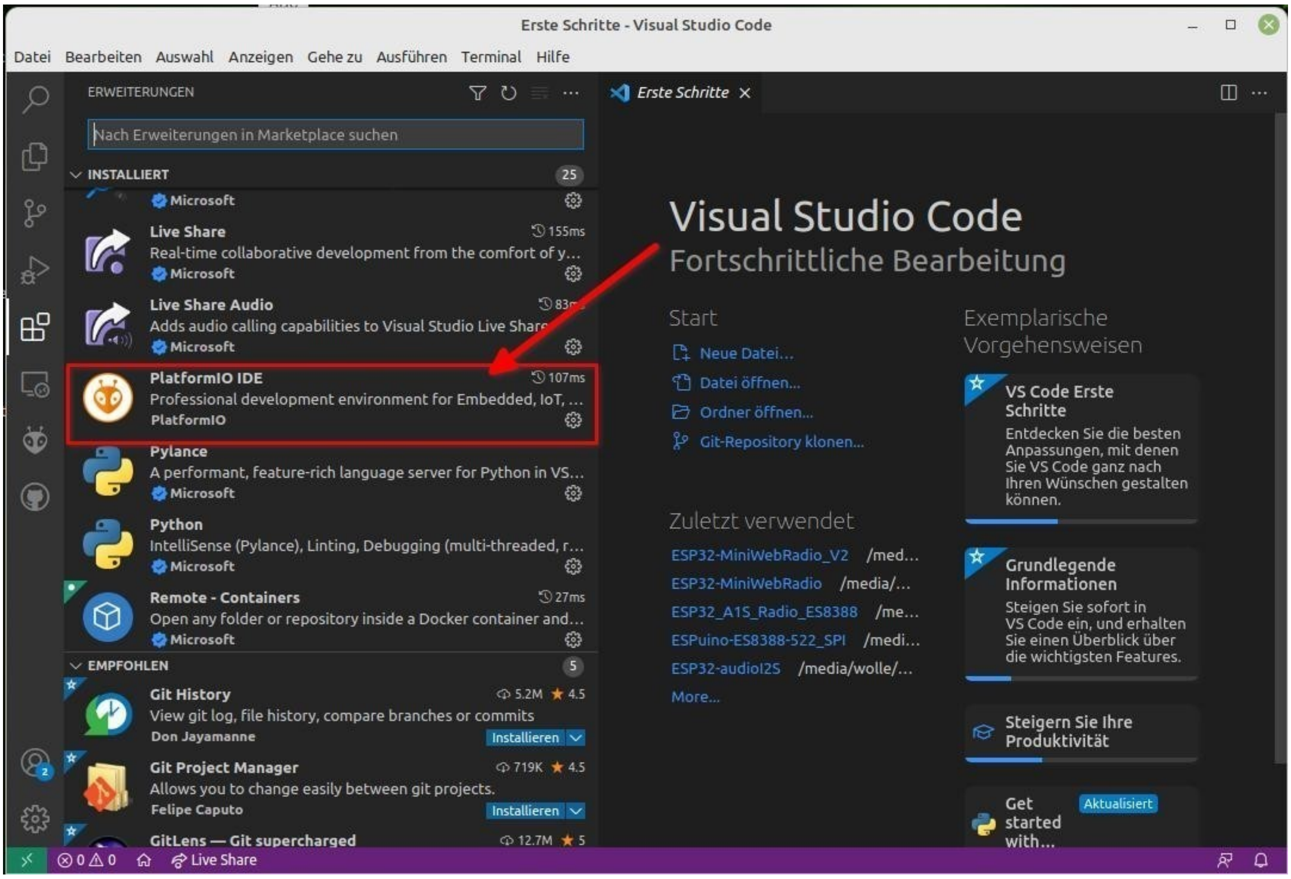
How to install ESP32-MiniWebRadio-V4

1) Install **Visual Studio Code** on your PC
2) Add extension **PlatformIO IDE**


Erste Schritte - Visual Studio Code
Datei Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe
ERWEITERUNGEN
Nach Erweiterungen in Marketplace suchen
INSTALLIERT 25
EMPFOHLEN 5
Erste Schritte
Installieren
Aktualisiert
Live Share

Live Share — Real-time collaborative development from the comfort of y… — Microsoft — 155ms

Live Share Audio — Adds audio calling capabilities to Visual Studio Live Share — Microsoft

PlatformIO IDE — Professional development environment for Embedded, IoT, … — PlatformIO — 107ms

Pylance — A performant, feature-rich language server for Python in VS… — Microsoft

Python — IntelliSense (Pylance), Linting, Debugging (multi-threaded, r… — Microsoft

Remote - Containers — Open any folder or repository inside a Docker container and… — Microsoft — 27ms

Git History — View git log, file history, compare branches or commits — Don Jayamanne — 5.2M ★ 4.5

Git Project Manager — Allows you to change easily between git projects. — Felipe Caputo — 719K ★ 4.5

GitLens — Git supercharged — 12.7M ★ 5

Visual Studio Code

Fortschrittliche Bearbeitung

Start: Neue Datei… / Datei öffnen… / Ordner öffnen… / Git-Repository klonen…

Zuletzt verwendet: ESP32-MiniWebRadio_V2 /med… ; ESP32-MiniWebRadio /media/… ; ESP32_A1S_Radio_ES8388 /me… ; ESPuino-ES8388-522_SPI /medi… ; ESP32-audioI2S /media/wolle/… ; More…

