.uk", // station host
                        "streams.br.de",
                        "media-ice.musicradio.com",
                        "bbcmedia.ic.llnwd.net"};
char * path[maxStat] = {"/stream/1/",        // station path
                        "/bayern3_2.m3u",
                        "/ClassicFMMP3",
                        "/stream/bbcmedia_radio4fm_mf_q"};
int port[] = {8100,80,80,80};                // default station port is 80
unsigned char mp3buff[32];                   // VS1053 loads data in 32 bytes
int station = 0;
int volume = 0;                              // volume level 0-100
volatile int newStation = 2;                 // station number at start up
volatile int newVolume = 80;                 // volume at start up
void setup ()
```

```
{
   Serial.begin(115200);              // Serial Monitor baud rate
   SPI.begin();                       // initialise SPI bus
   decoder.begin();                   // initialise VS1053 decoder
   decoder.switchToMp3Mode();         // MP3 format mode
   decoder.setVolume(volume);         // set decoder volume
   WiFi.begin(ssid, password);        // initialise Wi-Fi
   while (WiFi.status() != WL_CONNECTED) delay(500);
   Serial.println("WiFi connected");  // wait for Wi-Fi connection
   pinMode(volPin, INPUT_PULLUP);     // switch pin uses internal
   // pull-up resistor
   attachInterrupt(digitalPinToInterrupt(statPin), chan, RISING);
   attachInterrupt(digitalPinToInterrupt(volPin), vol, FALLING);
}                            // define interrupts for changing station and volume
void loop()
{
    if(station != newStation)         // new station selected
    {
       station = newStation;          // display updated station name
       Serial.print("connecting to CH"); Serial.print(station);
       Serial.print(" ");Serial.println(stationName[station]);
       if(client.connect(host[station], port[station]))
       {                              // connect to radio station URL
          client.println(String("GET ")+ path[station] + " HTTP/1.1");
          client.println(String("Host: ") + host[station]);
          client.println("Connection: close");
          client.println();           // new line is required
       }
    }
    if(volume != newVolume)           // change volume selected
    {
       volume = newVolume;            // display updated volume
       Serial.print("volume ");Serial.println(volume);
       decoder.setVolume(volume);     // set decoder volume
    }
    if(client.available() > 0)        // when audio data available
    {                                 // decode data 32 bytes at a time
       uint8_t bytesread = client.read(mp3buff, 32);
       decoder.playChunk(mp3buff, bytesread);
    }
    }
IRAM_ATTR void chan()                 // ISR to increment station number
{
   newStation++;
   if(newStation > maxStat-1) newStation = 0;
}                                     // stations numbered 0, 1, 2...
IRAM_ATTR void vol()                  // ISR to increase volume
{
   newVolume = newVolume + 5;
   if(newVolume > 101) newVolume = 50;
}                                     // maximum volume is 100
```

В случае использования платы ESP32 ее подключения к аудиодекодеру VS1053 показаны на рис. 1-3 и 1-2 соответственно, а также приведены в табл. 1-1. Оба контакта переключателя подключены к внутренним под-

тягивающим резисторам, поэтому оба прерывания активируются падающим фронтом (FALLING). Единственные изменения в листинге 1-1, помимо определения выводов декодера, кнопок выбора станции и регулятора громкости, – используется библиотека *WiFi*, а не библиотека *ESP8266WiFi*, и инструкция `pinMode (statPin, INPUT_PULLUP)` для смены условия прерывания на выводе переключателя станции вместо возрастающего фронта (RISING) на падающий (FALLING).

