ESP32-radio

This document describes the realization of an Internet radio based on an ESP32 WiFi module.
The ESP32 is the successor of the ESP8266. For the ESP8266 an Internet radio was build and now the software is ported to the ESP32. The Internet radio described here uses the ESP32 as well as a VS1053 module to decode the MP3 stream. A 1.8 TFT color display to give some information about the radio station that's playing. This display has also a SD card slot. Mp3 tracks on an SD card can be played by the ESP32-radio. Other types of display are also supported, including NEXTION.

Features:

- Can connect to thousands of Internet radio stations that broadcast MP3 audio streams.
- Can connect to a standalone mp3-file on a server.
- mp3 playlists supported.
- Can play MP3 files from SD card or USB flash drive.
- Uses a minimal number of components.
- Has a preset list of a maximum of 100 favorite radio stations as preferences in flash.
- Can be controlled by a tablet or other device through the built-in webserver.
- Can be controlled by MQTT commands. Status info is published by using MQTT.
- Can be controlled by commands on the serial input.
- Can be controlled by IR.
- Can be controlled by rotary switch encoder.
- Up to 14 input ports can be configured as control buttons like volume, skip to the next/previous/first/favorite preset station.
- I/O pins for TFT, VS1053, IR and rotary encoder can be configured in preferences.
- The strongest available WiFi network is automatically selected. Passwords are kept as preferences in flash. Heavily commented source code, easy to add extra functionality.
- Debug information through serial output.
- 2nd processor is used for smooth playback.
- Update of software over WiFi (OTA) through Arduino IDE or remote server.
- Parameters like WiFi information and presets can be edited in the web interface.
- Plays iHeartRadio streams.
- Preset, volume, bass and treble settings are saved in preferences.
- Displays time of day on TFT.
- Optional displays remaining battery charge.
- Bit-rates up to 320 kbps.
- Control of TFT back-light.
- PCB available.
- 3D printable box available.

Software:

The software for the radio is supplied as an Arduino sketch that can be compiled for the ESP32 using the Arduino IDE version 1.8.5. No Arduino is required in this project.
The following libraries are used:

- WiFi – for establishing the communication with WiFI, part of ESP32 core library
- SPI – for communication with VS1053 and TFT display, part of ESP32 core library
- WiFiMulti - to select the strongest WiFi network
- Adafruit_ST7735– driver for the TFT screen (if configurated).
  This driver works also for the red ST7735 display.
- ArduinoOTA for software update over WiFi.
- PubSubClient to handle MQTT messages, see:
  https://github.com/knolleary/pubsubclient
- SD and FS for reading from SD card.
- The map with the ESP32-radio sketch must also contain the supplied headerfiles “index_html.h”, “favicon_ico.h”, “radio_css.h”, “config_html.h”, "defaultprefs.h", "mp3play_html.h" and “about_html.h”. These files are included for the webinterface in PROGMEM.
  Furthermore the files “Dummytft.h”, “LCD1602”, “SSD1306.h”, “bluetft.h”, “ILI9341.h”, “NEXTION.h” must be included in this map for the different LCD/OLED displays.

Preferences

Preferences for ESP32-Radio are kept in Flash memory (NVS).
You may change the contents of NVS with the “Config”-button in the web interface. An example of the contents is:

```
bat0   = 2178   # ADC for 0% battery capacity left
bat100 = 3256   # ACD for 100% battery capacity
#
clk_dst = 1
clk_offset = -1
clk_server = pool.ntp.org
#
fs = USB
#
gpio_00 = uppreset = 1
gpio_12 = upvolume = 2
gpio_13 = downvolume = 2
#
ir_40BF = upvolume = 2
ir_C03F = downvolume = 2
#
mqttbroker = none
mqttpasswd = none
mqttport = 1883
mqttuser = carol
#
pin_enc_clk = 25     # GPIO rotary encoder CLK
pin_enc_dt = 26      # GPIO rotary encoder DT
pin_enc_sw = 27      # GPIO rotary encoder SW
pin_ir = 35          # GPIO IR receiver VS1838B
pin_sd_cs = 21       # GPIO SD card CS
pin_tft_cs = 15      # GPIO TFT CS
pin_tft_dc = 2       # GPIO TFT DC
pin_vs_cs = 5        # GPIO VS1053 CS
pin_vs_dcs = 16      # GPIO VS1053 DCS
pin_vs_dreq = 4      # GPIO VS1053 DREQ
#
preset = 4
preset_000 = 109.206.96.34:8100                          #  0 - NAXI LOVE RADIO, Belgrade, Serbia
preset_001 = airspectrum.cdnstream1.com:8114/1648_128 #  1 - Easy Hits Florida 128k
preset_002 = airspectrum.cdnstream1.com:8142/1303_128 #  2 - CLASSIC ROCK MIAMI 256k
preset_003 = airspectrum.cdnstream1.com:8000/1261_192 #  3 - Magic Oldies Florida
preset_004 = airspectrum.cdnstream1.com:8008/1604_128 #  4 - Magic 60s Florida 60s Classic Rock
preset_005 = 198.58.98.83:8258/stream                    #  5 - Classic Rock Florida HD
preset_006 = icecast.omroep.nl:80/radio1-bb-mp3          #  6 - Radio 1, NL
preset_007 = icecast.omroep.nl:80/radio2-bb-mp3          #  7 - Radio 2, NL
preset_008 = skonto.ls.lv:8002/mp3                       #  8 - Skonto 128k
preset_009 = 94.23.66.155:8106                           #  9 - *ILR CHILL and GROOVE
preset_010 = ihr/IHR_IEDM                                # 10 - iHeartRadio IHR_IEDM
preset_011 = ihr/IHR_TRAN                                # 11 - iHeartRadio IHR_TRAN
#
toneha = 0
tonehf = 0
tonela = 0
tonelf = 0
#
volume = 72
#
wifi_00 = ADSL-11_plus/xxxxxx
wifi_01 = NETGEAR-11/xxxxxx
wifi_02 = ADSL-11/xxxxxx
```

Not that the size of the NVS is limited to 20 kB.
Lines starting with “#” are comment lines and are ignored. Comments per line (after “#”) will also be ignored, except for the “preset_” lines. The comments on the “preset_” lines are used to identify the presets in the web interface.
The maximum line length is 150 characters.
Note that the preset numbers ranges from 000 to 199. The range can be extended by the definition of MAXPRESETS.
Note that there is also a limit on the NVS space. If the highest numbered station is reached, the next station will be 00 again. URLs with mp3 files or mp3 playlists (.m3u) are allowed.
Presets starting with "ihr/" are iHeartRadio stations.

Prteferences like "pin_ir", " pin_enc_clk", " pin_enc_dt", " pin_enc_sw", " pin_tft_cs", " pin_tft_dc", " pin_sd_cs", " pin_vs_cs", "pin_vs_dcs" and " pin_vs_dreq" specify the GPIO pins that are wired to the various devices. For a

FEATERBOARD you have to change some settings, especially for TFT and SD card. The pins for SPI default to SCK=18, MISO=19 and MOSI=23, but they may be configured by “pin_spi_sck”, “pin_spi_miso” and “pin_spi_mosi” respectively.
For an I2C display (OLED, LCD1602, LCD2004) there must be definitions for " pin_tft_scl" and " pin_tft_sda". Remove the lines if a device is not connected.

“pin_shutdown” is the pin that will be set if the radio is not playing or the volume is set to zero. This output-pin can be used to shut down the amplifier.
“pin_shutdownx” has the same function. The output signal is inversed: the pin will be LOW to shut down the amplifier.

"pin_tft_bl" is the pin that controls the back light of a TFT display. The BL is disbled after some time if there is no activity. This helps to reduce power when running on batteries.
"pin_tft_blx" has the same function. The output signal is inversed (LOW to enable the BL pin).

The "gpio_" lines specify input pins and the command that is executed if the input pin goes from HIGH to LOW.

The "touch_" lines specify the GPIO input pins for this signal and the command that is executed if the input pin is activated.

The "ir_XXXX" lines specify IR-codes and the command that is executed if the IR-code is received. The code part is a hexadecimal number in upper-case characters.

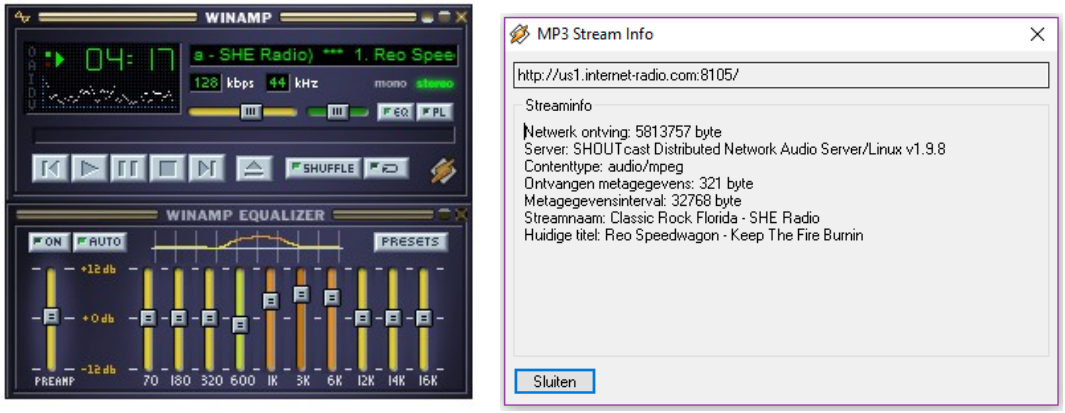
The "clk_" lines are used for the display of the current time on the TFT. Please change the values for your timezone. The values in the example are valid for the Netherlands.