Exemplarische Vorgehensweisen

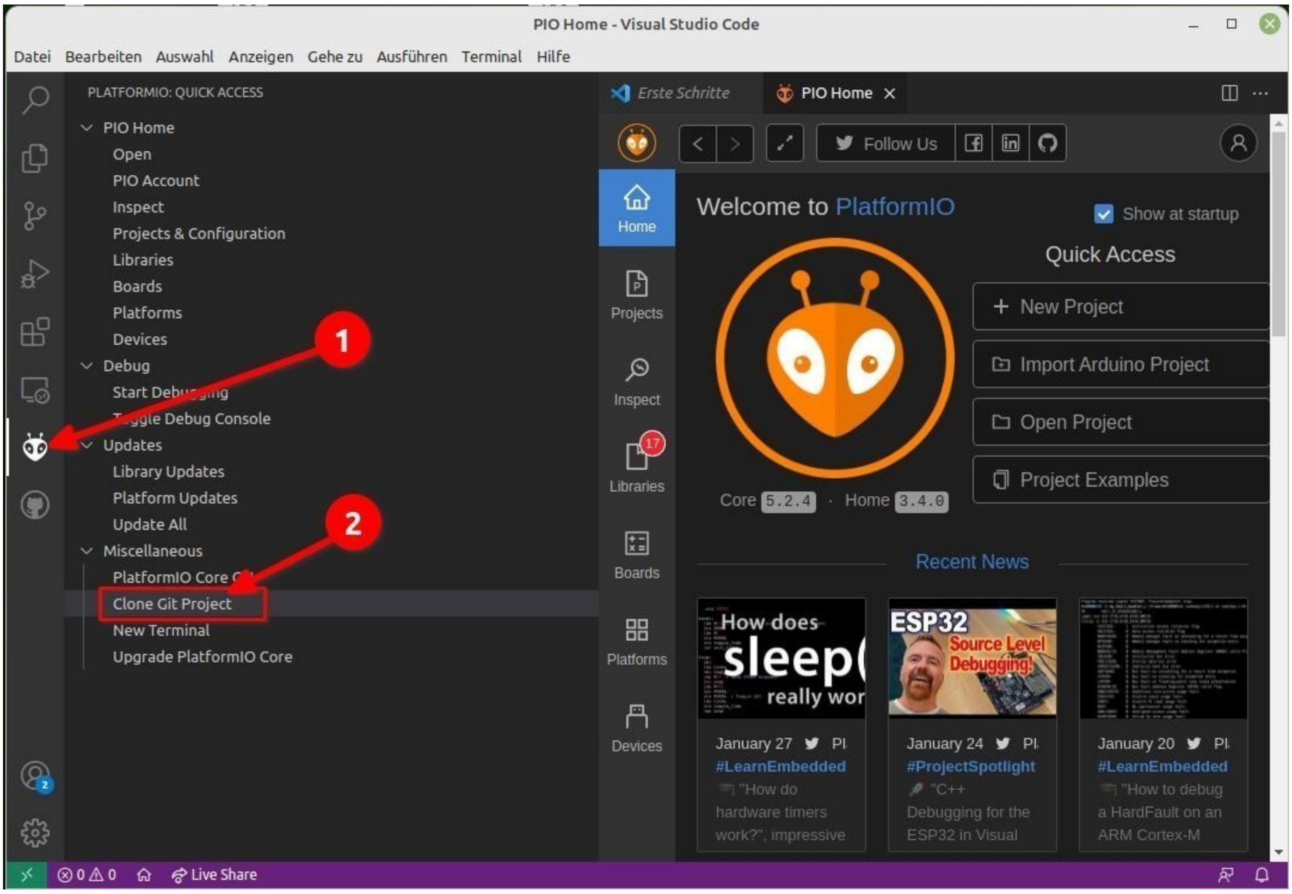
VS Code Erste Schritte — Entdecken Sie die besten Anpassungen, mit denen Sie VS Code ganz nach Ihren Wünschen gestalten können.

Grundlegende Informationen — Steigen Sie sofort in VS Code ein, und erhalten Sie einen Überblick über die wichtigsten Features.

Steigern Sie Ihre Produktivität

Get started with…

3) open **PlatformIO** and select **Clone Git Project**

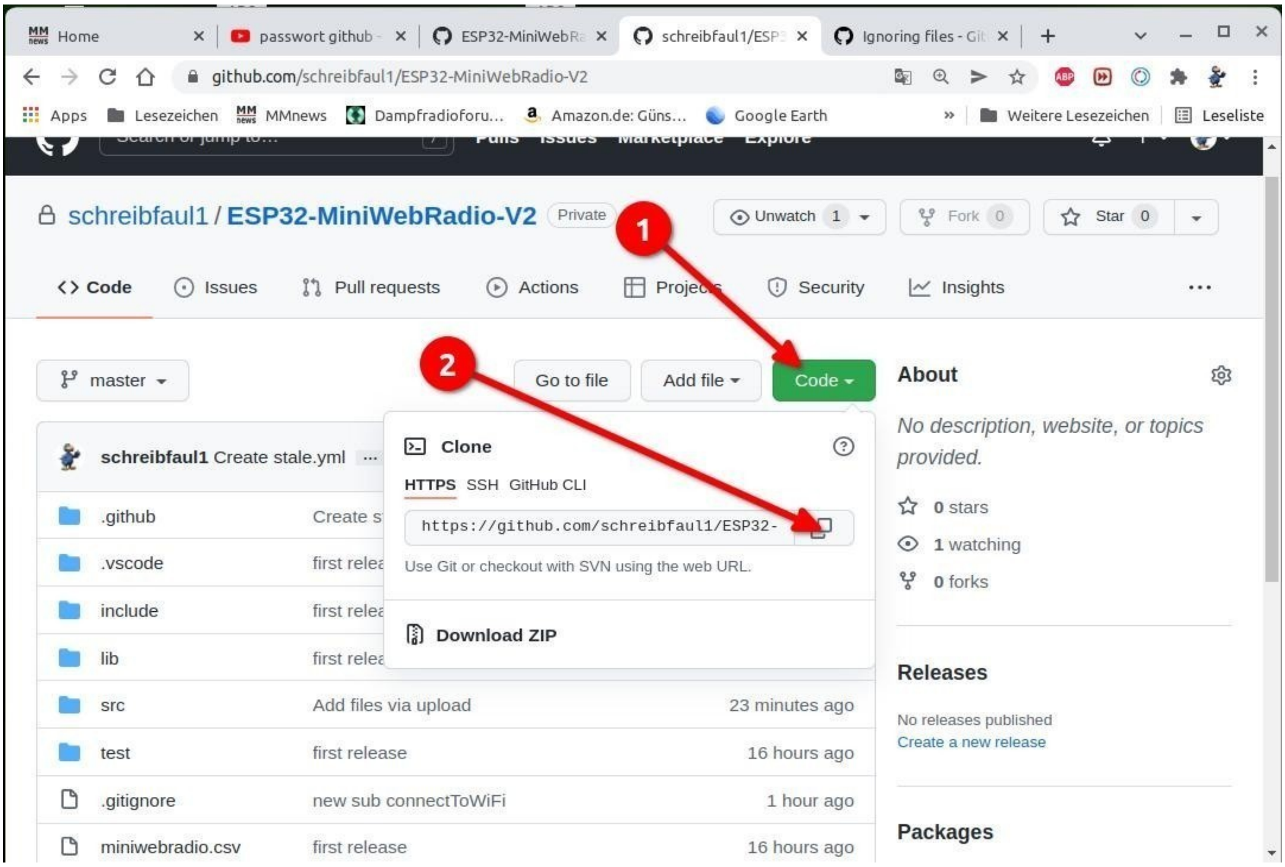


January 27 — #LearnEmbedded — "How do hardware timers work?", impressive

January 24 — #ProjectSpotlight — "C++ Debugging for the ESP32 in Visual

January 20 — #LearnEmbedded — "How to debug a HardFault on an ARM Cortex-M

4) goto Github, press **Code** and copy the URL



github.com/schreibfaul1/ESP32-MiniWebRadio-V2

https://github.com/schreibfaul1/ESP32-

Use Git or checkout with SVN using the web URL.