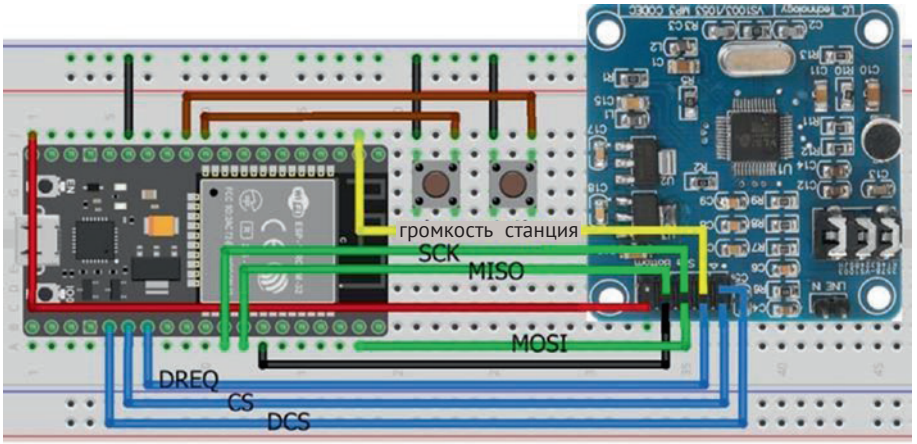

Рисунок 1-3. Интернет-радио с переключателями громкости и станций на основе платы ESP32

Выбор и отображение станции

В листинге 1-1 выбор станции и регулировка громкости активируются кнопками, при этом информация о станции и громкости отображается через монитор последовательного порта среды Arduino. Для портативного интернет-радио информация о станции и громкости отображается на TFT ЖК-дисплее ST7735, и выбор станции или регулировка громкости осуществляется с помощью поворотного энкодера (см. рис. 1-4 и 1-5 и подключения в табл. 1-2). Обратите внимание, что как поворотный энкодер, так и ЖК-экран ST7735 подключены к напряжению 3,3 В, только аудиодекодер VS1053 подключен к выводу 5V. Микроконтроллер ESP32 взаимодействует как с аудиодекодером VS1053, так и с ЖК-экраном ST7735 с помощью SPI, поэтому микроконтроллер имеет одни и те же подключения MOSI и SCK к аудиодекодеру и экрану, но соединения CS зависят от конкретного устройства[12].

[12] Расшифровка названий соединений SPI: MOSI – Master Output, Slave Input (выход ведущего, вход ведомого); SCK – Serial Clock (тактовый сигнал); MISO – Master Input, Slave Output (вход ведущего, выход ведомого); CS – Chip Select (выбор кристалла, т. е. микросхемы). Вывод CS обычно имеет отрицательную логику (активируется низким уровнем), поэтому часто указывается с надстрочной чертой. – *Прим. перев.*

Рисунок 1-4. Скриншоты дисплея интернет-радио

В скетче используется библиотека *ESP32 vs1053_ext* от Wolle, которая загружается в виде zip-файла с *github.com/schreibfaul1/ESP32-vs1053_ext*. Библиотека *ESP32 vs1053_ext* предназначена для микроконтроллера ESP32, в то время как библиотека *VS1053* Эда Смолленбурга (Ed Smallenburg) и Джеймса Колиза (James Coliz) совместима как с микроконтроллерами ESP8266, так и с ESP32. Библиотека *ESP32 vs1053_ext* предоставляет информацию о станции и треке, в том числе название трека. Инструкция для подключения к серверу интернет-радиостанции выглядит как `connecttohost("host:port/stream")`, например `connecttohost("1940sradio1.co.uk:8100/stream/1/")`. Номер порта требуется только в том случае, если он не равен значению по умолчанию 80. Функции `vs1053_showstation`, `vs1053_icyurl`, `vs1053_bitrate` и `vs1053_ showstreamtitle` содержат название интернет-радиостанции, URL-адрес домашней страницы, скорость передачи данных и название трека. При передаче нового трека автоматически обновляется функция `vs1053_showstreamtitle`. Переменная громкости `volume` имеет 22 уровня 0, 50, 60, 65, 70, 75, 80, 82…90, 91…100 %; при этом значению 10 соответствует уровень громкости 88 %, тогда как значение 0 соответствует громкости 0 %.

В листинге 1-2 демонстрируются выходные данные функций библиотеки *ESP32 vs1053_ext*, которые используются в листинге 1-3 для отображения информации об интернет-радиостанции и треке.