The “fs” line specifies the file system for MP3 player mode. The file system can be USB or SD for USB drive and SD card respectively. Note that the sketch must be compiled for USB or SD card support. See the defines for SDCARD and/or CH376.

The “bat0” and “bat100” lines are for displaying the remaining battery capacity on the display. See the details at the paragraph with optional features.

The preferences can be edited in the web interface. Changes will in some cases be effective after restart of the Esp-radio. If the list of preferences is empty (first start), you may use the default button. The list will show some default values which can be edited.

Using Winamp to find out the correct preset line for a station.



Press Alt-3 in the main window (left picture). You will see info for the playing station (right picture).
The top line (with “http”) will contain the information for the preset, in this example:

“us1.internet-radio.com:8105”. The complete line for this station in the preferences would be:

```
preset_05 = us1.internet-radio.com:8105            #  5 - Classic Rock Florida - SHE Radio
```

Optional features:

Digital control through input pins:

Normally the radio is controlled by the web interface. However, free digital inputs (GPIO) may be connected to buttons to control the radio. Their function can be programmed using the web interface.

You can assign commands to the digital inputs by adding lines in the configuration (Webinterface, "Config" page). Examples:

```
gpio_00 = uppreset = 1
gpio_12 = upvolume = 2
gpio_13 = downvolume = 2
gpio_14 = stop
gpio_17 = resume
gpio_21 = station = icecast.omroep.nl:80/radio1-bb-mp3
```

In this example the ESP32-Radio will execute the command "uppreset=1" if GPIO0 will go from HIGH to LOW. The commands are equal to the commands that are handled by the serial input or by the MQTT interface.

The same format can be used for the touch-inputs:

```
touch_04 = uppreset = 1
touch_13 = upvolume = 2
```

In this example the ESP32-Radio will execute the command "uppreset=1" if TOUCH0 (GPIO04) will be activated. The commands are equal to the commands that are handled by the serial input or by the MQTT interface.

IR Interface.

The radio can be controlled by an IR remote control like this:

To use this interface, the "out" pin of a VS1838B receiver must be connected to a GPIO pin of the ESP32:

Add the assigned GPIO pin to the preferences through the config page in the web interface. Example:

```
pin_ir = 35                    # GPIO Pin number for IR receiver VS1838B
```

VCC is connected to 3.3 Volt. A 220 µF capacitor should be connected between VCC and GND.
The software will read the raw code of the IR transmitter, making it possible to use virtually any remote control to be used. I tested it with the 21 button remote as well as with an LG TV remote.
To assign functions to the buttons, watch the debug log output while pressing a button. For example, press the +volume button. You will see something like:

```
D: IR code 807F received, but not found in preferences!
```

Now add the command:

```
ir_807F = upvolume = 2
```

to the preferences in the config page of the web interface. Likewise you can assign functions to all buttons, for example:

```
ir_8A31 = uppreset = 1
ir_719A = station = us1.internet-radio.com:8105
ir_1F6B = mute
```

Display remaining battery capacity:

The remaining battery capacity is computed by measuring the battery voltage on ping GPIO36 (ADC0).
The battery voltage must be supplied to ADC0 through a resistor voltage divider. The maximum voltage fort his input is 1.0 volt. I used 100 kΩ and 22 kΩ fort his purpose.
Furthermore you need to specify ADC value for both the full and the empty voltages. These values must be in the preferences like:

```
bat0   = 2178   # ADC for 0% battery capacity left
bat100 = 2690   # ACD for 100% battery capacity
```

To calibrate the settings you may use the “TEST” command in the serial monitor. This will display the current reading of the ADC0 pin. Do this for a fully charge battery and for an (almost) empty one.
Note that it will take about 6 seconds before the reading is stable. This is a result of a filter in the measurement of the ADC.

The remaining capacity is displayed as a green/red bar near the top of the display.

Hardware:

The radio is built with the following widely available hardware:

- An ESP-32 module. This is basically an ESP32 on a small print. I used a DOIT ESP32 Development Board. See figure 1 below. The ESP32 is running on 160 MHz. On Aliexpress: this.
- A VS1053 module. See figure 2.
- A 1.8 inch color TFT display 160x128. Optional, see figure 3 and 4. On Aliexpress: this or this.
- A 1.44 inch color TFT display 128x128. Optional, see figure 9. On Aliexpress: this.
- An OLED 128x64 display. Optional, see figure 7. On Aliexpress: this.
- An 1602 LCD display with I2C backpack. Optional, see figure 8.
- An 2004 LCD display with I2C backpack. Optional, see figure 13.
- A NEXTION display. Optional, see figure 10. Tested with a NX3224T024_011.
- Two small speakers.
- A Class D stereo amplifier to drive the speakers. Best quality if powered by a separate power source.
- A rotary encoder switch. Optional, see figure 4.
- A IR receiver. Optional, see figure 6.
- Two audio transformers. Optional, see figure 11.
- Optional CH376 drive for USB stick, see figure 12.

Fig. 1

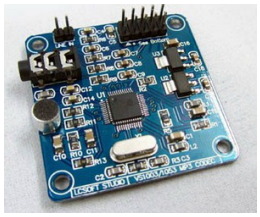


Fig. 2

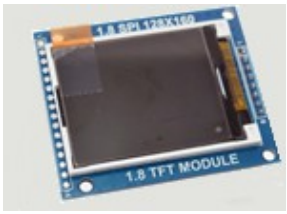


Fig. 3

Fig. 4

Fig. 5

Fig. 6

Fig. 7

F

fig. 8

Fig. 9

Fig. 10

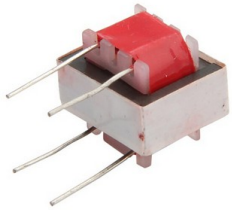
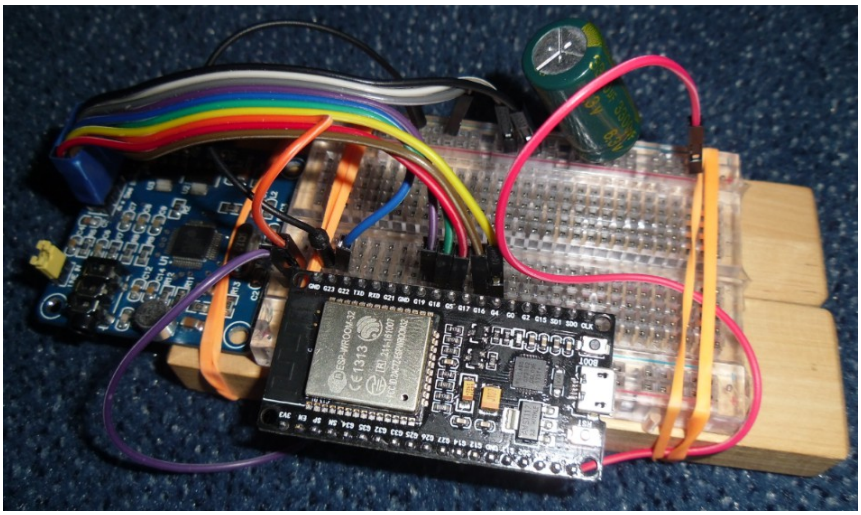


Fig. 11

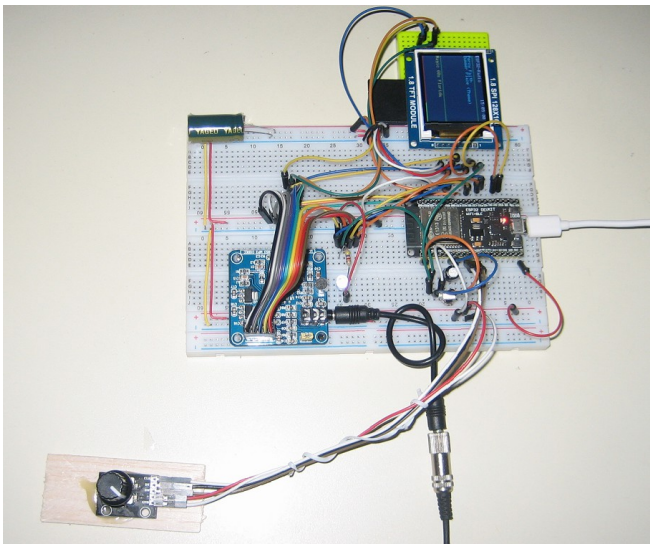
Fig. 12

Fig. 13

Here is a picture of the radio in a test configuration. Only the VS1053 is connected.



The module is not very breadboard friendly. The (hanging over) +5 Volt of the ESP32 is wired to the 5Volt rail of the breadboard. The big capacitor (3300 µF) is added to allow powering the module from USB. Without it, the USB has insufficient power to drive the ESP32.