schreibfaul1 Create stale.yml

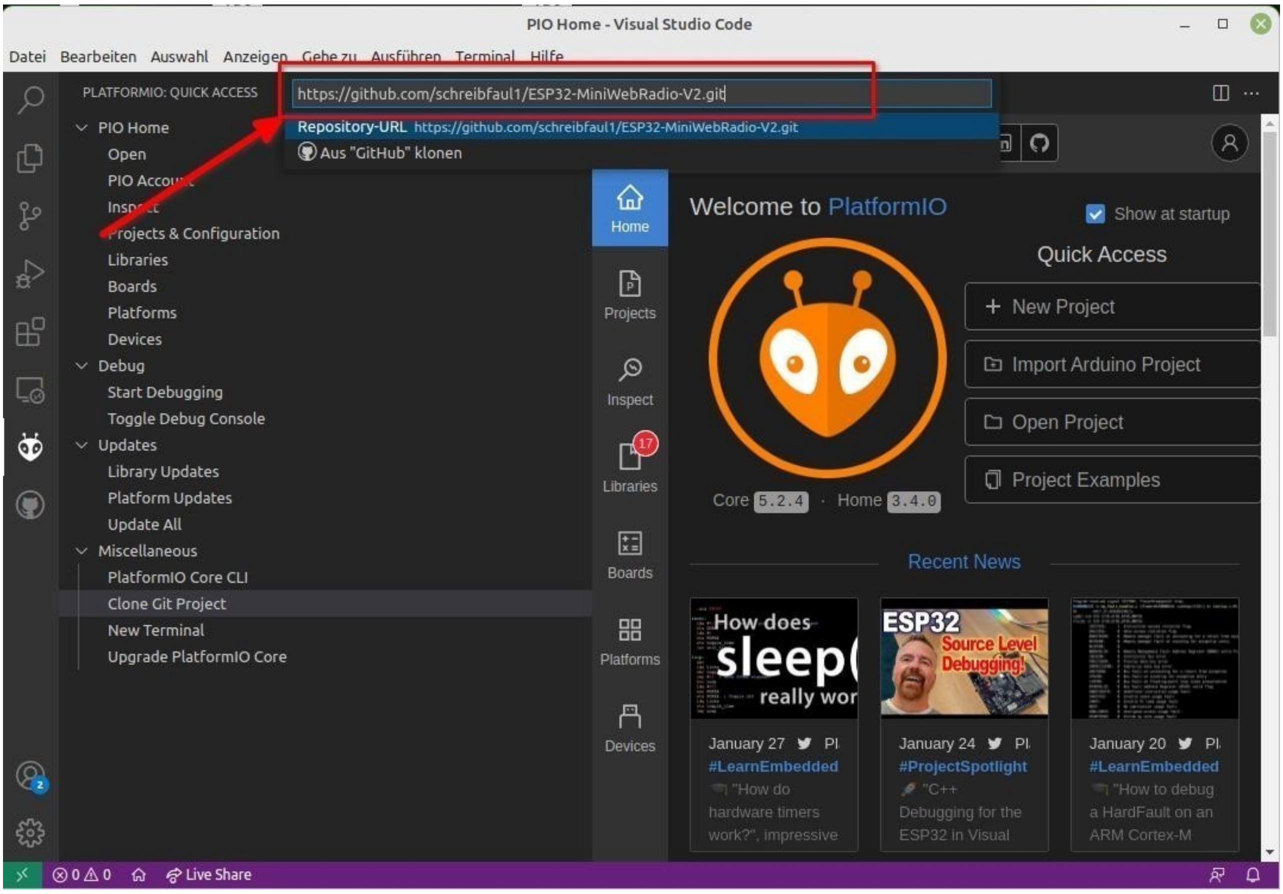
.github	Create s	
.vscode	first relea	
include	first relea	
lib	first relea	
src	Add files via upload	23 minutes ago
test	first release	16 hours ago
.gitignore	new sub connectToWiFi	1 hour ago
miniwebradio.csv	first release	16 hours ago

No description, website, or topics provided.

