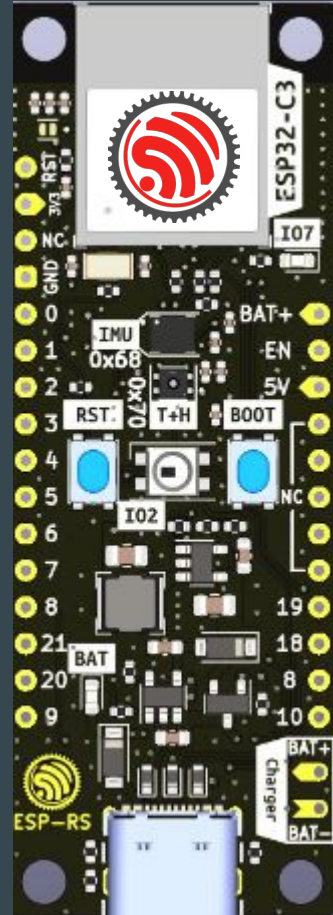


14 tips for Makers

2024-05-17

Espressif Community event
Impact Hub - Brno

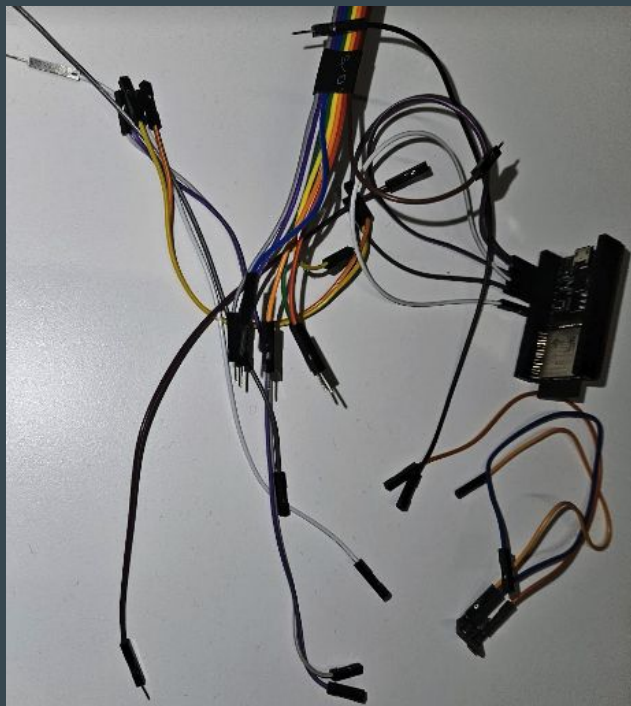
[Juraj Michálek](#) - [Espressif Systems](#)