Листинг 1-2. Библиотечные функции ESP32 vs1053_ext

```
#include <vs1053_ext.h>                 // include ESP32 VS1053_ext lib
#include <WiFi.h>                       // include Wi-Fi library
int CS = 0;
int DCS = 2;                            // define VS1053 decoder pins
int DREQ = 4;
VS1053 decoder(CS, DCS, DREQ);          // associate decoder with VS1053
char ssid[] = "xxxx";                   // change xxxx to Wi-Fi ssid
char password[] = "xxxx";               // change xxxx to Wi-Fi password
int volume = 10;                        // volume level

void setup()
{
   Serial.begin(115200);                // Serial Monitor baud rate
```

```
    SPI.begin();                          // initialise SPI bus
    WiFi.begin(ssid, password);           // initialise Wi-Fi
    while (WiFi.status() != WL_CONNECTED) delay(500);
    decoder.begin();                      // initialise VS0153 decoder
    decoder.setVolume(volume);            // set decoder volume level
    decoder.connecttohost
    ("media-ice.musicradio.com:80/ClassicFMMP3");
}

void loop()
{
    decoder.loop();
}
void vs1053_showstation(const char * info)
{                                         // display radio station name
    Serial.print("Station: ");
    Serial.println(info);
}
void vs1053_bitrate(const char * info)
{                                         // display streaming bit rate
    Serial.print("Bit rate: ");
    Serial.println(String(info)+"kBit/s");
}
void vs1053_icyurl(const char * info)
{                                         // display radio station URL
    Serial.print("Homepage: ");
    Serial.println(info);
}
void vs1053_showstreamtitle(const char * info)
{                                         // title of streamed track
    Serial.print("Stream title: ");
    Serial.println(info);
}
```

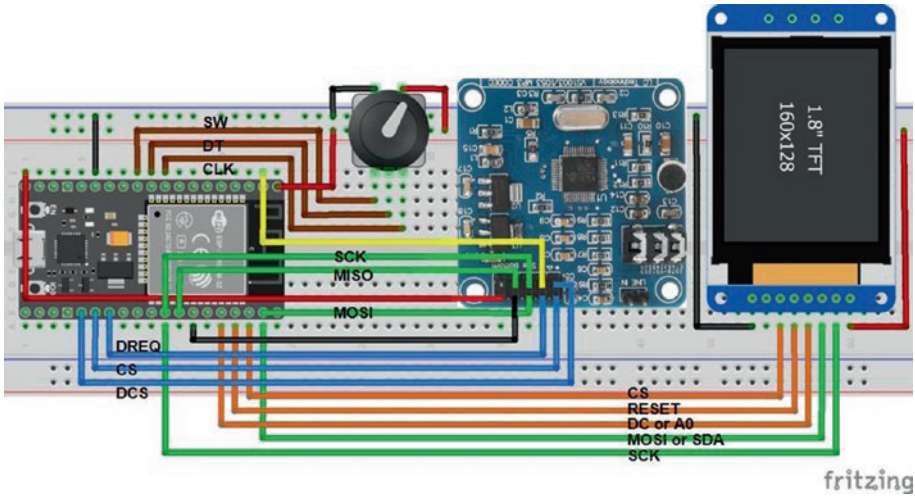

Рисунок 1-5. Интернет-радио с экраном, поворотным энкодером и платой ESP32

Таблица 1-2. Интернет-радио с экраном, поворотным энкодером и платой ESP32

Компонент	Подключено к EPS32
VS1053 audio decoder	См. табл. 1-1
Поворотный энкодер CLK	GPIO 25
Поворотный энкодер DT	GPIO 26
Поворотный энкодер SW	GPIO 27
Поворотный энкодер VCC	3V3
Поворотный энкодер GND	GND
ST7735 TFT LCD GND	GND
ST7735 TFT LCD CS	GPIO 22
ST7735 TFT LCD RESET	GPIO 1
ST7735 TFT LCD DC или A0	GPIO 3
ST7735 TFT LCD SDA	GPIO 23
ST7735 TFT LCD SCK	GPIO 18
ST7735 TFT LCD LED	3V3

Скетч для портативного интернет-радио приведен в листинге 1-3. При однократном нажатии переключателя, связанного с валом поворотного энкодера, отображается меню доступных радиостанций с регулировкой громкости в качестве первого пункта меню. Поворот энкодера перемещает меню радиостанций вверх или вниз по экрану ST7735. Станция в середине экрана, выделенная красным цветом, выбирается повторным нажатием на вал; и HTTP-запрос отправляется на сервер интернет-радиостанции для получения аудиоданных. Когда в меню выбирается *Volume*, отображается текущий уровень громкости, и поворот энкодера уменьшает или увеличивает этот уровень, который затем фиксируется нажатием на вал. На ЖК-экране ST7735 обновляется информация о текущей радиостанции и отображается обновленный уровень громкости, но меню радиостанции по-прежнему показывает текущую радиостанцию.