0 stars
1 watching
0 forks

No releases published
Create a new release

5) paste the URL in PlatformIO, press ENTER and choose a folder on your PC

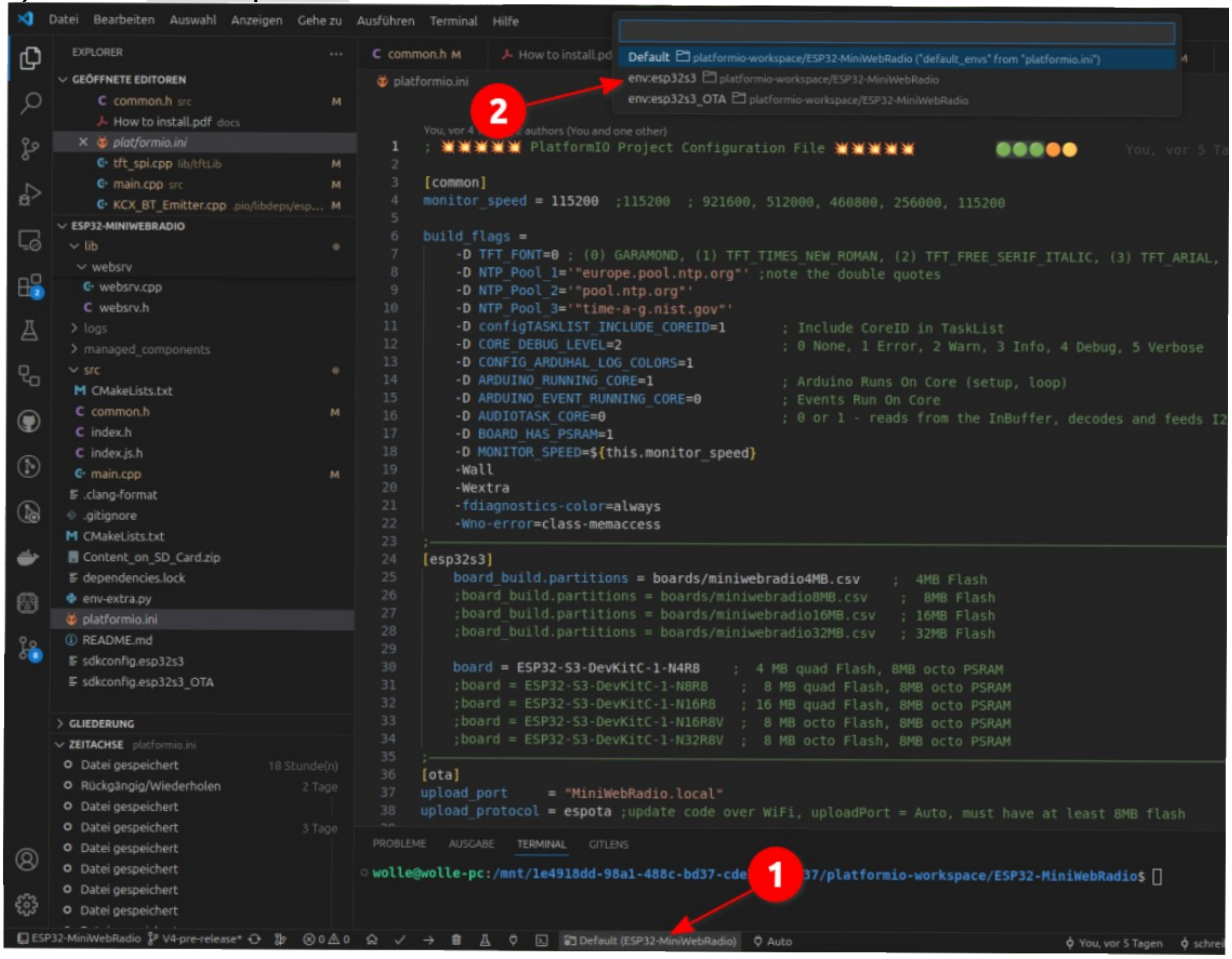


https://github.com/schreibfaul1/ESP32-MiniWebRadio-V2.git

Repository-URL https://github.com/schreibfaul1/ESP32-MiniWebRadio-V2.git

Aus "GitHub" klonen

6) select env:esp32s3

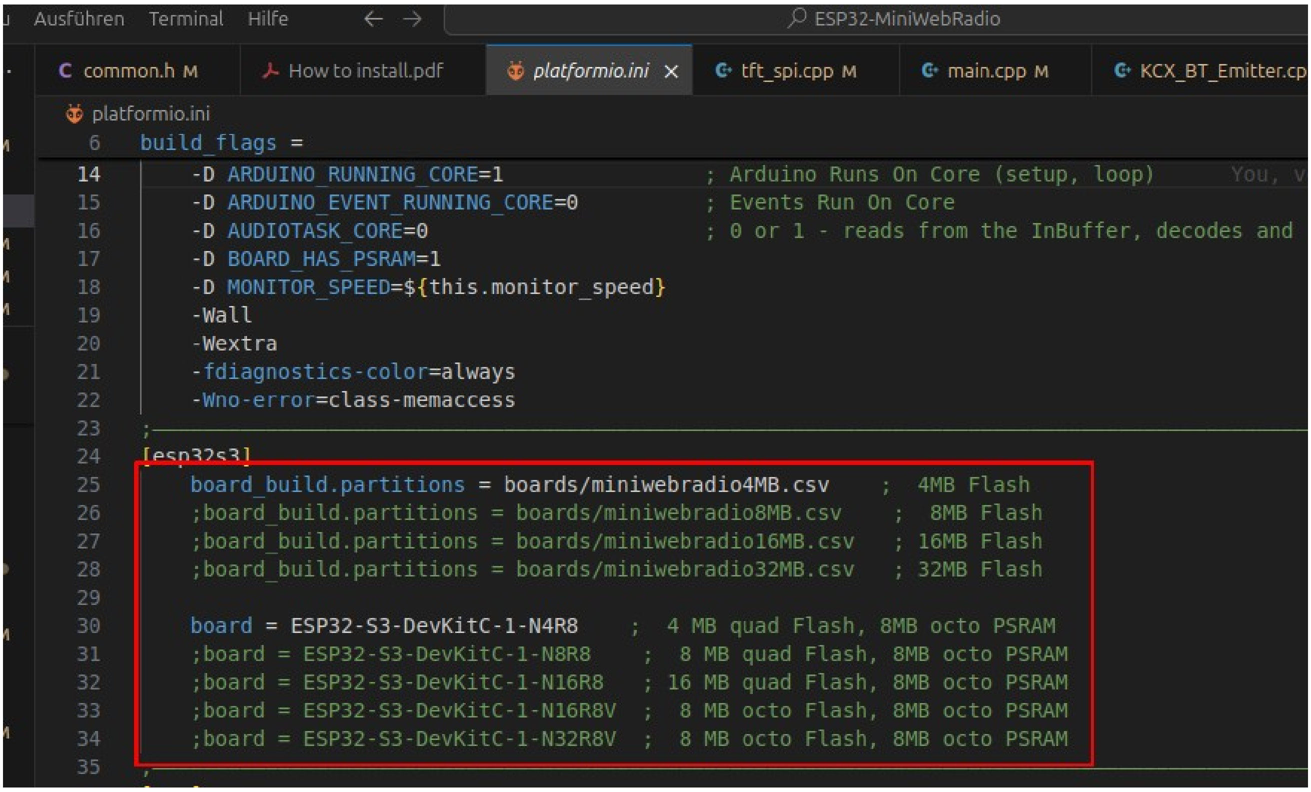


Default platformio-workspace/ESP32-MiniWebRadio ("default_envs" from "platformio.ini")
env:esp32s3 platformio-workspace/ESP32-MiniWebRadio
env:esp32s3_OTA platformio-workspace/ESP32-MiniWebRadio

```
; ✸✸✸✸✸ PlatformIO Project Configuration File ✸✸✸✸✸

[common]
monitor_speed = 115200  ;115200  ; 921600, 512000, 460800, 256000, 115200

build_flags =
    -D TFT_FONT=0 ; (0) GARAMOND, (1) TFT_TIMES_NEW_ROMAN, (2) TFT_FREE_SERIF_ITALIC, (3) TFT_ARIAL,
    -D NTP_Pool_1='"europe.pool.ntp.org"' ;note the double quotes
    -D NTP_Pool_2='"pool.ntp.org"'
    -D NTP_Pool_3='"time-a-g.nist.gov"'
    -D configTASKLIST_INCLUDE_COREID=1       ; Include CoreID in TaskList
    -D CORE_DEBUG_LEVEL=2                    ; 0 None, 1 Error, 2 Warn, 3 Info, 4 Debug, 5 Verbose
    -D CONFIG_ARDUHAL_LOG_COLORS=1
    -D ARDUINO_RUNNING_CORE=1                ; Arduino Runs On Core (setup, loop)
    -D ARDUINO_EVENT_RUNNING_CORE=0          ; Events Run On Core
    -D AUDIOTASK_CORE=0                      ; 0 or 1 - reads from the InBuffer, decodes and feeds I2
    -D BOARD_HAS_PSRAM=1
    -D MONITOR_SPEED=${this.monitor_speed}
    -Wall
    -Wextra
    -fdiagnostics-color=always
    -Wno-error=class-memaccess
;
[esp32s3]
    board_build.partitions = boards/miniwebradio4MB.csv    ;  4MB Flash
    ;board_build.partitions = boards/miniwebradio8MB.csv    ;  8MB Flash
    ;board_build.partitions = boards/miniwebradio16MB.csv   ; 16MB Flash
    ;board_build.partitions = boards/miniwebradio32MB.csv   ; 32MB Flash

    board = ESP32-S3-DevKitC-1-N4R8    ;  4 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N8R8    ;  8 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N16R8   ; 16 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N16R8V  ;  8 MB octo Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N32R8V  ;  8 MB octo Flash, 8MB octo PSRAM
;
[ota]
upload_port     = "MiniWebRadio.local"
upload_protocol = espota ;update code over WiFi, uploadPort = Auto, must have at least 8MB flash
```