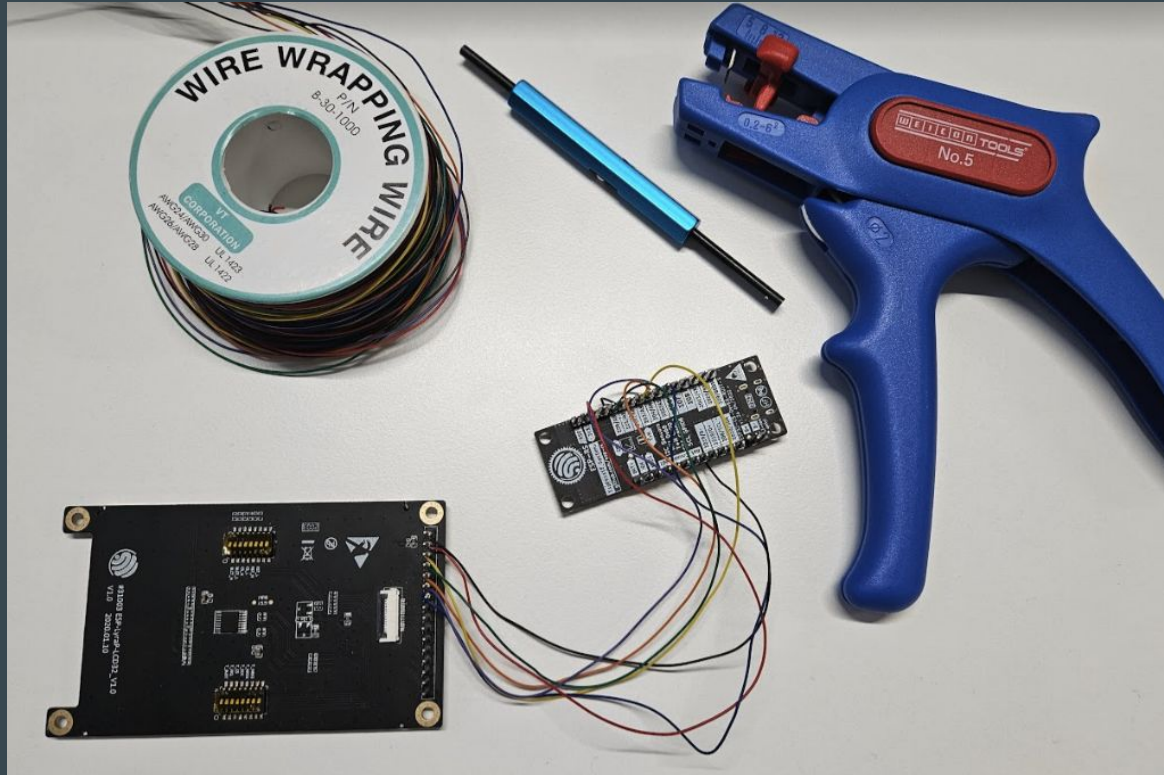
Tip #1

Connecting pins

Duponts?



Wire wrap



Wire wrap

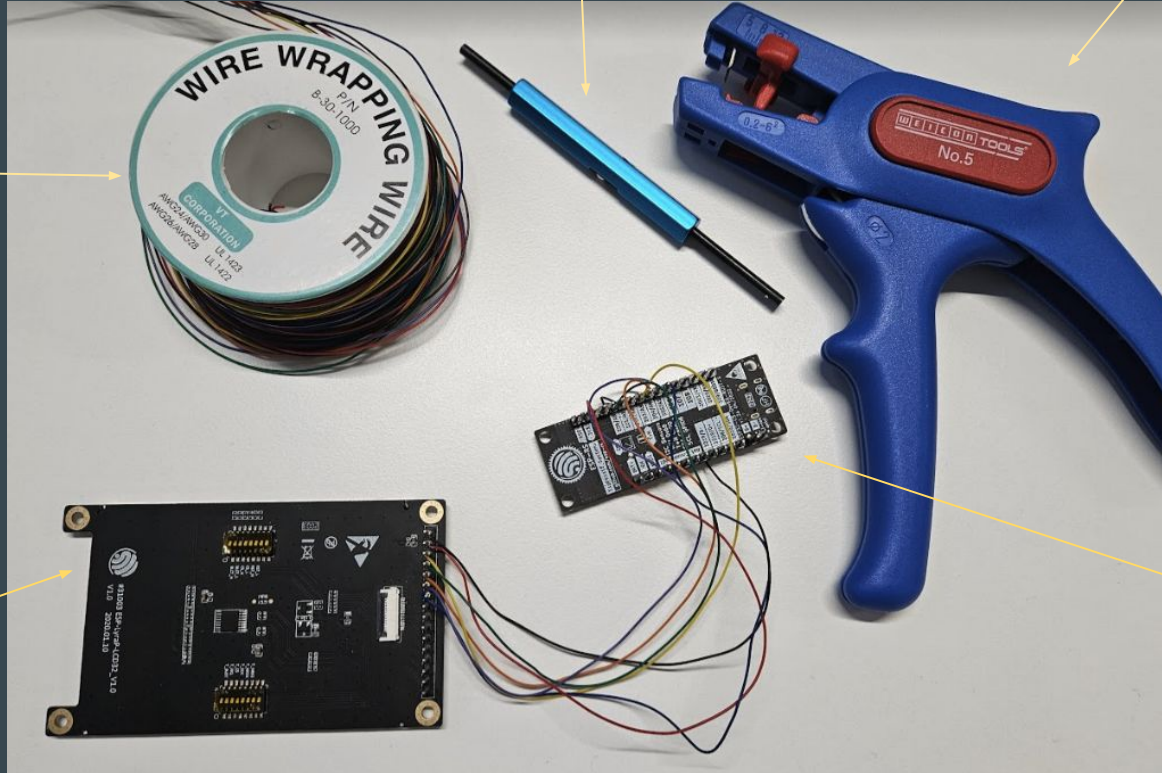
Wire Wrap Tool

Weicon No.5.