Скетч в листинге 1-3 состоит из нескольких функций для разделения инструкций. Длинный первый раздел скетча определяет библиотеки, URL-адреса интернет-радиостанций, номера выводов для аудиодекодера VS1053, ЖК-экрана ST7735, а также параметры поворотного энкодера, включая начальные значения для станции и уровня громкости. Библиотека *Adafruit ST7735* доступна в среде Arduino IDE. Библиотеки *ESP32 vs1053_ext* и *Adafruit ST7735* ссылаются на библиотеки *SPI* и *Adafruit GFX*, поэтому инструкции `#include <SPI. h>` и `#include <Adafruit_GFX.h>` не требуются. Функция `setup` устанавливает соединение Wi-Fi, инициализирует аудиодекодер VS1053 и ЖК-экран ST7735, подключает внутренние подтягивающие резисторы к контактам энкодера и определяет прерывания для них. Направление и количество оборотов поворотного энкодера определяются прерыванием по изменению уровня, как описано в главе 19 («Поворотный энкодер»).

При нажатии на вал замыкаются контакты переключателя энкодера, функция `loop` вызывает функцию `screen` для отображения меню громкости и радиостанции, функцию `readMenu` для определения выбранной радиостанции или функцию `readValue` для получения нового уровня громкости, а затем функцию `radio`. Функция `radio` либо подключает устройство к серверу выбранной радиостанции, либо изменяет громкость на аудиодекодере VS1053. Функции `readMenu` и `readValue` определяют номер выбранной строки меню, представляющего собой список станций, и выбранный уровень громкости, когда энкодер поворачивается. Функция `vs1053_icyurl` получает строку, начинающуюся с *https://*, за которой следует URL-адрес станции, и извлекает подстроку, начинающуюся через две позиции после расположения первой обратной косой черты. Функции `vs1053_showstation` и `vs1053_showstreamtitle` получают название радиостанции и название трека, а затем вызывают функцию `showStation`, которая отображает название станции, название трека, значение громкости и информацию об URL-адресе станции на ЖК-дисплее ST7735. Некоторый текст, такой как название станции или заголовок транслируемого трека, окажется длиннее ширины экрана ST7735, поэтому функция `lines` разбивает название или заголовок станции на подстроки размером с экран. Функции `encoder` и `swPress` подсчитывают направление и количество оборотов поворотного энкодера, а также количество нажатий переключателя энкодера.