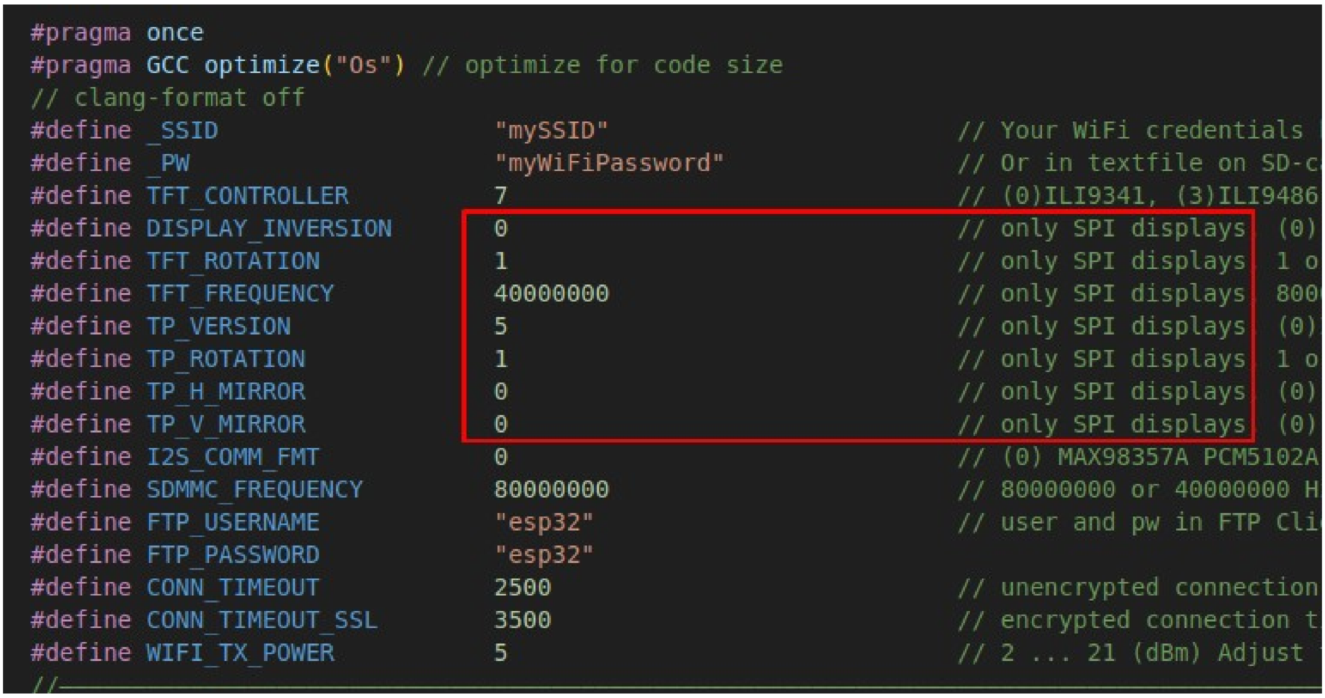
```
wolle@wolle-pc:/mnt/1e4918dd-98a1-488c-bd37-cde…37/platformio-workspace/ESP32-MiniWebRadio$
```

7) select the appropriate board and partition in platform.ini



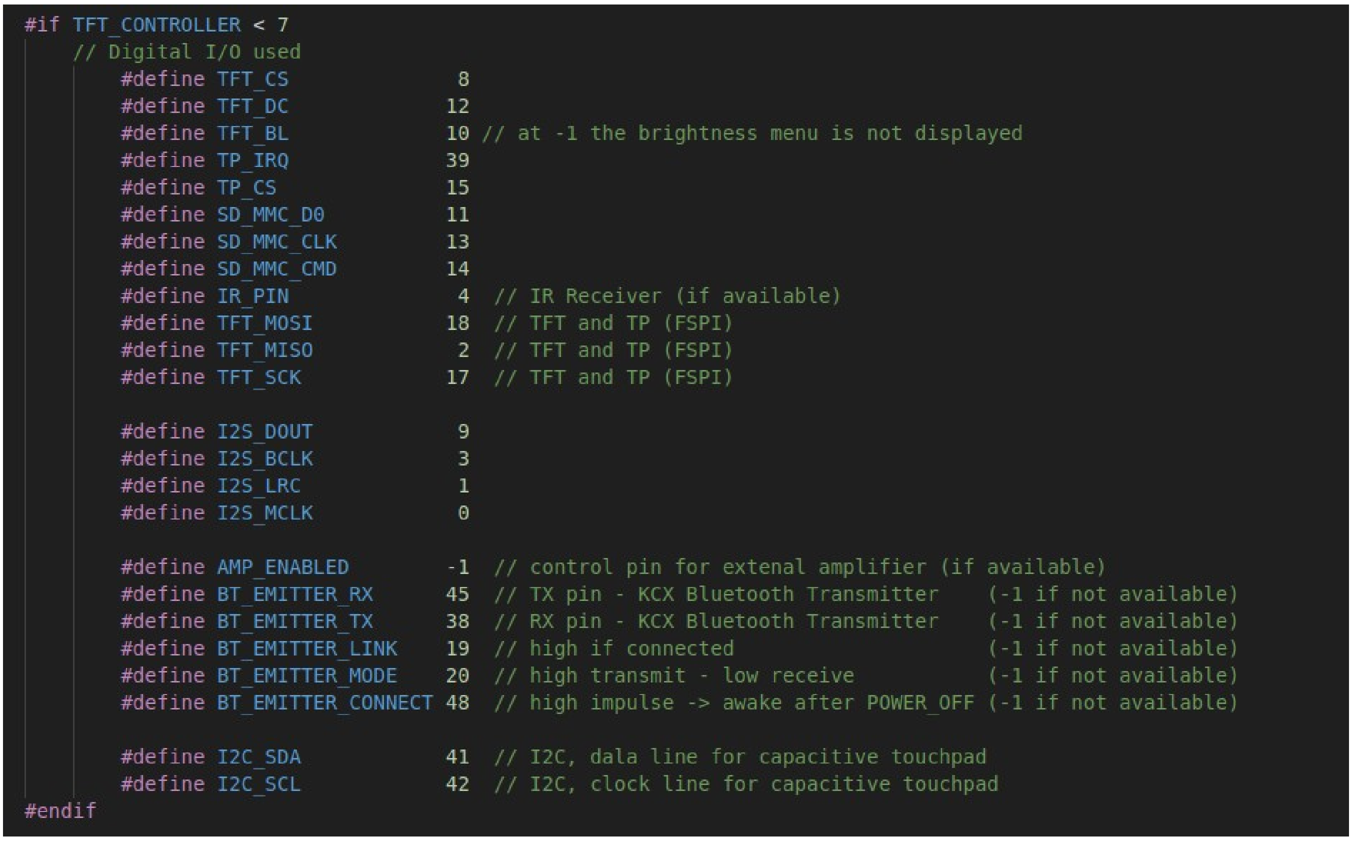
```
build_flags =
    -D ARDUINO_RUNNING_CORE=1                ; Arduino Runs On Core (setup, loop)
    -D ARDUINO_EVENT_RUNNING_CORE=0          ; Events Run On Core
    -D AUDIOTASK_CORE=0                      ; 0 or 1 - reads from the InBuffer, decodes and
    -D BOARD_HAS_PSRAM=1
    -D MONITOR_SPEED=${this.monitor_speed}
    -Wall
    -Wextra
    -fdiagnostics-color=always
    -Wno-error=class-memaccess
;
[esp32s3]
    board_build.partitions = boards/miniwebradio4MB.csv    ;  4MB Flash
    ;board_build.partitions = boards/miniwebradio8MB.csv    ;  8MB Flash
    ;board_build.partitions = boards/miniwebradio16MB.csv   ; 16MB Flash
    ;board_build.partitions = boards/miniwebradio32MB.csv   ; 32MB Flash

    board = ESP32-S3-DevKitC-1-N4R8    ;  4 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N8R8    ;  8 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N16R8   ; 16 MB quad Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N16R8V  ;  8 MB octo Flash, 8MB octo PSRAM
    ;board = ESP32-S3-DevKitC-1-N32R8V  ;  8 MB octo Flash, 8MB octo PSRAM
;
```