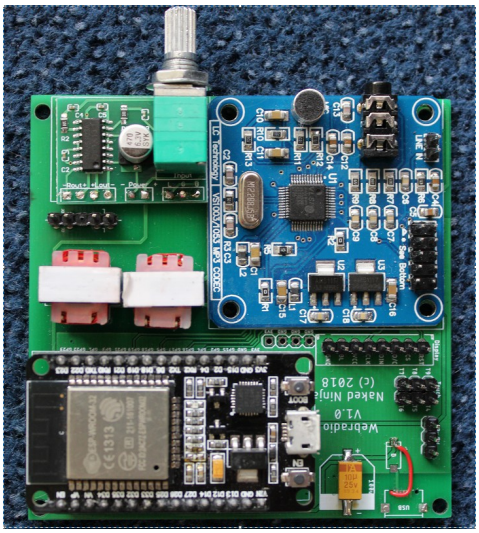


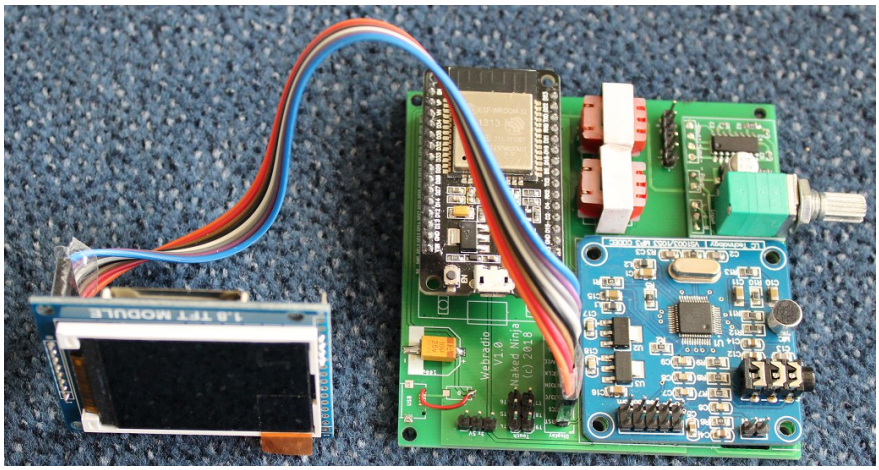
Another prototype. This time with SD-card, TFT, IR and rotary switch.

The radio may be powered by a 5 V adapter. The radio will function on single LiPo cell as well, so I added a small charge circuit powered by the 5 V input. The amplifier uses a separate LiPo cell to minimize noise caused by the ESP32. The TFT and VS1053 work on 3.8 to 5 Volt.

PCB.

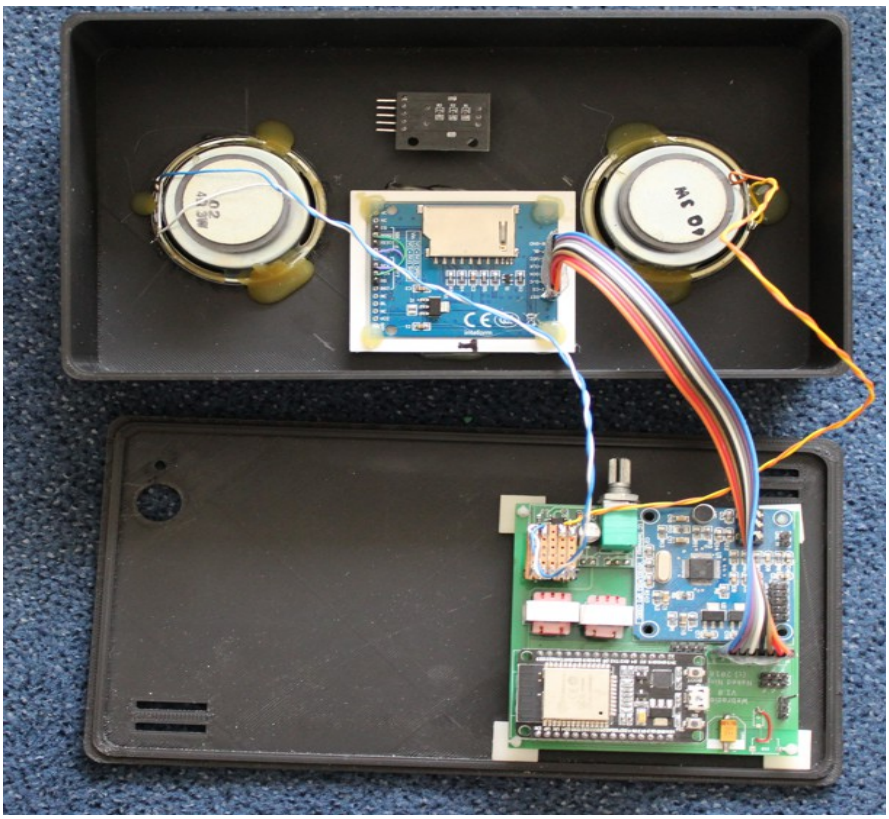
Harm Verbeek, see https://github.com/Edzelf/ESP32-Radio/issues/198, designed a PCB for this radio. I have built the radio on this PCB:





There is another PCB version available, see https://github.com/Edzelf/ESP32-Radio/issues/165.

Hans Vink designed a #D printable case for this radio. Mounted in tis case is looks like this:



See https://www.thingiverse.com/thing:3346460

Note that power supply and rotary switched are not connected here.

And playing:

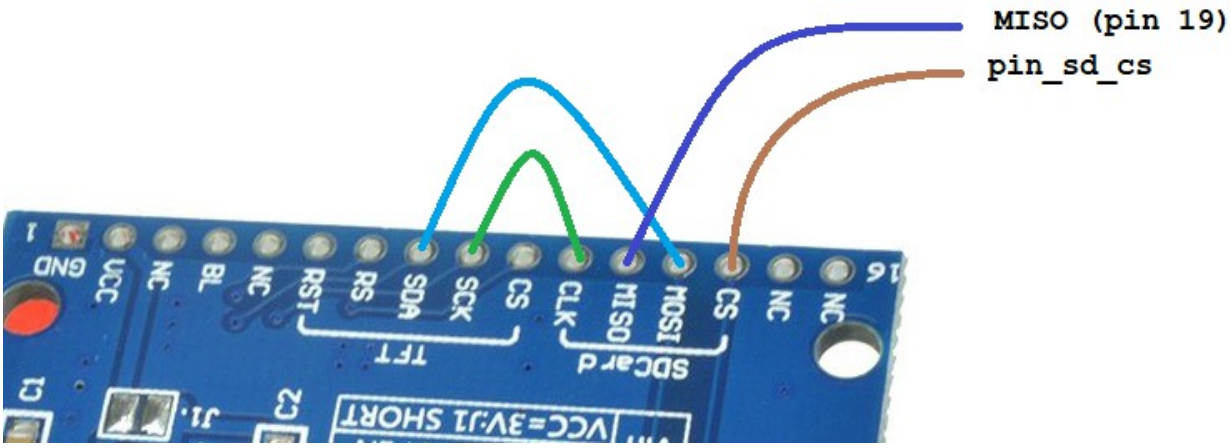
Wiring:

The logic wiring in the table below. The analog amplifier and the speakers are not included.

Note that the GPIO pins are just an example. The assignment of GPIO pins is defined in the preferences that can be edited in the "config" page of the web interface.

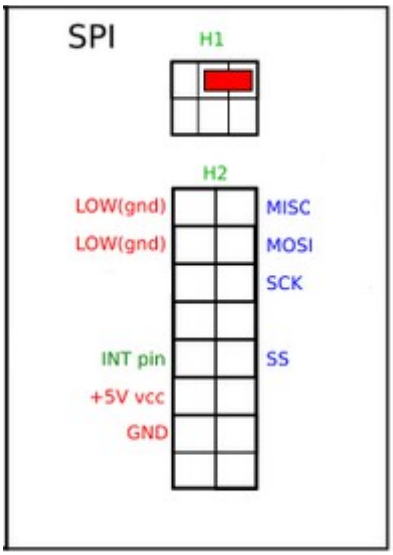
ESP32dev	Signal	Wired to LCD	Wired to VS1053	SDCARD	Wired to the rest
GPIO16		-	pin 1 XDCS	-	-
GPIO5		-	pin 2 XCS	-	-
GPIO4		-	pin 4 DREQ	-	-
GPIO2		pin 3 D/C or A0	-	-	-
GPIO22		-	-	CS	-
GPIO21	RXD2	-	-	-	TX of NEXTION (if in use)
GPIO17	TXD2	-	-	-	RX of NEXTION (if in use)
GPIO18	SCK	pin 5 CLK or SCK	pin 5 SCK	CLK	-
GPIO19	MISO	-	pin 7 MISO	MISO	-
GPIO23	MOSI	pin 4 DIN or SDA	pin 6 MOSI	MOSI	-
GPIO15		pin 2 CS	-	-	-
GPI03	RXD0	-	-	-	Reserved serial input
GPIO1	TXD0	-	-	-	Reserved serial output
GPIO34	-	-	-	-	Optional pull-up resistor
GPIO35	-	-	-	-	Infrared receiver VS1838B
GPIO25	-	-	-	-	Rotary encoder CLK
GPIO26	-	-	-	-	Rotary encoder DT
GPIO27	-	-	-	-	Rotary encoder SW
GND	-	pin 8 GND	pin 8 GND		Power supply GND
VCC 5 V	-	pin 7 BL	-		Power supply
VCC 5 V	-	pin 6 VCC	pin 9 5V		Power supply
EN	-	pin 1 RST	pin 3 XRST		-

The wiring of the 1.8 inch TFT module is on the right side of the board.
If you use the SD-card on this module, you can wire MOSI and CLK (SPI signals) on the connector on the left side:



In this case, only 2 extra wires are needed for using the SD feature: The MISO signal (pin GPIO19 of the ESP32) and the Chip Select. CS in this example is pin GPIO21, but it can be defined by “pin_sd_cs” in the configuration page of the web interface.

If you want to connect an USB drive, you have to wire the CH376 module as follows:



MISO, MOSI and SCK: wire to the SPI bus.
SS is the select pin, wire to the pin defined by “ch376_cs_pin”.
INT is the interrupt pin, wire to the pin defined by “ch376_int_pin”.