Листинг 1-3. Интернет-радио с экраном и поворотным энкодером и платой ESP32

```
#include <vs1053_ext.h>              // include ESP32 VS1053_ext,
#include <WiFi.h>                    // WiFi and
#include <Adafruit_ST7735.h> // Adafruit_ST7735 libraries
int CS = 0;
int DCS = 2;                         // define VS1053 decoder pins
int DREQ = 4;
VS1053 decoder(CS, DCS, DREQ);       // associate decoder with VS1053
char ssid[] = "xxxx";                // change xxxx to Wi-Fi ssid
char password[] = "xxxx";            // change xxxx to Wi-Fi password
const int maxStation = 11;   // number of radio stations
String stationName[] = {"Volume", // first item on menu
"1940 UK", "Berlin", "Bayern3", "Classic", "BBC4",
"Vermont", "Ketchikan", "Kathmandu", "Ithaca", "Trondeim",
"Virgin"};
char * URL[maxStation] = {   // radio station URLs
"1940sradio1.co.uk:8100/1",
"streambbr.ir-media-tec.com/berlin/mp3-128/vtuner_web_mp3/",
"streams.br.de/bayern3_2.m3u",
"media-ice.musicradio.com:80/ClassicFMMP3",
"bbcmedia.ic.llnwd.net/stream/bbcmedia_radio4fm_mf_q",
"vpr.streamguys.net/vpr64.mp3",
"96.31.83.94:8082/stream",
"streaming.softnep.net:8037/stream.nsv",
"17993.live.streamtheworld.com/WITHFM.mp3",
"stream.radiometro.no/metro128.mp3",
"radio.virginradio.co.uk/stream"
};
int TFT_CS = 22;
int DCpin = 3;               // define ST7735 TFT screen pins
```

```
int RSTpin = 1;              // associate tft with Adafruit ST7735
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, DCpin, RSTpin);
int CLKpin = 25;
int DTpin = 26;
int SWpin = 27;              // define rotary encoder pins
int oldRow = 0;
int newRow = 1;
int menuItem, val, upLimit;
int displayVol[] =           // define volume values for 22 levels
{0,50,60,65,70,75,80,82,84,86,88,90,91,92,93,94,95,96,97,98,
99,100};
int volume = 0;
int newVolume = 10;          // volume level at start up
int station = 0;
int newStation = 3;          // station level at start up
int textlen, textrows;
String showstatn, showtitle, showurl, text, text1, text2;
volatile int change = 0;     // rotary encoder variables
volatile int pressed = 0;
volatile int vals[] = {0,-1,1,0,1,0,0,-1,-1,0,0,1,0,1,0,-1,0};
volatile int score = 0;
volatile int oldState = 0;
volatile int turn;
void setup()
{
  SPI.begin();                        // initialise SPI bus
  WiFi.begin(ssid, password);         // initialise Wi-Fi
  while (WiFi.status() != WL_CONNECTED) delay(500);
  decoder.begin();                    // initialise VS0153 decoder
  decoder.setVolume(volume);          // set decoder volume level
  tft.initR(INITR_BLACKTAB);          // initialise screen
  tft.fillScreen(ST7735_BLACK);       // clear screen
  tft.setRotation(1);                 // orientate ST7735 screen
  tft.setTextSize(2);                 // set screen text size
  tft.drawRect(0,0,158,126,ST7735_WHITE); // draw white frame line
  tft.drawRect(2,2,154,122,ST7735_RED); // and second frame line
  pinMode(CLKpin, INPUT_PULLUP);
  pinMode(DTpin, INPUT_PULLUP);       // rotary encoder uses
  pinMode(SWpin, INPUT_PULLUP);       // internal pull-up resistors
  attachInterrupt(CLKpin, encoder, CHANGE);
  attachInterrupt(DTpin, encoder, CHANGE); // attach rotary encoder interrupts
  attachInterrupt(SWpin, swPress, CHANGE);
}
void loop()
{
  if(pressed == 1)                    // switch pin pressed first time
  {                                   // to change station or volume
    clearScreen();           // call clearScreen function
    screen();                         // call screen function
    menuItem = readMenu(maxStation); // selected row in menu
  }
  else if (pressed == 2)              // switch pin pressed second time
  {                                   // to select station
    if(menuItem > 0)                  // station selected
    {
      newStation = menuItem-1;        // selected station in menu
```

```
        clearScreen();                // call clearScreen function
        showStation(volume, showstatn, showtitle);
                                       // call showStation function
        if(newStation == station) showStation(volume, showstatn,
        showtitle);
      }                               // volume change selected
      else if(menuItem== 0) newVolume = readValue("volume: ",
      volume, 21, 1);
      pressed = 0;                    // reset variable
    }