8) Enter your access data in **common.h** and select the parameters according to the HW used.
If you use a SPI display (TFT_Controller <7), further settings such as ROTATION or TP_VERSION may be required



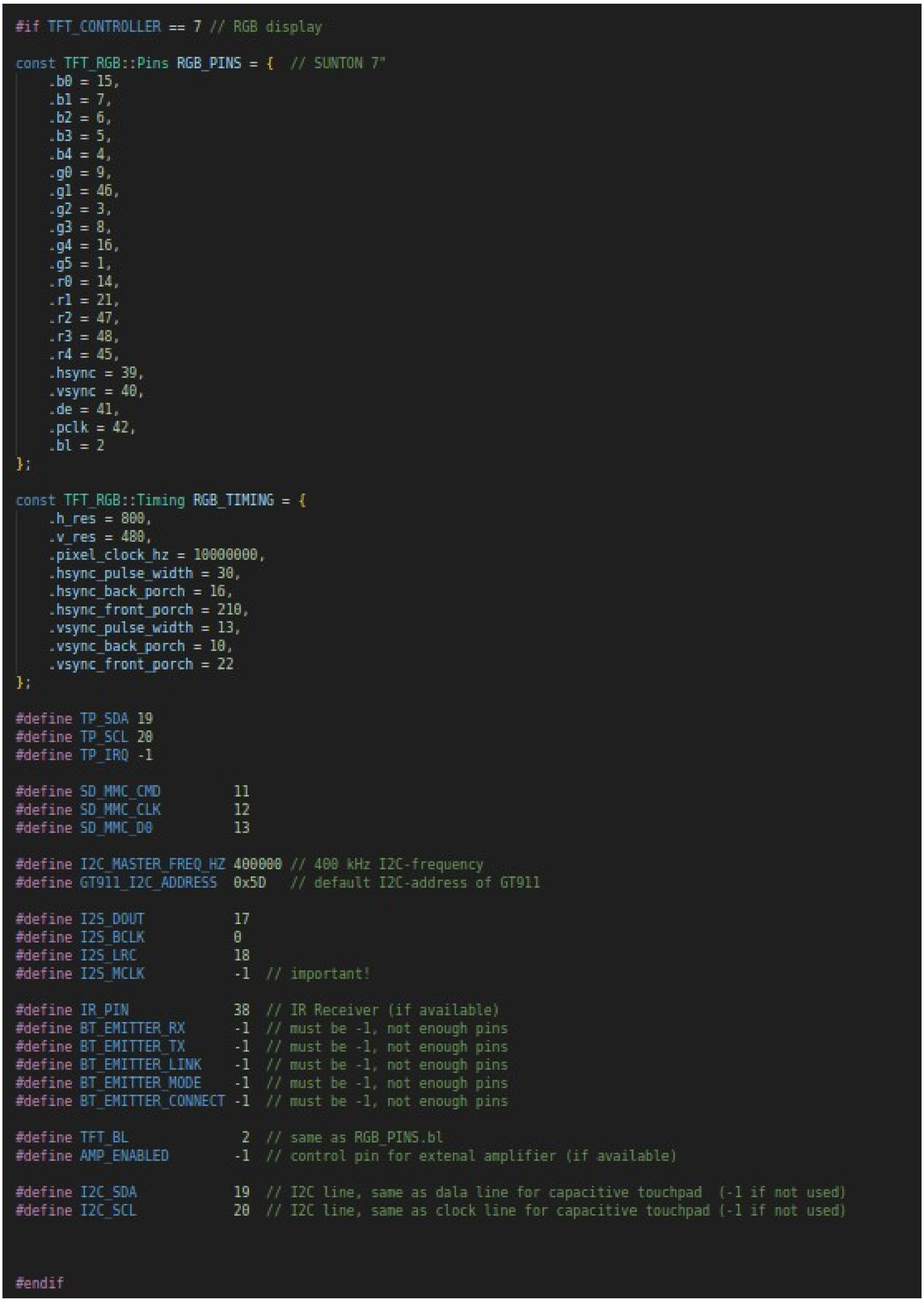
```
#pragma once
#pragma GCC optimize("Os") // optimize for code size
// clang-format off
#define _SSID                   "mySSID"                        // Your WiFi credentials
#define _PW                     "myWiFiPassword"                // Or in textfile on SD-c
#define TFT_CONTROLLER          7                               // (0)ILI9341, (3)ILI9486
#define DISPLAY_INVERSION       0                               // only SPI displays  (0)
#define TFT_ROTATION            1                               // only SPI displays  1 o
#define TFT_FREQUENCY           40000000                        // only SPI displays  800
#define TP_VERSION              5                               // only SPI displays  (0)
#define TP_ROTATION             1                               // only SPI displays  1 o
#define TP_H_MIRROR             0                               // only SPI displays  (0)
#define TP_V_MIRROR             0                               // only SPI displays  (0)
#define I2S_COMM_FMT            0                               // (0) MAX98357A PCM5102A
#define SDMMC_FREQUENCY         80000000                        // 80000000 or 40000000 H
#define FTP_USERNAME            "esp32"                         // user and pw in FTP Cli
#define FTP_PASSWORD            "esp32"
#define CONN_TIMEOUT            2500                            // unencrypted connection
#define CONN_TIMEOUT_SSL        3500                            // encrypted connection t
#define WIFI_TX_POWER           5                               // 2 ... 21 (dBm) Adjust
//────────────────────────────────────────────────────────────────────────────────────────
```

Further below you will find the assignment of the ESP32-S3 pins that you can change if