In the sketch you have to define “CH376” near line 166, like:

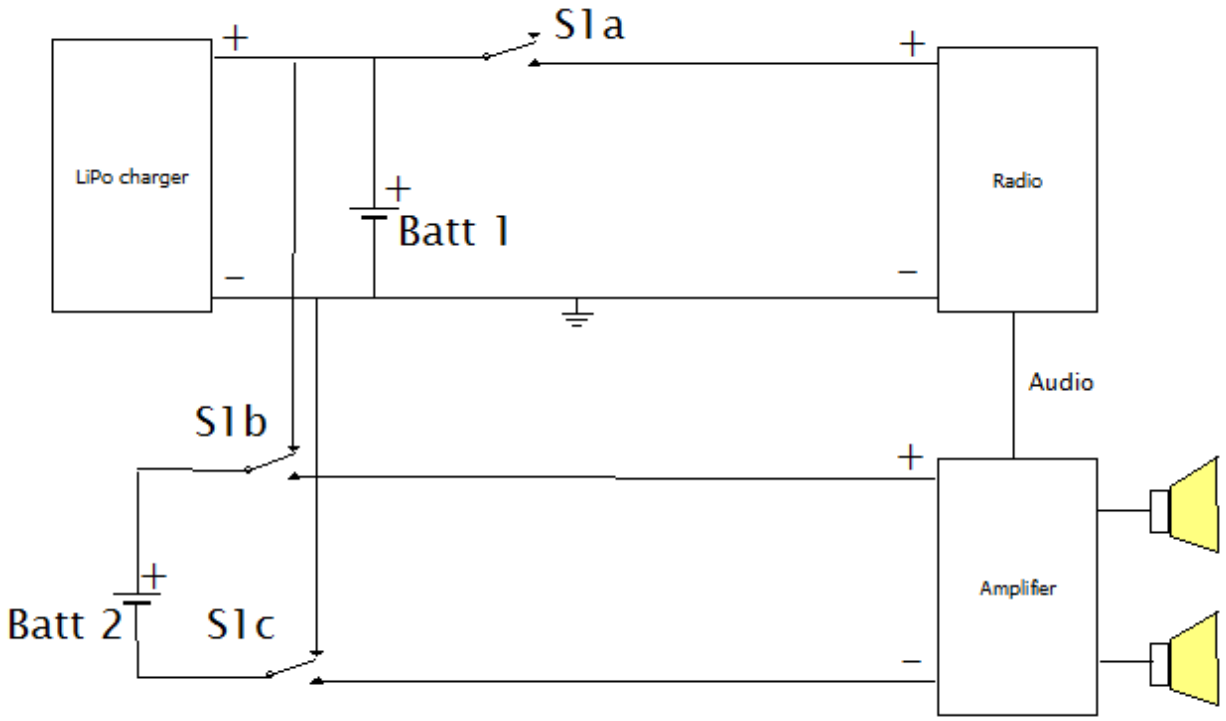
```
#define CH376  // For CXH376 support (reading files from USB stick)
```

Amplifier and power circuit.

The amplifier is a class D stereo amplifier. If the power is shared with the power supply of the radio, you will hear much noise. So I used a separate LiPo battery (Batt 2) for the amplifier.

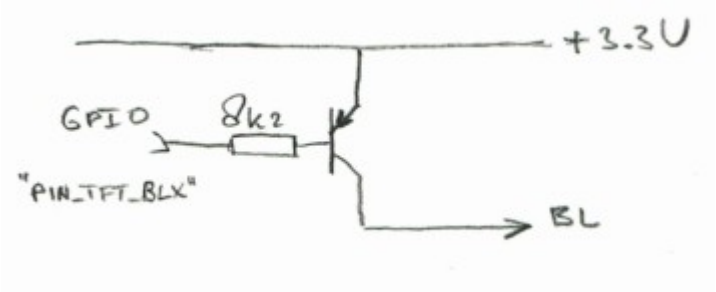
During operation only Batt 1 will be charged. If the radio is switched off, both batteries will be recharged by the LiPo charger. S1a, S1b and S1c is a triple On-On switch.

Note that there may be high currents in the “off”-position if one of the batteries fully discharged. Use protected batteries only!



A better approach is to use 2 small audio transformers (Fout: Bron van verwijzing niet gevonden) to isolate the output of the VS1053 from the power circuit. This is used on the PCB made by Harm Verbeek.

Backlight control circuit.



Web interface.

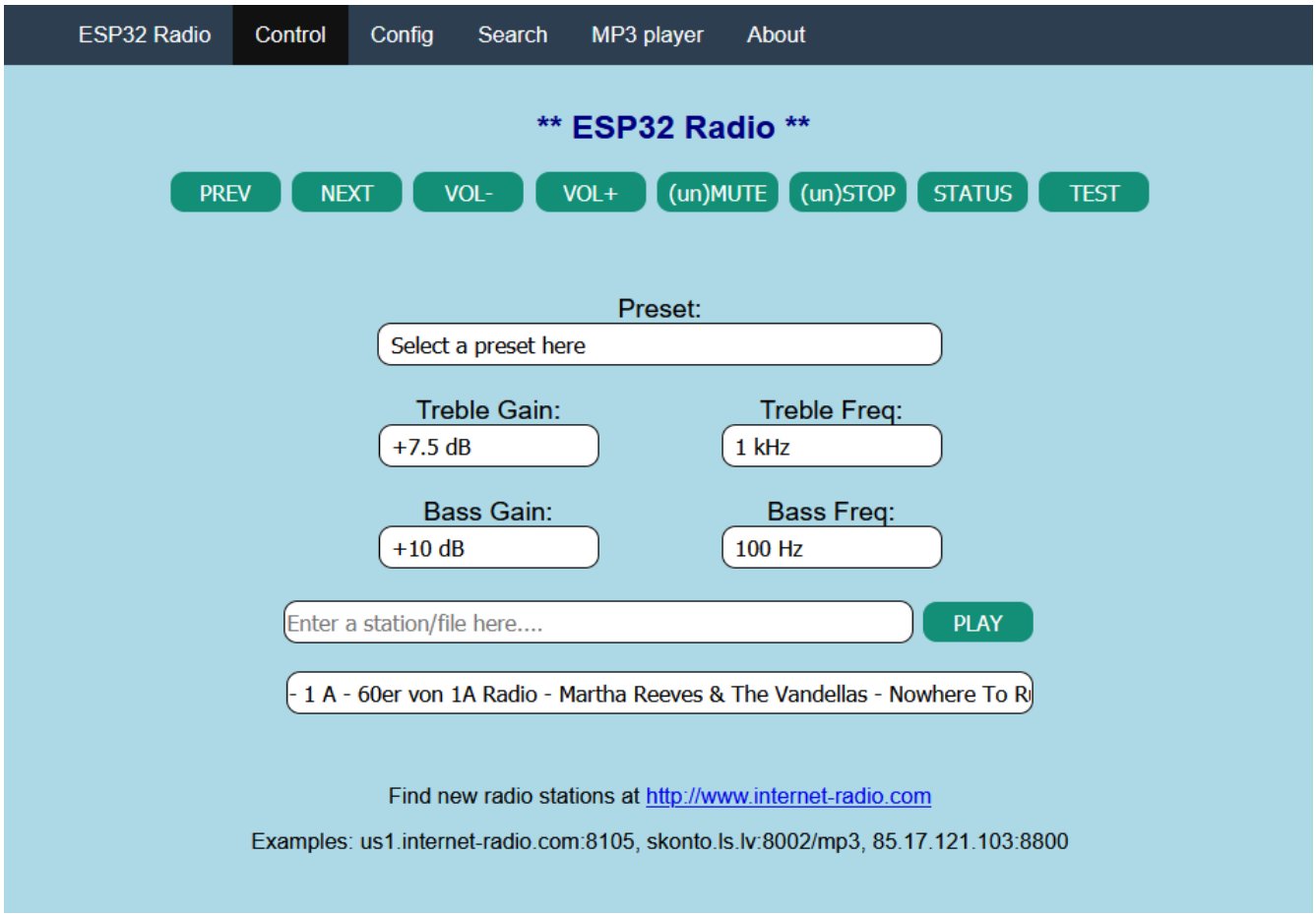
The web interface is simple and can be adapted to your needs. The basic idea is to have a html page with embedded javascript that displays an interface to the radio. Command to the radio can be sent to the http server on the ESP32. The IP address of the webserver will be displayed on the TFT during startup. The webpages are defined in PROGMEM.

Capabilities of the webserver:

Let's assume that the IP of the ESP32-radio is 192.168.2.12. From your browser you can show a simple root page by entering the following URL: http://192.168.2.12. This will display the index_html.h page from the PROGMEM as well as favicon_ico.h.

If your computer is configured for mDNS, you can also use http://ESP32Radio.local in your browser.

The following simple web interface will be displayed:



Clicking on one of the available buttons will control the ESP32-radio. The reply of the webserver will be visible in the status box below the buttons. A click will be translated into a command to the ESP32-radio in the form:

http://192.168.2.12/?<parameter>=<value>

For example: http://192.168.2.12/?upvolume=2

The "STATUS" and "TEST" buttons give some info about the playing stream/file. The "Search" menu option can be used to discover stations. The list of stations is maintained by Community Radio Browser. See https://www.radio-browser.info.

