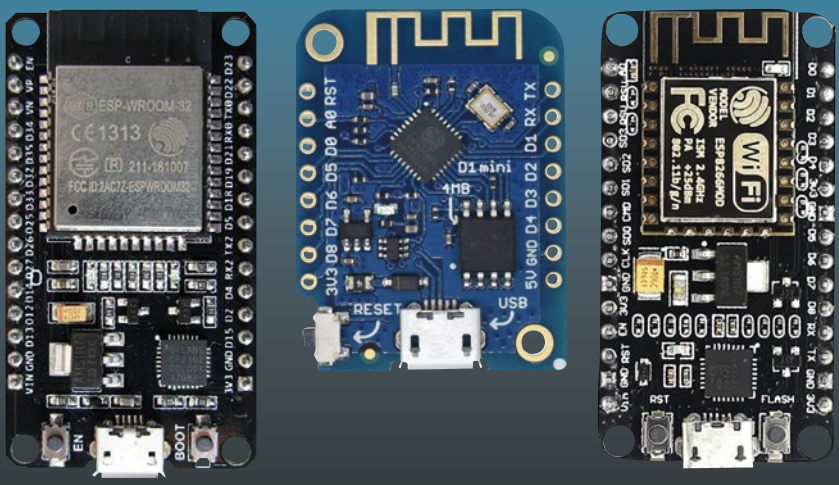
TECHNOLOGY IN ACTION™




# Electronics Projects with the ESP8266 and ESP32


Building Web Pages, Applications, and WiFi Enabled Devices

—

Neil Cameron

Apress®

# Electronics Projects with the ESP8266 and ESP32

## Building Web Pages, Applications, and WiFi Enabled Devices

**Neil Cameron**

Apress®

*Electronics Projects with the ESP8266 and ESP32: Building Web Pages, Applications, and WiFi Enabled Devices*

Neil Cameron
Edinburgh, UK

ISBN-13 (pbk): 978-1-4842-6335-8 ISBN-13 (electronic): 978-1-4842-6336-5
https://doi.org/10.1007/978-1-4842-6336-5



Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Natalie Pao
Development Editor: James Markham
Coordinating Editor: Jessica Vakili

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 NY Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/978-1-4842-6335-8. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

# Table of Contents

**About the Author** ..... xi

**About the Technical Reviewer** ..... xiii

**Preface** ..... xv

**Chapter 1: Internet radio** ..... 1

Station display and selection ..... 9

Minimal Internet radio ..... 24

Summary ..... 25

Components List ..... 25

**Chapter 2: Intranet camera** ..... 27

Save images to the SD card ..... 30

Load images on a web page ..... 36

Stream images to a web page ..... 40

PIR trigger to stream images to a web page ..... 45

Summary ..... 49

Components List ..... 50

**Chapter 3: International weather station** ..... 51

ILI9341 SPI TFT LCD touch screen ..... 52

Touch screen calibration ..... 57

Painting on-screen ..... 60

ESP8266-specific touch screen calibration and paint ..... 62

Weather data for several cities ..... 69

Summary....84
Components List....84

**Chapter 4: Internet clock....85**

WS2812 RGB LEDs responsive to sound....90
ESP8266 and multiplexer....94
LED rings clock....99
Network Time Protocol....105
ESP32 and Internet clock....109
Summary....110
Components List....111

**Chapter 5: MP3 player....113**

Control command for the MP3 player....115
MP3 player control with a microcontroller....117
Infrared remote control of an MP3 player....126
Creating sound tracks and two alarm systems....132
Movement detection alarm....138
Speaking clock....141
Voice recorder....147
Summary....149
Components List....150

**Chapter 6: Bluetooth speaker....151**

Summary....157
Components List....157

**Chapter 7: Wireless local area network....159**

HTTP request....162
HTML code....168

XML HTTP requests, JavaScript, and AJAX ..... 171
Summary ..... 180
Components List ..... 180

**Chapter 8: Updating a web page ..... 181**

XML HTTP requests, JavaScript, and AJAX ..... 188
JSON ..... 192
Accessing WWW data ..... 196
MQTT broker and IFTTT ..... 202
Parsing text ..... 216
Console log ..... 219
Wi-Fi connection ..... 220
Access information file ..... 221
Summary ..... 222
Components List ..... 222