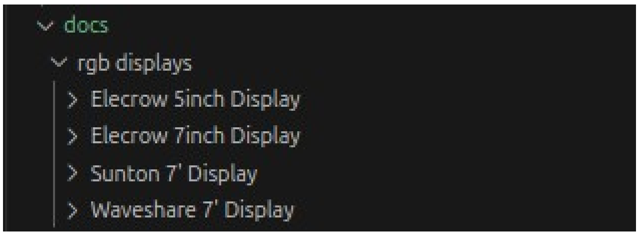
```
#if TFT_CONTROLLER < 7
    // Digital I/O used
        #define TFT_CS               8
        #define TFT_DC              12
        #define TFT_BL              10 // at -1 the brightness menu is not displayed
        #define TP_IRQ              39
        #define TP_CS               15
        #define SD_MMC_D0           11
        #define SD_MMC_CLK          13
        #define SD_MMC_CMD          14
        #define IR_PIN               4  // IR Receiver (if available)
        #define TFT_MOSI            18  // TFT and TP (FSPI)
        #define TFT_MISO             2  // TFT and TP (FSPI)
        #define TFT_SCK             17  // TFT and TP (FSPI)

        #define I2S_DOUT             9
        #define I2S_BCLK             3
        #define I2S_LRC              1
        #define I2S_MCLK             0

        #define AMP_ENABLED         -1  // control pin for extenal amplifier (if available)
        #define BT_EMITTER_RX       45  // TX pin - KCX Bluetooth Transmitter    (-1 if not available)
        #define BT_EMITTER_TX       38  // RX pin - KCX Bluetooth Transmitter    (-1 if not available)
        #define BT_EMITTER_LINK     19  // high if connected                     (-1 if not available)
        #define BT_EMITTER_MODE     20  // high transmit - low receive           (-1 if not available)
        #define BT_EMITTER_CONNECT 48  // high impulse -> awake after POWER_OFF (-1 if not available)

        #define I2C_SDA             41  // I2C, dala line for capacitive touchpad
        #define I2C_SCL             42  // I2C, clock line for capacitive touchpad
#endif
```

These settings apply to RGB-HMI displays (TFT_Controller == 7)



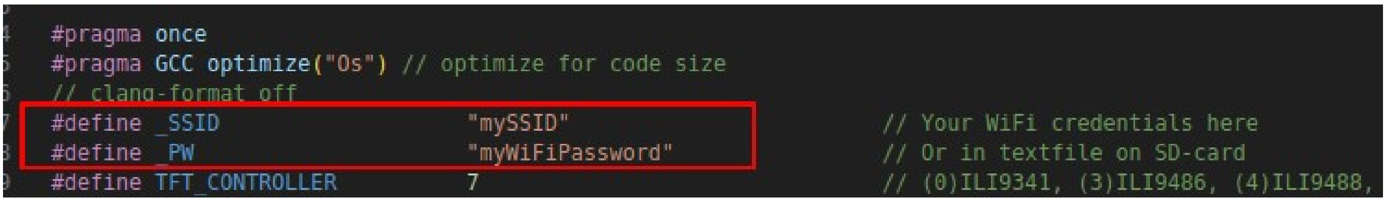
```
#if TFT_CONTROLLER == 7 // RGB display

const TFT_RGB::Pins RGB_PINS = {  // SUNTON 7"
    .b0 = 15,
    .b1 = 7,
    .b2 = 6,
    .b3 = 5,
    .b4 = 4,
    .g0 = 9,
    .g1 = 46,
    .g2 = 3,
    .g3 = 8,
    .g4 = 16,
    .g5 = 1,
    .r0 = 14,
    .r1 = 21,
    .r2 = 47,
    .r3 = 48,
    .r4 = 45,
    .hsync = 39,
    .vsync = 40,
    .de = 41,
    .pclk = 42,
    .bl = 2
};

const TFT_RGB::Timing RGB_TIMING = {
    .h_res = 800,
    .v_res = 480,
    .pixel_clock_hz = 10000000,
    .hsync_pulse_width = 30,
    .hsync_back_porch = 16,
    .hsync_front_porch = 210,
    .vsync_pulse_width = 13,
    .vsync_back_porch = 10,
    .vsync_front_porch = 22
};

#define TP_SDA 19
#define TP_SCL 20
#define TP_IRQ -1

#define SD_MMC_CMD         11
#define SD_MMC_CLK         12
#define SD_MMC_D0          13

#define I2C_MASTER_FREQ_HZ 400000 // 400 kHz I2C-frequency
#define GT911_I2C_ADDRESS  0x5D   // default I2C-address of GT911

#define I2S_DOUT           17
#define I2S_BCLK           0
#define I2S_LRC            18
#define I2S_MCLK           -1  // important!

#define IR_PIN             38  // IR Receiver (if available)
#define BT_EMITTER_RX      -1  // must be -1, not enough pins
#define BT_EMITTER_TX      -1  // must be -1, not enough pins
#define BT_EMITTER_LINK    -1  // must be -1, not enough pins
#define BT_EMITTER_MODE    -1  // must be -1, not enough pins
#define BT_EMITTER_CONNECT -1  // must be -1, not enough pins

#define TFT_BL              2  // same as RGB_PINS.bl
#define AMP_ENABLED        -1  // control pin for extenal amplifier (if available)

#define I2C_SDA            19  // I2C line, same as dala line for capacitive touchpad  (-1 if not used)
#define I2C_SCL            20  // I2C line, same as clock line for capacitive touchpad (-1 if not used)



#endif
```

For various RGB display you will find templates in the docs



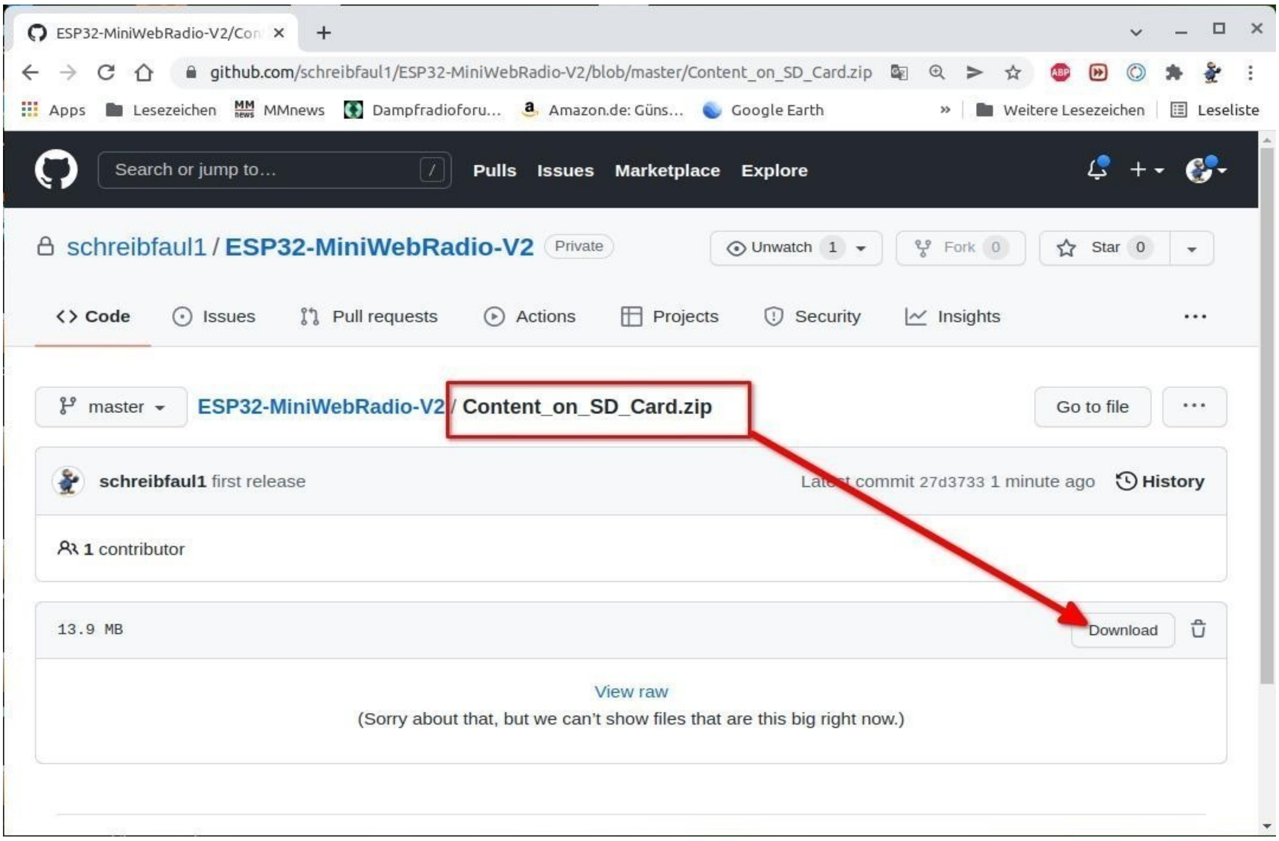
```
⌄ docs
  ⌄ rgb displays
    > Elecrow 5inch Display
    > Elecrow 7inch Display
    > Sunton 7' Display
    > Waveshare 7' Display
```