Not all functions are available as buttons in the web interface shown above. Commands may also come from MQTT or serial input. Not all commands are meaningful on MQTT or serial input. Working commands are:

```
preset     = 12                   Select start preset to connect to
preset_00  = <mp3 stream>         Specify station for a preset 00-max *)
volume     = 95                   Percentage between 0 and 100
upvolume   = 2                    Add percentage to current volume
downvolume = 2                    Subtract percentage from current volume
toneha     = <0..15>              Setting treble gain
tonehf     = <0..15>              Setting treble frequency
tonela     = <0..15>              Setting bass gain
tonelf     = <0..15>              Setting treble frequency
station    = <mp3 stream>         Select new station (will not be saved)
station    = <URL>.mp3            Play standalone .mp3 file (not saved)
station    = <URL>.m3u            Select playlist (will not be saved)
stop                              Stop playing
resume                            Resume playing
mute                              Mute/unmute the music (toggle)
wifi_00    = mySSID/mypassword    Set WiFi SSID and password *)
mqttbroker = mybroker.com         Set MQTT broker to use *)
mqttprefix = XP93g                Set MQTT broker to use
mqttport   = 1883                 Set MQTT port to use, default 1883 *)
mqttuser   = myuser               Set MQTT user for authentication *)
mqttpasswd = mypassword           Set MQTT password for authentication *)
clk_server = pool.ntp.org         Time server to be used *)
clk_offset = <-11..+14>           Offset with respect to UTC in hours *)
clk_dst    = <1..2>               Offset during dst in hours *)
mp3track   = <nodeID>             Play track from SD card, nodeID 0 = random
settings                          Returns setting like presets and tone
status                            Show current URL to play
test                              For test purposes
debug      = 0 or 1               Switch debugging on or off
reset                             Restart the ESP32
bat0       = 2318                 ADC value for an empty battery
bat100     = 2916                 ADC value for a fully charged battery
fs         = USB or SD            Select local filesystem for MP# player mode.
```

Commands marked with "*)" are sensible during power-up only.

Station may also be of the form “skonto.ls.lv:8002/mp3”. The default port is 80.
Station may also point to an mp3 playlist. Example: “www.rockantenne.de/webradio/rockantenne.m3u”.
Station may be an .mp3-file on a remote server. Example: "www.stephaniequinn.com/Music/Rondeau.mp3".
Station may also point to a local .mp3-file on SD card. Example: “localhost/friendly.mp3”.
It is allowed to have multiple (max 100) “preset_” lines. The number after the “_” will be used as the preset number. The comment part (after the “#”) will be shown in the web interface.

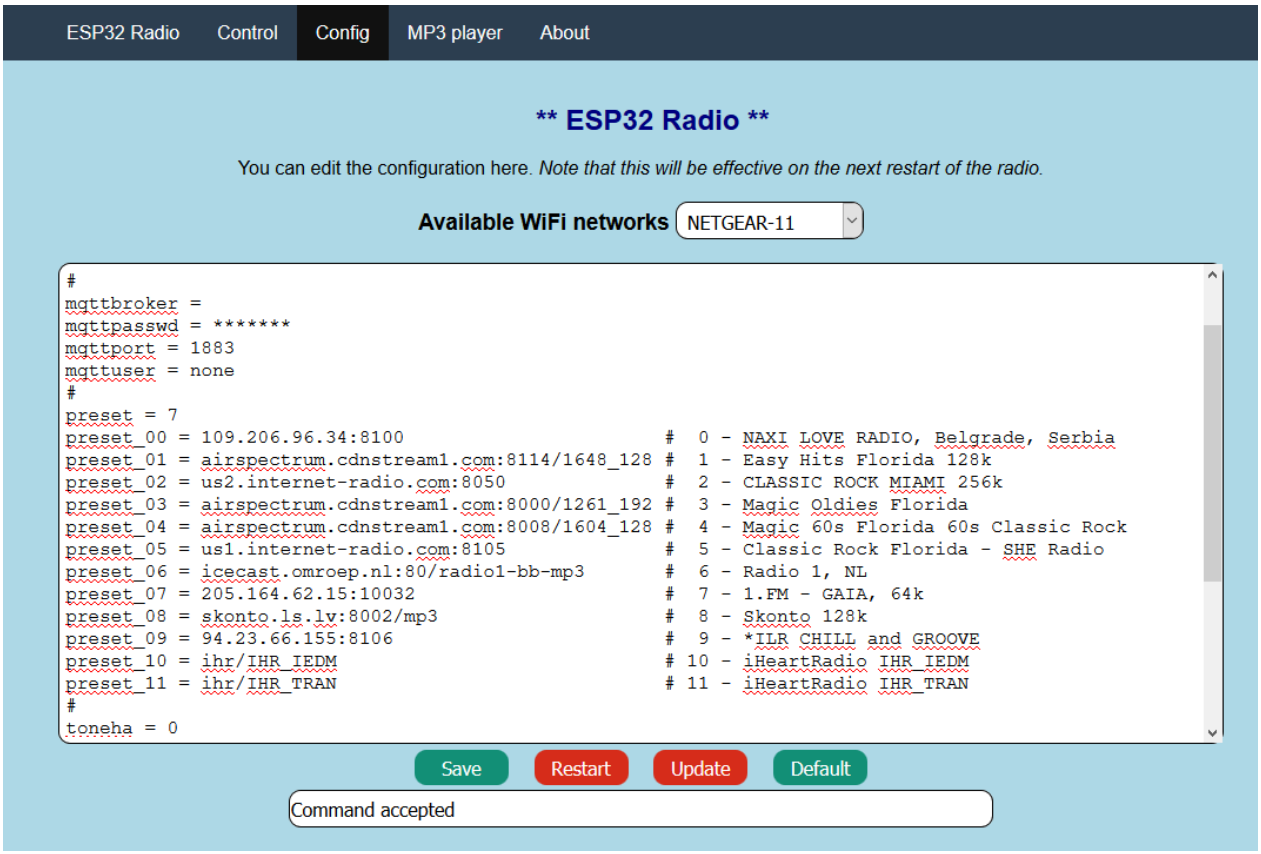
It is also allowed to have multiple “wifi_” lines. The strongest Wifi access point will be used.

Configuration

The "Config" button will bring up a second screen. Here you can edit the preferences. The available Wifi networks are listed as well. The config screen will be shown automatically if the ESP32 cannot connect to one of the WiFi stations specified in the preferences. In that case the ESP32 will act as an accesspoint with the name “ESP32Radio”. You have to connect to this AP with password “ESP32Radio”. Then the ESP32-radio can be reached at http:// 192.168.4.1.
The configuration screen will be displayed and you have the opportunity to edit the preferences. A "default" button is available to fill the editbox with some example data. For a quick start, just fill in you WiFi network name and password and click "Save". Restart the radio afterwards.
Note that passwords are hidden in this screen to give some protection against abuses. You may however change them here. If unchanged, the original values are preserved.

After changing the contents of this page, it must be saved to the preferences by clicking the “Save” button. Changes will have effect on the next restart of the ESP-radio, so click the “Restart” button.

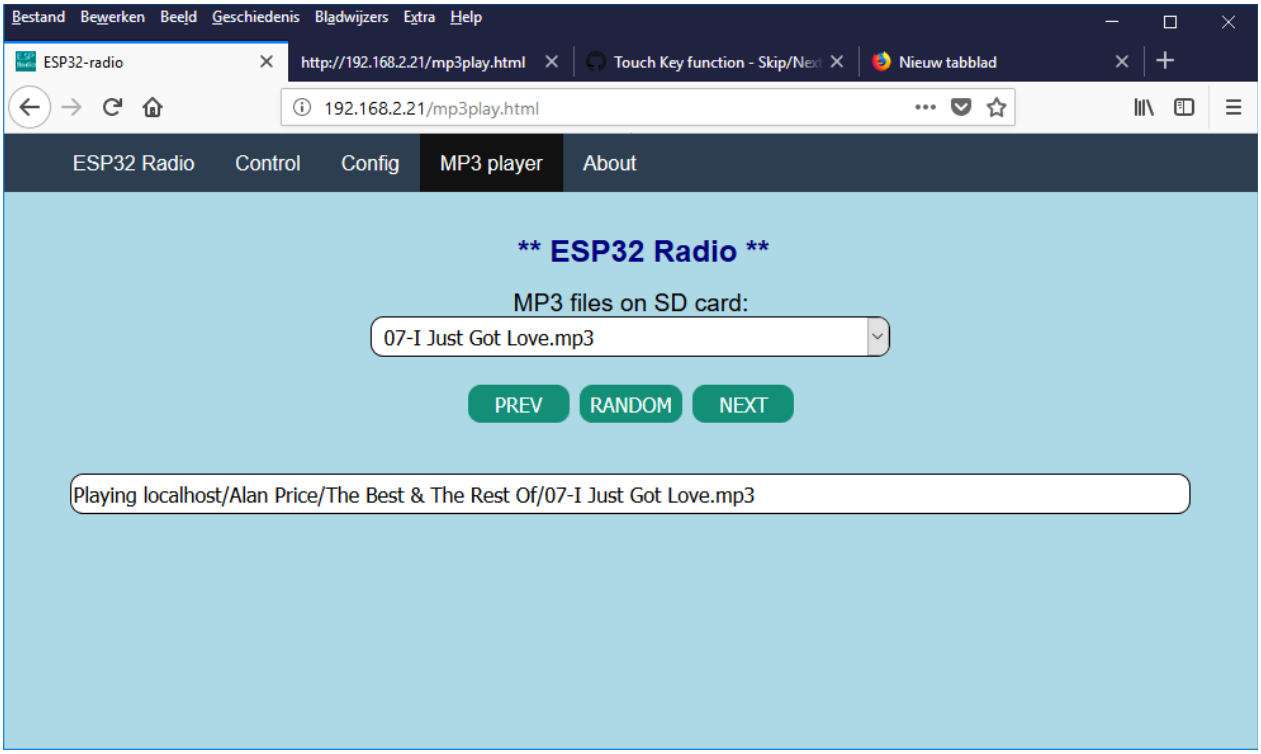
The “Update”-button can be used to get the latest version of the software by downloading it from http:smallenburg.nl/Arduino/ Esp32_radio.ino.bin.