**Chapter 9: WebSocket ..... 223**

Remote control and WebSocket communication ..... 229
WebSocket and AJAX ..... 237
Access images, time, and sensor data over the Internet ..... 243
Summary ..... 255
Components List ..... 255

**Chapter 10: Build an app ..... 257**

Control and feedback app ..... 259
Install the app ..... 271
Servo-robot control app ..... 271
Speech recognition app ..... 281
Summary ..... 286
Components List ..... 287

**Chapter 11: App database and Google Maps ..... 289**

MIT App Inventor database ..... 289

MIT App Inventor and Google Maps ..... 296

Summary ..... 303

Components List ..... 303

**Chapter 12: GPS tracking app with Google Maps ..... 305**

GPS position transmit ..... 315

GPS position receive ..... 321

Validate transmission of GPS position ..... 323

Improve GPS position signal ..... 336

Summary ..... 344

Components List ..... 345

**Chapter 13: USB OTG communication ..... 347**

App receive ..... 348

App transmit ..... 354

App receive and transmit ..... 360

Summary ..... 363

Components List ..... 363

**Chapter 14: ESP-NOW and LoRa communication ..... 365**

ESP-NOW ..... 365

LoRa communication ..... 382

Summary ..... 396

Components ..... 397

**Chapter 15: Radio frequency communication ..... 399**

Transmitting and receiving text ..... 403

Decode remote control signals ..... 410

Control pan-tilt servos with RF communication....415
Control relay with RF communication....423
Relays....428
Solid-state relay....433
Summary....435
Components List....436

**Chapter 16: Signal generation....437**

Signal generation....441
Digital to analog converter....444
Generating waves....451
ESP32 8-bit DAC....457
12-bit DAC....458
Summary....465
Components List....466

**Chapter 17: Signal generation with 555 timer IC....467**

555 timer IC....468
Monostable mode....471
Bistable mode....474
Astable mode....475
Variable duty cycle....480
50 % duty cycle....483
PWM mode....486
Function generator....488
Square wave to sine wave....493
Bipolar junction transistor as a switch....495
MP3 player and PIR sensor application....498

Summary....502
Components List....502

**Chapter 18: Measuring electricity....505**

Voltage divider....505
Analog to digital converter....507
Voltage meter....509
Voltage meter with a load....513
Resistance meter (ohmmeter)....519
Capacitance meter....522
Current meter (ammeter)....527
Current sensor....534
Current and voltage sensor....536
Solar panel and battery meter....540
Inductance meter....551
Summary....557
Components List....558

**Chapter 19: Rotary encoder control....559**

Debouncing....564
Interrupts....564
Square wave states....567
State switching....575
Incrementing a value....577
Summary....583
Components List....583

**Chapter 20: OTA and saving data to EEPROM, SPIFFS, and Excel .....585**

OTA updating .....586

Saving data .....589

Saving to EEPROM .....591

Saving to SPIFFS .....596

Downloading SPIFFS files .....602

Saving data directly to Excel .....605

Summary .....609

Components List .....610

**Chapter 21: Microcontrollers .....611**

Arduino Uno .....617

Arduino Nano .....618

Arduino Pro Micro .....619

ESP8266 development board .....621

ESP8266 analog input .....624

ESP8266 interrupts .....625

ESP8266 watchdog timer .....628

ESP32 development board .....629

ESP32 digital input .....632

ESP32 analog input .....632

ESP32 pulse width modulation .....634

ESP32 serial input .....635

Wi-Fi communication and web server .....636

ESP8266 and ESP32 interrupts .....637

ESP8266 and ESP32 and an OLED screen ..... 637
ESP32 and servo motors ..... 638
Summary ..... 639
Components List ..... 639

**Chapter 22: ESP32 microcontroller features ..... 641**

Microcontroller CPU and memory ..... 642
ESP32 cores ..... 643
Bluetooth communication ..... 653
Bluetooth Low Energy communication ..... 656
Timers ..... 672
Real-time clock and sleep mode ..... 675
Digital to analog converter ..... 678
Capacitive touch sensor ..... 679
Hall effect sensor ..... 680
Summary ..... 681
Components List ..... 682

**Appendix: Libraries ..... 683**

**Index ..... 689**

# About the Author

**Neil Cameron** is an experienced analyst and programmer with a deep interest in understanding the application of electronics. Neil wrote the book *Arduino Applied: Comprehensive Projects for Everyday Electronics* by Apress. He was a research scientist and has previously taught at the University of Edinburgh and Cornell University.