For the first start you can enter the WiFi access data in common.h. Alternatively, the SSID can be selected on the display and the password is entered.



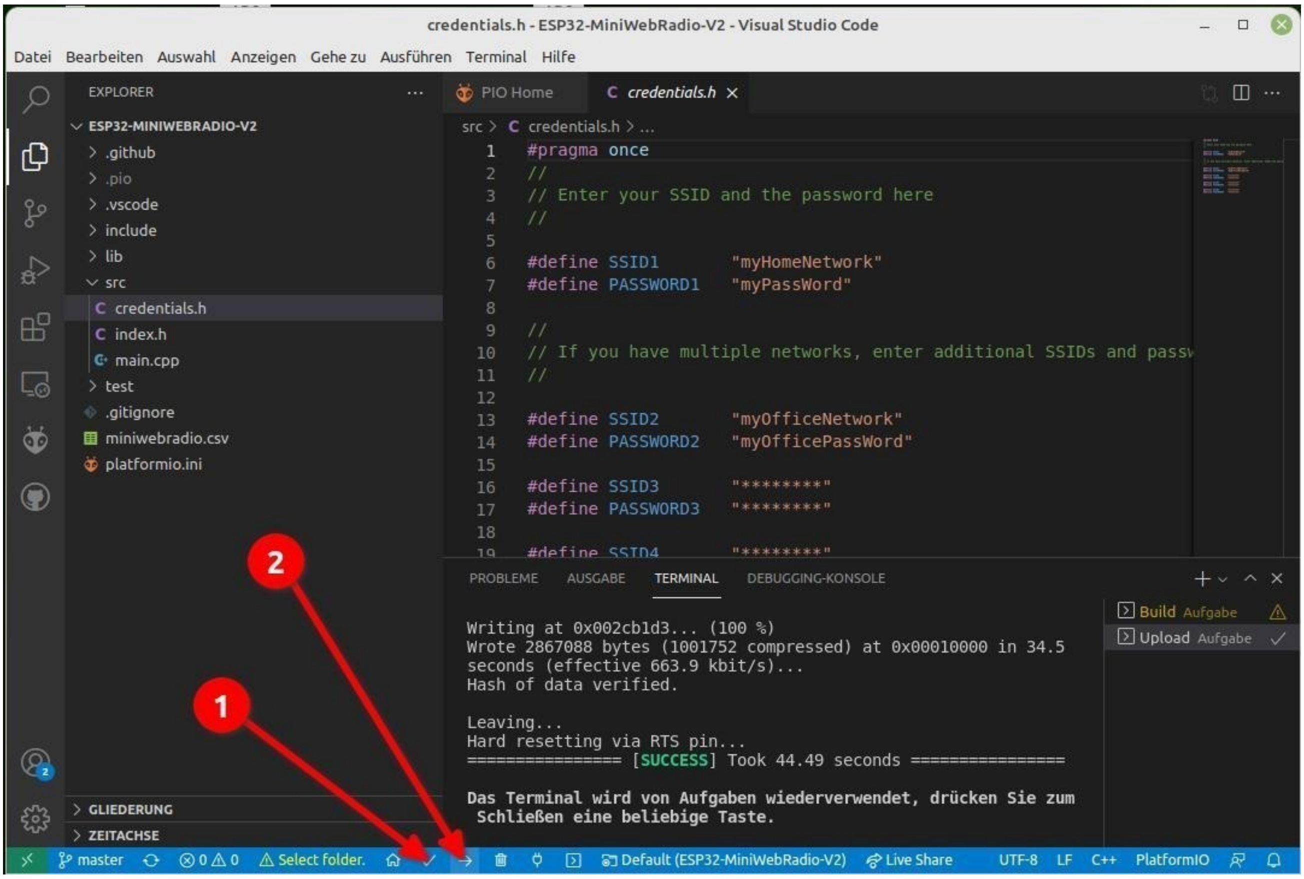
```
#pragma once
#pragma GCC optimize("Os") // optimize for code size
// clang-format off
#define _SSID                    "mySSID"                  // Your WiFi credentials here
#define _PW                      "myWiFiPassword"          // Or in textfile on SD-card
#define TFT_CONTROLLER           7                         // (0)ILI9341, (3)ILI9486, (4)ILI9488,
```

9) back to Github download the Content_On_SD_Card.zip file and extract to SD



(Sorry about that, but we can't show files that are this big right now.)

10) Connect the ESP32 to USB, press build and then upload, Thats all



```
#pragma once
//
// Enter your SSID and the password here
//

#define SSID1        "myHomeNetwork"
#define PASSWORD1    "myPassWord"

//
// If you have multiple networks, enter additional SSIDs and passw
//

#define SSID2        "myOfficeNetwork"
#define PASSWORD2    "myOfficePassWord"

#define SSID3        "********"
#define PASSWORD3    "********"

#define SSID4        "********"
```

```
Writing at 0x002cb1d3... (100 %)
Wrote 2867088 bytes (1001752 compressed) at 0x00010000 in 34.5
seconds (effective 663.9 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
================ [SUCCESS] Took 44.49 seconds ================

Das Terminal wird von Aufgaben wiederverwendet, drücken Sie zum
 Schließen eine beliebige Taste.
```