It may take some time to fill this screen.

MP3 player

ESP32-Radio can also be used as an mp3 player if an SD card interface is available. The screen will look like this:



When this page is called for the first time, a search for all tracks on the SD card will be executed. This may take some time. All found tracks will be available in the drop-down list on the screen. You may pick a track and it will play. When the track has finished, the next track in the drop-down list will be played automatically.

The “PREV” and “NEXT” buttons will select the previous or next track to be played respectively. You may also click the "RANDOM" button and a random track will be played. If the track has finished a new random song will be selected. In “RANDOM” mode, the “PREV” and “NEXT” buttons will also select a random track.

It may take some time to fill this page because the directories on the SD card are searched for MP3 tracks.

Playlist.

You may use playlists for the MP3 player as well. Playlists have the “.m3u” extension and4 can reside on every sub-directory of the SD card. An example of a playlist (“playlist1.m3u” in the rootdirectory) with tracks pointing to subdirectories (MAP0, MAP4) may look like:

```
#EXTM3U
#EXTINF:124,01bachfugue.mp3
MAP0/bachfugue.mp3
#EXTINF:177,02gtr-nylon22.mp3
MAP0/gtr-nylon22.mp3
#EXTINF:207,03harpsi-cs.mp3
MAP0/harpsi-cs.mp3
#EXTINF:100,01viola2.mp3
MAP4/viola2.mp3
#EXTINF:101,02violin.mp3
MAP4/violin.mp3
#EXTINF:102.03vln-lin-cs.mp3
MAP4/vln-lin-cs.mp3
```

A preset to this playlist could be:

```
preset_008 = /localhost/playlist1.m3u
```

You may also start this playlist by entering “`localhost/playlist1.m3u`” in the “Play” box of the web interface.

Rotary encoder interface.

The rotary encoder switch can control some essential functions of the ESP32-radio. The "GND"- and "+"-pins must be connected to ground and 3.3 Volt pins of the ESP32 DEV module.

The default function is volume control. Turning the knob will result in lower or higher volume. Pressing the knob will mute/unmute the signal. The text “Mute” or “Unmute” will be shown on the TFT during 4 seconds.

A double click selects the preset-mode. Rotation of the switch will select one of the preset stations. The preset will be shown on the TFT. Once a preset is selected, you can activate this preset by a single click.
Without rotation, the next preset is selected.

A triple click will select the mp3-player (SD card required). Rotation of the switch will select one of the tracks on the SD card. The track will be displayed on the TFT. Once a track is selected, you can activate it by a single click.
Without rotation, the next track is selected.
After the triple click the player stops, as reading filenames will overload the SD card I/O.

A long click (longer than 1 second) will start playing random tracks from SD card.

After an inactivity of 4 seconds the rotary encoder will return to its default function (VOLUME).

Supported displays.

The preferred display is a 1.8 inch TFT with 160x128 pixels. Other displays may also be used:

1. No display. Edit the sketch (around line 123), so that “DUMMYTFT” is defined and all other display types are commented out.
2. The preferred display 160x128 pixels. Define “BLUETFT” and comment out the other types.
3. An ILI9341 display, default 320x240 pixels. Define “ILI9341” and comment out the other types.
4. A 128x128 1.44 inch display. Edit bluetft.h (around line 58), use “INITR_144GREENTAB” and set dsp_getwidth() to 128.
5. A 128x64 OLED. Define “”OLED” and comment out the other types.
6. AN LCD 1602 display with I2C backpack. Define “LCD1602I2C” and comment out the other types. Note that this display is limited as it has only 32 characters.
7. AN LCD 2004 display with I2C backpack. Define “LCD2004I2C” and comment out the other types. Note that this display is limited as it has only 80 characters.
8. A NEXTON model NX#@@$T024_11 display connected to GPIO 16 and 17.

The OLED, 2004 and 1602 displays use the serial I2C bus. There must be definitions for this in the preferences, for example:

```
pin_tft_scl = 13                              # GPIO Pin number for SCL
pin_tft_sda = 14                              # GPIO Pin number for SDA
```

Th I2C address of the 2004 and 1602 displays is defined in the corresponding module.

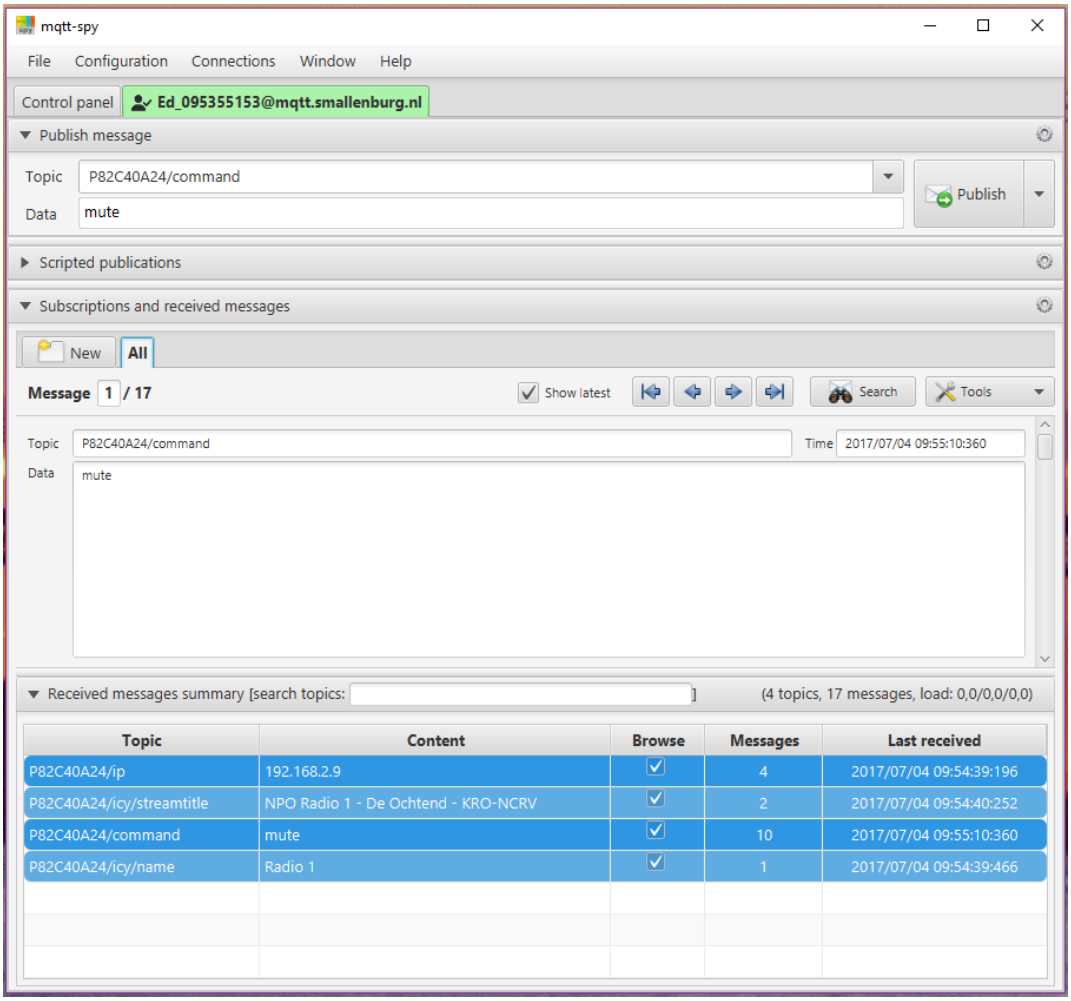
MQTT interface.

The MQTT interface can handle the same commands as the web interface.
As publish command on a Linux system may look like:

```
$ mosquitto_pub -h broker.hivemq.com -t espradio -m volume=80
```

Note that `broker.hivemq.com` is heavily used, this may cause some delay. If you use your own broker the reaction on commands will be much better.
Remove the lines starting with “mqtt” if no MQTT is required or set mqttbroker to "none".

You can use an MQTT client like https://kamilfb.github.io/mqtt-spy/ to view the MQTT interface.



Subscribe to “P82C40A24/ip” and you will see the IP-address of your radio. You will see the IP address of your radio or the IP address of a different user. So be sure to use unique names for your topics. This can be accomplished by specifying a unique prefix in the preferences, see below.
The parameters in preferences for this example are:

```
# MQTT broker, credentials and topic  to subscribe
mqttbroker = broker.hivemq.com          # Broker to connect with
mqttprefix = P82C40A24                  # Prefix for pub/sub.  Default is part of MAC-address.
mqttport = 1883                         # Portnumber (1883 is default)
mqttuser = none                         # (No) username for broker
mqttpasswd = none                       # (No) password for broker
#
```

In this example I published the command “mute” to the radio. The radio published the IP-address 192.168.2.8 to the broker (once every 10 minutes).

MQTT PUB topics.

In this version 7 topics will be published to MQTT and can be subscribed to by an MQTT client:

prefix/ip	The IP-address of the ESP32-Radio
prefix/icy/name	The name of the station
prefix/icy/streamtitle	The name of the stream
prefix/nowplaying	Track information
prefix/preset	Preset currently plaing
prefix/volume	Current volume setting
prefix/playing	1 if playing, 0 if stopped

Command to control the ESP32-Radio can be published to "prefix/command".

Alternative way of configuration.

An extra sketch "Esp32_radio_init" is supplied as an alternative to initialize the preferences (in Non-Voltile Storage of the ESP32). Just change lines 39 and 40 (the specs for WiFi networks) to match your network(s).