# About the Technical Reviewer

**Mike McRoberts** is the author of *Beginning Arduino* by Apress. He is winner of Pi Wars 2018 and member of Medway Makers. He is an Arduino and Raspberry Pi enthusiast.

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

# Preface

It's never been so easy and practical to access information over the Internet, develop web pages to update sensor information, build mobile apps to remotely control devices with speech recognition, or incorporate *Google Maps* in a GPS route tracking app. The combination of Wi-Fi functionality, high computing power, and low cost of the ESP8266 and ESP32 development boards extends the range of opportunities for microcontrollers. Communicating with devices and accessing information over the Internet with the ESP8266 and ESP32 microcontrollers is the focus of *Electronics Projects with the ESP8266 and ESP32.*

The first section (Chapters 1 to 6) of the book demonstrates the ease of use and the power of the ESP8266 and ESP32 microcontrollers to access and display information on the Internet. Projects include building an Internet radio, an Internet-based clock, and an international weather station and a project with the ESP32-CAM camera to upload pictures to a web page.

The book's second section (Chapters 7 to 9) covers web page design projects for updating your web page with sensor information using real-time graphics or controlling a remote device through a web page. You'll learn about AJAX (Asynchronous JavaScript and XML), which combines XML (eXtensible Markup Language) HTTP (Hypertext Transfer Protocol) requests for updating a web page with JavaScript to manage those requests, JSON (JavaScript Object Notation) to combine information transmitted by a server to the client, the two-way fast communication WebSocket protocol, MQTT brokers, and IFTTT (If This, Then That) for communication between devices on different networks. The practical projects include uploading information to the Internet and controlling

devices from anywhere in the world with the ESP8266 and ESP32 microcontrollers.

Mobile apps are now ubiquitous, making the app build projects in the book's third section (Chapters 10 to 13) very relevant. An app to control remotely located motors connected to an ESP8266 or ESP32 development board mimics robotics used in the automotive industry; a speech recognition app controls devices; and a GPS tracking app, incorporating *Google Maps*, displays the current position and route information. Each project with the ESP8266 and ESP32 microcontrollers is fully described, as no previous experience in mobile app design and build is required.

Communication between ESP8266 and ESP32 microcontrollers is described in the fourth section (Chapters 14 to 18) of the book. The built-in *ESP-NOW* communication system, LoRa (long range), and RF (Radio Frequency) communication are applied to controlling remotely located devices with the device information updated on a web page by the ESP8266 and ESP32 microcontrollers. Communication protocols are extended to signal generation with the ESP8266 and ESP32 microcontrollers transmitting alphanumeric text or signals to produce sounds, as used in electronic music. Signal generation without a microcontroller is illustrated with an electronic piano, a motor control project, and an alarm system including an MP3 player with a movement detector. The book's fourth section spans the built-in communication protocol of the ESP8266 and ESP32 microcontrollers to communication with back-to-basics electronics. A chapter on measuring electricity with an ESP8266 or ESP32 microcontroller, applied to a solar panel project, continues the electronics theme to understand the methodology behind sensors.

The ESP32 microcontroller is more powerful than the ESP8266 microcontroller and also includes Bluetooth and Bluetooth Low Energy (BLE) communication. Chapters on practical differences between the ESP8266 and ESP32 microcontrollers and on specific features of the ESP32 microcontroller form the last section (Chapters 21 and 22) of the book.

Throughout the book, all differences in libraries or instructions for the ESP8266 and ESP32 microcontrollers are described, as each project is compatible with both microcontrollers.