    else if(pressed > 2) pressed = 0; // volume changed
    radio();                          // call radio function
}
void radio()                  // function to connect to
{                              // selected radio station server
    if(station != newStation)  // new station selected
    {
      clearScreen();          // call clearScreen function
      station = newStation;
      showurl = "";
      decoder.connecttohost(URL[station]);
    }                         // connect to radio station server
    if(volume != newVolume)    // new volume level selected
    {
      volume = newVolume;
      newRow = station+1;             // retain station number on menu
      decoder.setVolume(volume);      // update VS1053 volume
      clearScreen();          // call clearScreen function
      showStation(volume, showstatn, showtitle);
    }                                 // call showStation function
    decoder.loop();
}
    int readMenu (int rows)           // function to obtain station
    {                                 // number on menu
      while(pressed < 2)              // while station not selected
      {
        if(change != 0)               // rotary encoder turned
        {
          newRow = oldRow + change; // retain row number on menu
          newRow = constrain(newRow, 0, rows);
          clearScreen();              // call clearScreen function
          screen();                   // call screen function
          oldRow = newRow;
          change = 0;
        }
        delay(10);
      }
      return newRow;                  // return row number on menu
}
                                      // function to obtain volume level
int readValue(String text, int current, int upLimit, int gain)
{
    val = current;                    // current volume level
    clearScreen();                    // call clearScreen function
    tft.setTextColor(ST7735_WHITE);
    tft.setTextSize(2);
```

```
   tft.setCursor(10, 50);
   tft.print(text);                    // display text and
   tft.print(displayVol[val]);         // current volume value
   while(pressed < 3)                  // while switch pin is not pressed
   {
      if(change != 0)                  // rotary encoded turned
      {
         val = val + change * gain; // increment volume level
         val = constrain(val, 0, upLimit);
                                       // constrain volume level
         clearScreen();                // call clearScreen function
         tft.setCursor(10, 50);
         tft.print(text);              // display text and
         tft.print(displayVol[val]);   // new volume value
         change = 0;
      }
      delay(10);
   }
   return val;                         // return new volume level
}
void vs1053_showstation(const char * info)
{                                      // function to obtain station name
   showstatn = String(info);  // station name
   showtitle = "";
   if(showstatn == "No Name") showstatn = stationName[station+1];
   clearScreen();
   showStation(volume, showstatn, showtitle);
}                             // call showStation function
void vs1053_showstreamtitle(const char * info)
{                             // function to obtain streamed title
   showtitle = String(info);
   clearScreen();
   showStation(volume, showstatn, showtitle);
}
void vs1053_icyurl(const char * info)
{                                      // function to obtain station URL
   showurl = String(info);
   int i = showurl.indexOf("/");       // position of first / in string
   showurl = showurl.substring(i+2); // station URL as substring
   clearScreen();
   showStation(volume, showstatn, showtitle);
}
void showStation(int volume, String showstatn, String showtitle)
{                             // function to display station name, streamed title
                              // and station URL on screen
   tft.setTextColor(ST7735_GREEN);
   tft.setTextSize(1);
   lines(showstatn, 10);      // lines function to display station
   tft.setTextColor(ST7735_YELLOW);
   lines(showtitle, 40);      // lines function to display title
   tft.setTextColor(ST7735_GREEN);
   tft.setCursor(80, 100);    // display volume value
   tft.print("volume: ");tft.print(displayVol[volume]);
   tft.setCursor(5, 110);
   tft.print(showurl);                 // display URL
}
void lines(String text, int line)
```

```