Upload and run the sketch once and then load the ESP32-Radio.

OTA update through remote server.

The software can be update by uploading the compiled project to the ESP32. There is also a build-in feature to load the latest version of the software from a remote server. The software for the NEXTION (if in use) will also be updated.

The update can be initiated by the “update”-command through the serial interface or the webinterface.

The update will fail if there is no new version available or if no connection can be made to the server.

The radio connects to the host “smallenburg.nl” and download the file “`/Arduino/Esp32_radio/firmware.bin`” and “`/Arduino/ESP32-Radio.tft`”. The binary file is flashed to the ESP32 and the NEXTION and will start the new version. The process may take some 2 to 3 minutes.

Compiler/upload output.

```
> Executing task in folder ESP32-Radio: C:\Users\hp\.platformio\penv\Scripts\platformio.exe
        run --target upload <

Processing esp32doit-devkit-v1 (platform: espressif32; board: esp32doit-devkit-v1;
                                framework: arduino)
----------------------------Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/esp32doit-devkit-v1.html
PLATFORM: Espressif 32 (3.3.0) > DOIT ESP32 DEVKIT V1
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (esp-prog) External (esp-prog, iot-bus-jtag, jlink, minimodule, olimex-arm-usb-
ocd, olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa)
PACKAGES:
 - framework-arduinoespressif32 3.10006.210326 (1.0.6)
 - tool-esptoolpy 1.30100.210531 (3.1.0)
 - tool-mkspiffs 2.230.0 (2.30)
 - toolchain-xtensa32 2.50200.97 (5.2.0)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 33 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <Wire> 1.0.1
|-- <SD(esp32)> 1.0.5
|   |-- <FS> 1.0
|   |-- <SPI> 1.0
|-- <PubSubClient> 2.8.0
|-- <Adafruit ST7735 and ST7789 Library> 1.6.0
|   |-- <Adafruit GFX Library> 1.10.4
|   |   |-- <SPI> 1.0
|   |   |-- <Adafruit BusIO> 1.7.1
|   |   |   |-- <Wire> 1.0.1
|   |   |   |-- <SPI> 1.0
|   |   |-- <Wire> 1.0.1
|   |-- <SPI> 1.0
|-- <Adafruit BusIO> 1.7.1
|   |-- <Wire> 1.0.1
|   |-- <SPI> 1.0
|-- <Adafruit GFX Library> 1.10.4
|   |-- <SPI> 1.0
|   |-- <Adafruit BusIO> 1.7.1
|   |   |-- <Wire> 1.0.1
|   |   |-- <SPI> 1.0
|   |-- <Wire> 1.0.1
|-- <Ch376msc> 1.4.4
|   |-- <SPI> 1.0
|-- <FS> 1.0
|-- <SPI> 1.0
|-- <ArduinoOTA> 1.0
|   |-- <Update> 1.0
|   |-- <WiFi> 1.0
|   |-- <ESPmDNS> 1.0
|   |   |-- <WiFi> 1.0
|-- <ESPmDNS> 1.0
|   |-- <WiFi> 1.0
|-- <Update> 1.0
|-- <WiFi> 1.0
Building in release mode
Compiling .pio\build\esp32doit-devkit-v1\src\main.cpp.o
Generating partitions .pio\build\esp32doit-devkit-v1\partitions.bin
Compiling .pio\build\esp32doit-devkit-v1\lib674\Wire\Wire.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib5dd\FS\FS.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib5dd\FS\vfs_api.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb0c\SPI\SPI.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib4d3\SD\SD.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib4d3\SD\sd_diskio.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\libb0c\libSPI.a
Compiling .pio\build\esp32doit-devkit-v1\lib4d3\SD\sd_diskio_crc.c.o
Compiling .pio\build\esp32doit-devkit-v1\lib19e\PubSubClient\PubSubClient.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib297\Adafruit BusIO\Adafruit_BusIO_Register.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib674\libWire.a
Compiling .pio\build\esp32doit-devkit-v1\lib297\Adafruit BusIO\Adafruit_I2CDevice.cpp.o
```

```
Compiling .pio\build\esp32doit-devkit-v1\lib297\Adafruit BusIO\Adafruit_SPIDevice.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib5dd\libFS.a
Compiling .pio\build\esp32doit-devkit-v1\lib0ed\Adafruit GFX Library\Adafruit_GFX.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib0ed\Adafruit GFX Library\Adafruit_GrayOLED.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib4d3\libSD.a
Compiling .pio\build\esp32doit-devkit-v1\lib0ed\Adafruit GFX Library\Adafruit_SPITFT.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib0ed\Adafruit GFX Library\glcdfont.c.o
Archiving .pio\build\esp32doit-devkit-v1\lib19e\libPubSubClient.a
Compiling .pio\build\esp32doit-devkit-v1\lib81f\Adafruit ST7735 and ST7789
Library\Adafruit_ST7735.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib81f\Adafruit ST7735 and ST7789
Library\Adafruit_ST7789.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib81f\Adafruit ST7735 and ST7789
Library\Adafruit_ST77xx.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib297\libAdafruit BusIO.a
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\API.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\Ch376msc.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\Comm.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\Read.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\SetGet.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libb1e\Ch376msc\Write.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib81f\libAdafruit ST7735 and ST7789 Library.a
Compiling .pio\build\esp32doit-devkit-v1\lib1b7\Update\HttpsOTAUpdate.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib0ed\libAdafruit GFX Library.a
Compiling .pio\build\esp32doit-devkit-v1\lib1b7\Update\Updater.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\ETH.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFi.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiAP.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiClient.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiGeneric.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\libb1e\libCh376msc.a
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiMulti.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiSTA.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib1b7\libUpdate.a
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiScan.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiServer.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib160\WiFi\WiFiUdp.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\libbf8\ESPmDNS\ESPmDNS.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\lib8b7\ArduinoOTA\ArduinoOTA.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\libFrameworkArduinoVariant.a
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\Esp.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\FunctionalInterrupt.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\HardwareSerial.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\IPAddress.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\IPv6Address.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib160\libWiFi.a
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\MD5Builder.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\Print.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\libbf8\libESPmDNS.a
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\Stream.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\StreamString.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\WMath.cpp.o
Archiving .pio\build\esp32doit-devkit-v1\lib8b7\libArduinoOTA.a
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\WString.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\base64.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\cbuf.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-adc.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-bt.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-cpu.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-dac.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-gpio.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-i2c.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-ledc.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-log.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-matrix.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-misc.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-psram.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-rmt.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-sigmadelta.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-spi.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-time.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-timer.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-touch.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\esp32-hal-uart.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\libb64\cdecode.c.o
```

```
C:\users\hp\.platformio\packages\framework-arduinoespressif32\cores\esp32\esp32-hal-spi.c: In
function 'spiTransferBytesNL':
C:\users\hp\.platformio\packages\framework-arduinoespressif32\cores\esp32\esp32-hal-
spi.c:922:39: warning: initialization from incompatible pointer type [-Wincompatible-pointer-
types]
                uint8_t * last_out8 = &result[c_longs-1];
                                      ^
C:\users\hp\.platformio\packages\framework-arduinoespressif32\cores\esp32\esp32-hal-
spi.c:923:40: warning: initialization from incompatible pointer type [-Wincompatible-pointer-
types]
                uint8_t * last_data8 = &last_data;
                                       ^
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\libb64\cencode.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\main.cpp.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\stdlib_noniso.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\wiring_pulse.c.o
Compiling .pio\build\esp32doit-devkit-v1\FrameworkArduino\wiring_shift.c.o
Archiving .pio\build\esp32doit-devkit-v1\libFrameworkArduino.a
Linking .pio\build\esp32doit-devkit-v1\firmware.elf
Retrieving maximum program size .pio\build\esp32doit-devkit-v1\firmware.elf
Checking size .pio\build\esp32doit-devkit-v1\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [==        ]  18.6% (used 61092 bytes from 327680 bytes)
Flash: [=======   ]  66.5% (used 871018 bytes from 1310720 bytes)
Building .pio\build\esp32doit-devkit-v1\firmware.bin
esptool.py v3.1
Merged 1 ELF section
Configuring upload protocol...
AVAILABLE: esp-prog, espota, esptool, iot-bus-jtag, jlink, minimodule, olimex-arm-usb-ocd,
olimex-arm-usb-ocd-h, olimex-arm-usb-tiny-h, olimex-jtag-tiny, tumpa
CURRENT: upload_protocol = esptool
Looking for upload port...
Use manually specified: COM4
Uploading .pio\build\esp32doit-devkit-v1\firmware.bin
esptool.py v3.1
Serial port COM4
Connecting....
Chip is ESP32-D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 24:0a:c4:ae:84:48
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Auto-detected Flash size: 4MB
Flash will be erased from 0x00001000 to 0x00005fff...
Flash will be erased from 0x00008000 to 0x00008fff...
Flash will be erased from 0x0000e000 to 0x0000ffff...
Flash will be erased from 0x00010000 to 0x000e4fff...
Compressed 17104 bytes to 11191...
Writing at 0x00001000... (100 %)
Wrote 17104 bytes (11191 compressed) at 0x00001000 in 0.5 seconds (effective 260.5 kbit/s)...
Hash of data verified.
Compressed 3072 bytes to 128...
Writing at 0x00008000... (100 %)
Wrote 3072 bytes (128 compressed) at 0x00008000 in 0.1 seconds (effective 365.0 kbit/s)...
Hash of data verified.
Compressed 8192 bytes to 47...
Writing at 0x0000e000... (100 %)
Wrote 8192 bytes (47 compressed) at 0x0000e000 in 0.1 seconds (effective 552.0 kbit/s)...
Hash of data verified.
Compressed 871120 bytes to 494819...
Writing at 0x0001c678... (6 %)
Writing at 0x00028a04... (9 %)
Writing at 0x0003528e... (12 %)
Writing at 0x0004b4fe... (16 %)
Writing at 0x00050f18... (19 %)
Writing at 0x00056b08... (22 %)
Writing at 0x0005c483... (25 %)
Writing at 0x00061f04... (29 %)
Writing at 0x0006715a... (32 %)
. . .
```

```
. . .
Writing at 0x000d7581... (93 %)
Writing at 0x000dd645... (96 %)
Writing at 0x000e3377... (100 %)
Wrote 871120 bytes (494819 compressed) at 0x00010000 in 12.0 seconds (effective 580.1 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
======================================== [SUCCESS] Took 42.10 seconds
========================================
Terminal will be reused by tasks, press any key to close it.
```

Debug (serial 115200 Baud) output.