All sections of the book are stand-alone, so you can delve into a section of the book rather than having to start from the beginning. Several chapters build on information from earlier chapters. For example, Chapter 12 (*GPS tracking app with Google Maps)* incorporates mobile app design, Bluetooth communication, sourcing information from the Internet, and updating a web page. Some programming experience with the Arduino IDE is assumed, although all sketches are completely described and comprehensively commented. The book *Arduino Applied: Comprehensive Projects for Everyday Electronics* is recommended as an introduction to microcontrollers ranging from blinking an LED to building a robot car. Schematic diagrams were produced with *Fritzing* software (`www.fritzing.org`), with an emphasis on maximizing the clarity of component layout and minimizing overlapping connections. Authors of libraries used in the book are acknowledged in each chapter, with library details included in the Appendix. All the Arduino IDE sketches and MIT App Inventor source code for the apps are available to download at *GitHub* (`github.com/Apress/ESP8266-and-ESP32`). The Arduino programming environment and libraries are constantly being updated, so information on consequences of the updates is also available on the *GitHub* website.

# CHAPTER 1

# Internet radio

Internet radio is the continuous streaming of digital audio over the Internet. Digital audio, in MP3 format, is received by the ESP8266 or ESP32 microcontroller through a Wi-Fi connection. The ESP8266 or ESP32 microcontroller communicates with a VS1053 audio decoder by Serial Peripheral Interface (SPI), and the MP3-formatted data is decoded by an 18-bit digital to analog converter (DAC) to an audio signal that is amplified for a loudspeaker. ESP8266 and ESP32 microcontrollers have Wi-Fi functionality and sufficient processor speed for an Internet radio. Connection to the wireless local area network (WLAN) requires the Wi-Fi network SSID (Service Set Identifier) and password.


N. Cameron, *Electronics Projects with the ESP8266 and ESP32*,
https://doi.org/10.1007/978-1-4842-6336-5_1

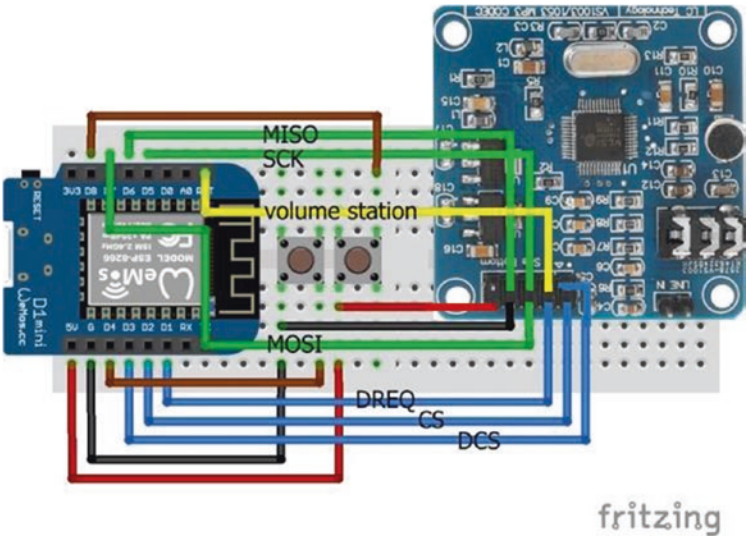



***Figure 1-1.*** *Internet radio with volume and station switches and a LOLIN (WeMos) D1 mini*

Connections for the ESP8266 development board and the VS1053 audio decoder are shown in Figure 1-1, with a detail in Figure 1-2, and listed in Table 1-1. Connections for SPI communication are indicated in green, with data connections in blue. Two switches, attached to interrupts, control the volume and Internet radio station selection. For the ESP8266 development board, the volume and station switches on pins *D4* and *D8* are connected to GND and 5V, as pins *D4* and *D8* are connected to internal pull-up and pull-down resistors, respectively. Connections for an ESP32 development board are also given in Table 1-1. When using an ESP32 development board, the volume and station switches are both connected to GND.

An amplifier and loudspeaker, or a mini-loudspeaker as used with a mobile phone, are connected to the VS1053 audio decoder by plugging into the audio jack socket of the VS1053 audio decoder.