{                    // function to split string into screen sized substrings
  textlen = text.length();           // get string length
  textrows = 1+textlen/23;           // required number of screen rows
  for(int i=0; i<textrows; i++)
  {
    tft.setCursor(10, line + i*10); // move cursor to next row
    tft.println(text.substring(i*23, (i+1)*23));
  }                                  // display substring
}
void screen()                        // function to display station menu
{
  tft.setTextSize(2);
  tft.setTextColor(ST7735_RED);      // selected station in RED
  tft.setCursor(20, 55);
  tft.print
  (stationName[newRow]);             // display station name
  tft.setTextSize(1);
  tft.setTextColor(ST7735_WHITE);     // all other stations in WHITE
  for (int i=1; i<4; i++)            // display other station names
  {
    tft.setCursor(30, 50 - i*12); // above selected station
    if(newRow-i >=0) tft.print(stationName[newRow-i]);
    tft.setCursor(30, 65 + i*12); // below selected station
    if(newRow+i < maxStation+1) tft.print(stationName
    [newRow+i]);
  }
}
void clearScreen()                   // function to clear screen
{                                    // by displaying a BLACK rectangle
  tft.fillRect(3,3,152,120,ST7735_BLACK);
}
IRAM_ATTR void encoder()             // function to count rotary
{ // encoder turns
  int newState = (oldState<<2)+(digitalRead(CLKpin)<<1)
  +digitalRead(DTpin);
  score = score + vals[newState]; // allocate score from array
  oldState = newState % 4; // remainder to leave new CLK and DT
  if(score == 2 || score == -2)      // 2 steps for complete rotation
  {
    change = score/2;                // unit change per two steps
    score = 0;                       // reset score
  }
}
IRAM_ATTR void swPress()             // function to count switch presses
{                                    // pressed = 1, 2, 3 to change station,
                                     // station selected, volume changed
  if(digitalRead(SWpin) == HIGH) pressed = pressed + 1;
}
```

Простейшее интернет-радио

Скетч в листинге 1-3, включающий ЖК-экран ST7735 для отображения названия и URL-адреса радиостанции, названия потокового трека и уровня громкости с поворотным кодером для выбора станции и регулировки громкости, состоит из 250 строк кода. Напротив, в листинге 1-4 для минимальной настройки

интернет-радио на одну радиостанцию с одним значением громкости содержится всего 21 строка кода. Просто измените URL-адрес интернет-радиостанции в скетче на требуемый URL-адрес!

Листинг 1-4. Простейшее интернет-радио

```
#include <vs1053_ext.h>                        // include ESP32 VS1053_ext
#include <WiFi.h>                      // and WiFi libraries
int CS = 0;
int DCS = 2;                           // define VS1053 decoder pins
int DREQ = 4;
VS1053 decoder(CS, DCS, DREQ);         // associate decoder with VS1053
char ssid[] = "xxxx";                          // change xxxx to Wi-Fi ssid
char password[] = "xxxx";              // change xxxx to Wi-Fi password
void setup()
{
  SPI.begin();                                 // initialise SPI bus
  WiFi.begin(ssid, password);                  // initialise Wi-Fi
  while (WiFi.status() != WL_CONNECTED) delay(500);
  decoder.begin();                     // initialise VS0153 decoder
  decoder.setVolume(10);               // pre-set decoder volume level
  decoder.connecttohost
  ("media-ice.musicradio.com:80/ClassicFMMP3");
}                                      // connect to pre-set radio station server
void loop()
{
  decoder.loop();
}
```

Итоги

Интернет-радио было построено с использованием аудиодекодера VS1053 и микроконтроллера ESP8266 или ESP32, с возможностью выбора радиостанции и регулировки громкости с помощью кнопок. Портативное интернет-радио состояло из аудиодекодера VS1053, платы ESP32 и ЖК-дисплея ST7735 для отображения сведений о радиостанции, названия транслируемого трека и уровня громкости с поворотным энкодером для управления выбором станции и громкостью. Скетч для минимального интернет-радио состоял всего из 21 строки кода.

Перечень компонентов

- Микроконтроллер ESP8266: плата LOLIN D1 mini (WeMos) или NodeMCU
- Микроконтроллер ESP32: плата ESP32 DEVKIT DOIT или NodeMCU
- Модуль аудиодекодера VS1053
- Миниатюрные аудиоколонки
- Тактовые кнопки: 2 шт.
- Поворотный энкодер KY-040
- Дисплей TFT LCD ST7735, 1,8 дюйма