This is an example of the debug output.

```
ets Jun  8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:10124
load:0x40080400,len:5828
entry 0x400806a8

D: Starting ESP32-radio running on CPU 1 at 240 MHz.  Version Mon, 28 Jun 2021 12:40:00 GMT.  Free memory 302164
D: Display type is BLUETFT
D: Partition nvs found, 20480 bytes
D: Read 51 keys from NVS
D: pin_ir set to 35
D: pin_enc_clk set to -1
D: pin_enc_dt set to -1
D: pin_enc_sw set to -1
D: pin_tft_cs set to 15
D: pin_tft_dc set to 2
D: pin_tft_scl set to 13
D: pin_tft_sda set to 14
D: pin_tft_bl set to -1
D: pin_tft_blx set to -1
D: pin_sd_cs set to 21
D: pin_ch376_cs set to 32
D: pin_ch376_int set to 33
D: pin_vs_cs set to 5
D: pin_vs_dcs set to 16
D: pin_vs_dreq set to 4
D: pin_shutdown set to -1
D: pin_shutdownx set to -1
D: pin_spi_sck set to 18
D: pin_spi_miso set to 19
D: pin_spi_mosi set to 23
D: GPIO0 is HIGH
D: GPIO2 is LOW, probably no PULL-UP
D: GPIO4 is HIGH
D: GPIO5 is HIGH
D: GPIO12 is HIGH
D: GPIO13 is HIGH
D: GPIO14 is HIGH
D: GPIO15 is HIGH
D: GPIO16 is HIGH
D: GPIO17 is HIGH
D: GPIO18 is HIGH
D: GPIO19 is HIGH
D: GPIO21 is HIGH
D: GPIO22 is HIGH
D: GPIO23 is HIGH
D: GPIO25 is HIGH
D: GPIO26 is HIGH
D: GPIO27 is HIGH
D: GPIO32 is HIGH
D: GPIO33 is HIGH
D: GPIO34 is LOW, probably no PULL-UP
D: GPIO35 is LOW, probably no PULL-UP
D: GPIO39 is LOW, probably no PULL-UP
D: gpio_00 will execute uppreset = 1
D: gpio_12 will execute upvolume = 2
D: gpio_34 will execute station = icecast.omroep.nl:80/radio1-bb-mp3
D: Enable pin 35 for IR
D: Start display
D: Create list with acceptable WiFi networks
D: Added ADSL-11 to list of networks
D: Added SSID to list of networks
D: End adding networks
D: Scan Networks
D: Scan completed
D: Number of available networks: 7
D:  1 - ADSL-11                  Signal: -38 dBm, Encryption WPA2_PSK, Acceptable
D:  2 - ESP_A43E4F               Signal: -56 dBm, Encryption OPEN,
D:  3 - NETGEAR-11               Signal: -64 dBm, Encryption WPA2_PSK,
D:  4 - Ziggo8571367             Signal: -81 dBm, Encryption WPA2_PSK,
D:  5 - ADSL-11                  Signal: -81 dBm, Encryption WPA2_PSK, Acceptable
D:  6 - Ziggo                    Signal: -83 dBm, Encryption ????,
D:  7 - ADSL-11                  Signal: -89 dBm, Encryption WPA_WPA2_PSK, Acceptable
D: End of list
D: Command: clk_dst with parameter 1
D: Command: clk_offset with parameter 1
D: Command: clk_server with parameter pool.ntp.org
D: Command: gpio_00 with parameter uppreset = 1
```

```
D: Command: gpio_12 with parameter upvolume = 2
D: Command: gpio_13 with parameter downvolume = 2
D: Command: gpio_34 with parameter station = icecast.omroep.nl:80/radio1-bb-mp3
D: Command: ir_40bf with parameter upvolume = 2
D: Command: ir_c03f with parameter downvolume = 2
D: Command: lstmods with parameter Sun, 25 Apr 2021 12:45:21 GMT
D: Command: mqqprefix with parameter none
D: Command: mqttbroker with parameter none
D: Command: mqttpasswd with parameter *******
D: Command: mqttport with parameter 1883
D: Command: mqttuser with parameter none
D: Command: pin_ch376_cs with parameter 32
D: Command: pin_ch376_int with parameter 33
D: Command: pin_ir with parameter 35
D: Command: pin_sd_cs with parameter 21
D: Command: pin_tft_cs with parameter 15
D: Command: pin_tft_dc with parameter 2
D: Command: pin_tft_scl with parameter 13
D: Command: pin_tft_sda with parameter 14
D: Command: pin_vs_cs with parameter 5
D: Command: pin_vs_dcs with parameter 16
D: Command: pin_vs_dreq with parameter 4
D: Command: preset with parameter 16
D: Command: preset_000 with parameter 109.206.96.34:8100
D: Command: preset_001 with parameter airspectrum.cdnstream1.com:8114/1648_128
D: Command: preset_002 with parameter airspectrum.cdnstream1.com:8142/1303_128
D: Command: preset_003 with parameter airspectrum.cdnstream1.com:8000/1261_192
D: Command: preset_004 with parameter airspectrum.cdnstream1.com:8008/1604_128
D: Command: preset_005 with parameter us1.internet-radio.com:8105
D: Command: preset_006 with parameter icecast.omroep.nl:80/radio1-bb-mp3
D: Command: preset_007 with parameter icecast.omroep.nl:80/radio5-bb-mp3
D: Command: preset_008 with parameter icecast.omroep.nl:80/radio2-bb-mp3
D: Command: preset_009 with parameter streaming.hotmix-radio.net/hotmixradio-90-128.mp3
D: Command: preset_010 with parameter streams.radiobob.de/bob-80srock/mp3-192/streams.radiobob.de
D: Command: preset_011 with parameter wdr-edge-101d-dus-dtag-cdn.cast.addradio.de/wdr/diemaus/live/mp3/high
D: Command: preset_012 with parameter stream.laut.fm/kinderlieder-123
D: Command: preset_013 with parameter bbcwssc.ic.llnwd.net/stream/bbcwssc_mp1_ws-eieuk
D: Command: preset_014 with parameter metafiles.gl-systemhaus.de/hr/hr1_2.m3u
D: Command: preset_015 with parameter cbcmp3.ic.llnwd.net/stream/cbcmp3_M-7QMTL0_MTL
D: Command: preset_016 with parameter streamer.radio.co/sa7bf61271/listen
D: Command: toneha with parameter 5
D: Command: tonehf with parameter 1
D: Command: tonela with parameter 8
D: Command: tonelf with parameter 11
D: Command: volume with parameter 96
D: Slow SPI, Testing VS1053 read/write registers...
D: Fast SPI, Testing VS1053 read/write registers again...
D: endFillByte is 0
D: Connect to WiFi
D: Connected to ADSL-11
D: IP = 192.168.2.25
D: Start server for commands
D: Network found. Starting mqtt and OTA
D: MDNS responder started
D: Rotary encoder is disabled (-1/-1/-1)
D: Sync TOD
D: Sync TOD, new value is 09:02:23
D: STOP requested
D: New preset/file requested (16/0) from streamer.radio.co/sa7bf61271/listen
D: New station request
D: Connect to new host streamer.radio.co/sa7bf61271/listen
D: Connect to streamer.radio.co on port 80, extension /sa7bf61271/listen
D: Song stopped correctly after 0 msec
D: Connected to server
D: Switch to HEADER
D: Headerline: Content-Type: audio/mpeg
D: audio/mpeg seen.
D: Headerline: icy-br:192
D: Headerline: ice-audio-info: channels=2;samplerate=44100;bitrate=192
D: Headerline: icy-br:192
D: Headerline: icy-name:JB&#39;s Rock N Roll - DooWop Jukebox
D: Headerline: icy-pub:1
D: Headerline: icy-url:http://www.jbs-rocknroll-doowop-jukebox.com
D: Headerline: Server: streamer-gravelines.radio.co
D: Headerline: Cache-Control: no-cache, no-store
D: Headerline: Access-Control-Allow-Origin: *
D: Headerline: Access-Control-Allow-Headers: Origin, Accept, X-Requested-With, Content-Type
D: l-Allow-Headers: Origin, Accept, X-Requested-With, Content-Type seen.
D: Headerline: Access-Control-Allow-Methods: GET, OPTIONS, HEAD
D: Headerline: Connection: Close
D: Headerline: Expires: Mon, 26 Jul 1997 05:00:00 GMT
D: Headerline: icy-metaint:16000
D: Switch to DATA, bitrate is 192, metaint is 16000
D: Duration mp3loop 79
D: Metadata block 48 bytes
D: Streamtitle found, 45 bytes
D: StreamTitle='The Passions - I Only Want You';
```