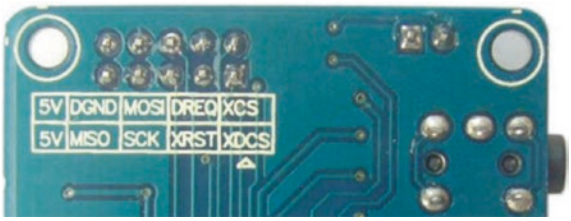



***Figure 1-2.*** *VS1053 connections*

***Table 1-1.*** *Internet radio and switches*

| Component | Connect to ESP8266 | Connect to ESP32 |
|---|---|---|
| VS1053 5V | 5V | VIN or V5 |
| VS1053 DGND | GND | GND |
| VS1053 MOSI | (MOSI) D7 | (MOSI) GPIO 23 |
| VS1053 DREQ (data request) | D1 | GPIO 4 |
| VS1053 XCS (chip select) | D2 | GPIO 0 |
| VS1053 MISO | (MISO) D6 | (MISO) GPIO 19 |
| VS1053 SCK | (SCK) D5 | (CLK) GPIO 18 |
| VS1053 XRST (reset) | RST | GPIO EN |
| VS1053 XDCS (data chip select) | D3 | GPIO 2 |
| Switch volume left | GND | GND |
| Switch volume right | D4 | GPIO 26 |
| Switch station left | 5V | GND |
| Switch station right | D8 | GPIO 27 |

The URL (Uniform Resource Locator) or web address of an Internet radio station is obtained from the website `www.radio.de`. Search for the required station, click the *play* button, and select *View Page Source*. In the displayed HTML (HyperText Markup Language) file, search for *streams*, which precedes the radio station URL. The URL is formatted as *host:port/path*. For example, *The UK 1940s Radio Station* has URL `1940sradio1.co.uk:8100/stream/1/` with *host* equal to the text before the first backslash: 1940sradio1.co.uk – and *path* equal to the remaining text: stream/1/. If the *port* is not equal to default value of 80, which is the web browsing port, then it follows the colon after *host*, such as *8100*.

The sketch for an Internet radio with an ESP8266 development board (see Listing 1-1) uses the *VS1053* library by Ed Smallenburg and James Coliz that is downloaded as a *.zip* file from `github.com/baldram/ESP_VS1053_Library`. The first section of the sketch defines the number of Internet radio stations and URLs, initializes the audio decoder, establishes a Wi-Fi connection, and defines the interrupts. The variables *newStation* and *newVolume* are defined as `volatile`, as they are accessed by both the main sketch and the interrupts. With an ESP32 development board, the station change switch pin is set *HIGH* with an internal pull-up resistor using the instruction `pinMode(statPin, INPUT_PULLUP)`, and the interrupt attached to the station switch is set to *FALLING*. The ESP8266 and ESP32 microcontrollers store compiled code in internal RAM (IRAM), rather than in the slower flash memory, by prefixing code with the *IRAM_ATTR* attribute. The interrupt ISR (Interrupt Service Routine) is defined as `IRAM_ATTR void ISR()` rather than `void ISR()`.

In the *loop* function, a connection is made to an Internet radio station website, and the VS1053 audio decoder processes data in 32-byte batches. The two interrupt service routines, *chan* and *vol*, move to the next radio station and increase the volume, respectively. The volume scale is from 0 to 100%. The *VS1053* library references the *SPI* library, and the `#include <SPI.h>` instruction is not required.

Connection to the Internet radio station server with the instruction `connect(host[station], port[station])` is followed by an HTTP (Hypertext Transfer Protocol) request. The *VS1053* library uses HTTP for communication between the client, which is the web browser, and the Internet radio station server. The client submits an HTTP request to the server for audio data, and the server sends a response to the client with the required data. The HTTP request instructions `"GET pathname HTTP/1.1"` and `"Host: hostname"` are followed by an instruction to close the connection `"Connection: close"`. Using the example of "*The UK 1940s Radio Station,*" the request instructions are

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

Note that the fourth instruction of carriage return, `\r`, and new line, `\n`, is required, which is equivalent to a `println()` instruction.

***Listing 1-1.*** Internet radio with volume and station switches and an ESP8266 board

```
#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
```

Connections for the ESP32 development board and to the VS1053 audio decoder are shown in Figures 1-3 and 1-2, respectively, and given in Table 1-1. Both switch pins are connected to internal pull-up resistors, so both interrupts are activated by a *FALLING* signal. The only changes to Listing 1-1, other than defining the decoder, station, and volume control pins, are inclusion of the *WiFi* library rather than the *ESP8266WiFi* library and the instruction `pinMode(statPin, INPUT_PULLUP)` to change the interrupt on the station switch pin from *RISING* to *FALLING*.

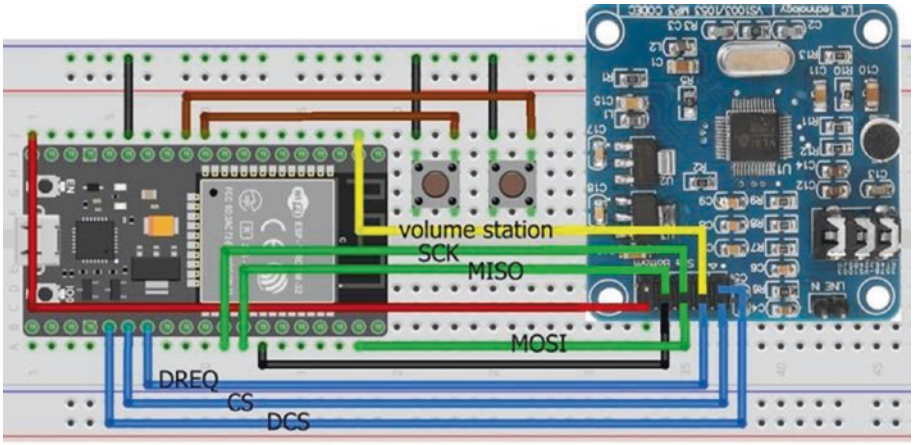



***Figure 1-3.*** *Internet radio with volume and station switches and an ESP32 board*

# Station display and selection

In Listing 1-1, station selection and volume control are activated by switches, with station and volume information displayed on the Serial Monitor. For a portable Internet radio, station and volume information is displayed on an ST7735 TFT LCD (Thin-Film Transistor Liquid Crystal Display) screen, and a station is selected or the volume is controlled with a rotary encoder (see Figures 1-4 and 1-5 with connections in Table 1-2). Note that both the rotary encoder and ST7735 TFT LCD screen are connected to 3.3V, with only the VS1053 audio decoder connected to 5V. The ESP32 microcontroller communicates with both the VS1053 audio decoder and ST7735 TFT LCD screen by SPI, so the microcontroller has the same MOSI (Main-Out Secondary-In) and SCK (Serial Clock) connections to the audio decoder and screen, but the CS (Chip Select) connections are device specific.


***Figure 1-4.*** *Internet radio screenshots*

The sketch uses the *ESP32 vs1053_ext* library by Wolle that is downloaded as a *.zip* file from `github.com/schreibfaul1/ESP32-vs1053_ext`. The *ESP32 vs1053_ext* library is for the ESP32 microcontroller, while the *VS1053* library by Ed Smallenburg and James Coliz is compatible with both the ESP8266 and ESP32 microcontrollers. The *ESP32 vs1053_ext* library provides station and track information, such as the streamed track title. The instruction to connect to an Internet radio station server is `connecttohost("host:port/stream")`, for example, `connecttohost("1940sradio1.co.uk:8100/stream/1/")`. The port number is only required when it does not equal the default value of 80. The functions *vs1053_showstation*, *vs1053_icyurl*, *vs1053_bitrate,* and *vs1053_showstreamtitle* hold the Internet radio station name and homepage URL, the bit rate, and the streamed track title. When a new track is streamed, the *vs1053_showstreamtitle* function is automatically updated. The *volume* variable has 22 levels of 0,50,60,65,70,75,80,82…90,91…100%, with volume level 10 equal to 88%, as volume level 0 has value 0%.

Listing 1-2 demonstrates the output of the *ESP32 vs1053_ext* library functions that are used in Listing 1-3 to display information about the Internet radio station and the streamed track.

***Listing 1-2.*** ESP32 vs1053_ext library functions

```
#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);
}
```

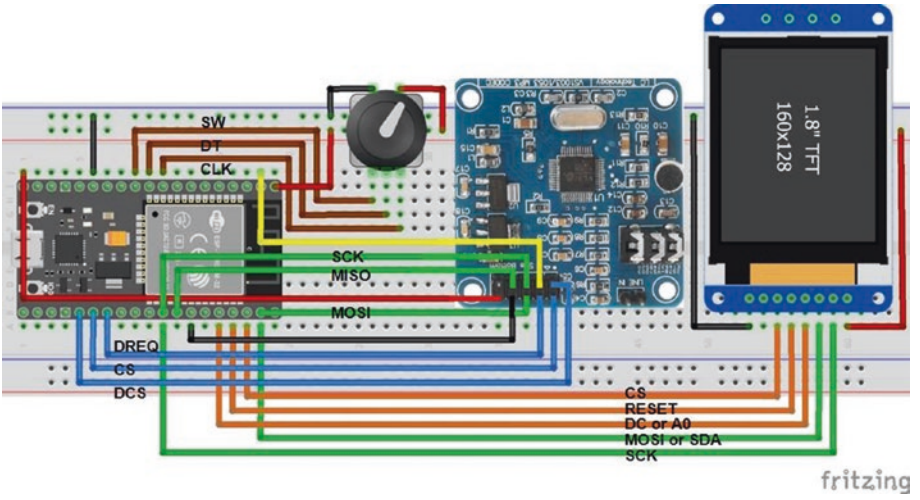



***Figure 1-5.*** *Internet radio with screen and rotary encoder and an ESP32 board*

***Table 1-2.*** *Internet radio with screen and rotary encoder and an ESP32 board*

| Component | Connect to ESP32 |
|---|---|
| VS1053 audio decoder | See Table 1-1 |
| Rotary encoder CLK | GPIO 25 |
| Rotary encoder DT | GPIO 26 |
| Rotary encoder SW | GPIO 27 |
| Rotary encoder VCC | 3V3 |
| Rotary encoder GND | GND |
| ST7735 TFT LCD GND | GND |
| ST7735 TFT LCD CS | GPIO 22 |
| ST7735 TFT LCD RESET | GPIO 1 |
| ST7735 TFT LCD DC or A0 | GPIO 3 |
| ST7735 TFT LCD SDA | GPIO 23 |
| ST7735 TFT LCD SCK | GPIO 18 |
| ST7735 TFT LCD LED | 3V3 |

The sketch for a portable Internet radio is given in Listing 1-3. Pressing the rotary encoder switch once displays the menu of available radio stations, with volume control as the first menu item. Turning the rotary encoder moves the menu of radio stations up or down the ST7735 TFT LCD screen. The mid-screen station, which is highlighted in *RED*, is selected by pressing the rotary encoder for a second time; and an HTTP request is made to the Internet radio station server for audio data. When *Volume* is selected on the menu, the current volume level is displayed

and turning the rotary encoder decreases or increases the volume level, which is selected by pressing the rotary encoder switch. The ST7735 TFT LCD screen is refreshed with the current radio station information and the updated volume level displayed, but the station menu is still positioned at the current radio station.

The sketch in Listing 1-3 consists of several functions to compartmentalize the instructions. The lengthy first section of the sketch defines the libraries, the Internet radio station URLs, pin numbers for the VS1053 audio decoder, the ST7735 TFT LCD screen, and the rotary encoder, with initial values for the station and volume level and the rotary encoder parameters. The *Adafruit ST7735* library is available in the Arduino IDE. The *ESP32 vs1053_ext* and *Adafruit ST7735* libraries reference the *SPI* and *Adafruit GFX* libraries, so the `#include <SPI.h>` and `#include <Adafruit_GFX.h>` instructions are not required. The *setup* function establishes the Wi-Fi connection, initializes the VS1053 audio decoder and the ST7735 TFT LCD screen, attaches internal pull-up resistors to the rotary encoder, and defines interrupts for the rotary encoder. The direction and number of turns of the rotary encoder are determined by the *change* interrupt, as described in Chapter 19 (Rotary encoder control).

On pressing the rotary encoder switch, the *loop* function calls the *screen* function to display the volume and station menu, the *readMenu* function to determine the selected radio station or the *readValue* function function to obtain the new volume level, and then the *radio* function. The *radio* function either connects to the selected radio station server or changes the volume on the VS1053 audio decoder. The *readMenu* and *readValue* functions determine the selected row number of the menu, which is a list of stations, and the selected volume level, when the rotary encoder is turned. The *vs1053_icyurl* function obtains a string, starting with *https://* and followed by the station URL, and extracts a substring starting two positions after the location of the first backslash. The *vs1053_showstation* and *vs1053_showstreamtitle* functions obtain