30AWG Wire Wrapping

LCD ILI9341

ESP32-C3
Rust Board

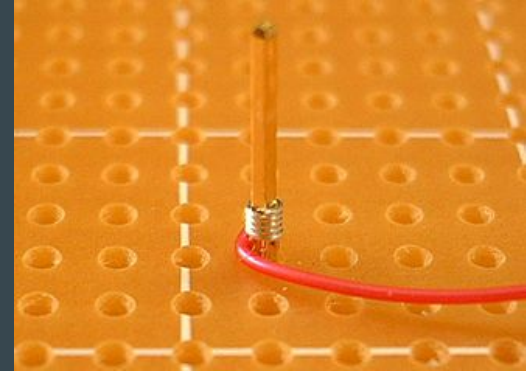


Wire wrap

Quick, safe way how to connect pins and wires

Wire Wrapping for our Projects by Andreas Spiess:

<https://youtu.be/L-463vchW0o?si=U96aLAWlsLpsPGTW>



picture from Wikipedia: https://en.wikipedia.org/wiki/Wire_wrap

Tip #2

Soldering

Soldering iron?



Clean oxidized tip

Thin oxide layer is blocking heat transfer.



Steel Wire
Dish Pot
Cleaning
Scourer

Pinecil



PINE64

Tip #3

Start playing in browser

wokwi.com/esp32

Contribute: <https://github.com/wokwi>

EDC22 Day 1 Talk 9: Your browser is ESP32
- Wokwi - <https://youtu.be/TKe4MgD6O8o>

Rust project examples

+ NEW PROJECT

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32-blink.rs

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32s2-blink.rs

```
#![no_std]
#![no_main]

use esp32c3::hal::ledc;
use esp32c3::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32c3-blink.rs

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32-nostd-blink.rs

```
#![no_std]
#![no_main]

use esp32s2::hal::ledc;
use esp32s2::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32s2-nostd-blink.rs

```
#![no_std]
#![no_main]

use esp32c3::hal::ledc;
use esp32c3::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp32c3-nostd-blink.rs

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp-clock (WiFi)

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

Crispy Click

```
#![no_std]
#![no_main]

use esp32c3::hal::ledc;
use esp32c3::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

ESP32C3 + ILI9341 Display

```
#![no_std]
#![no_main]

use esp32s3::hal::ledc;
use esp32s3::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

ESP32S3 + ILI9341 Display

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

ESP32 + ILI9341 Display

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

ESP32 + 8x8 LED Dot Matrix

```
#![no_std]
#![no_main]

use esp32s2::hal::ledc;
use esp32s2::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

ESP32S2 + Keypad

```
#![no_std]
#![no_main]

use esp32::hal::ledc;
use esp32::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

Joystick Etch-a-Sketch

```
#![no_std]
#![no_main]

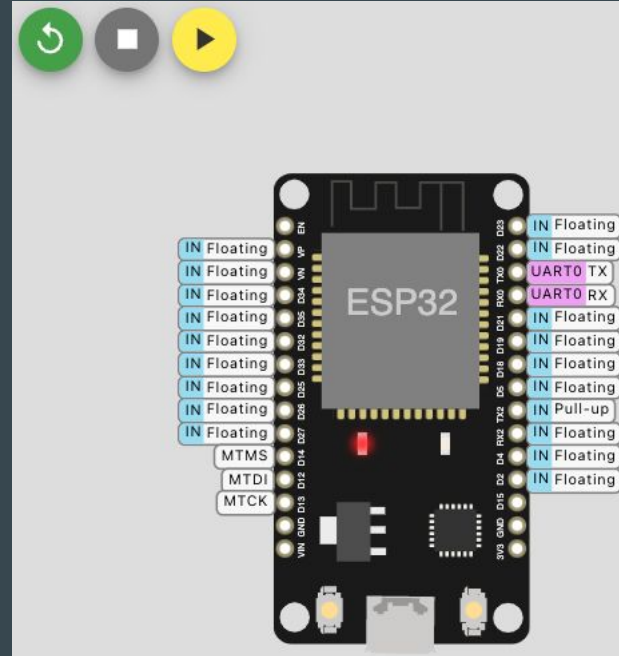
use esp32c3::hal::ledc;
use esp32c3::hal::timer;

fn main() {
    let mut ledc = ledc::Leds::new();
    let mut timer = timer::TimerGroup::new();

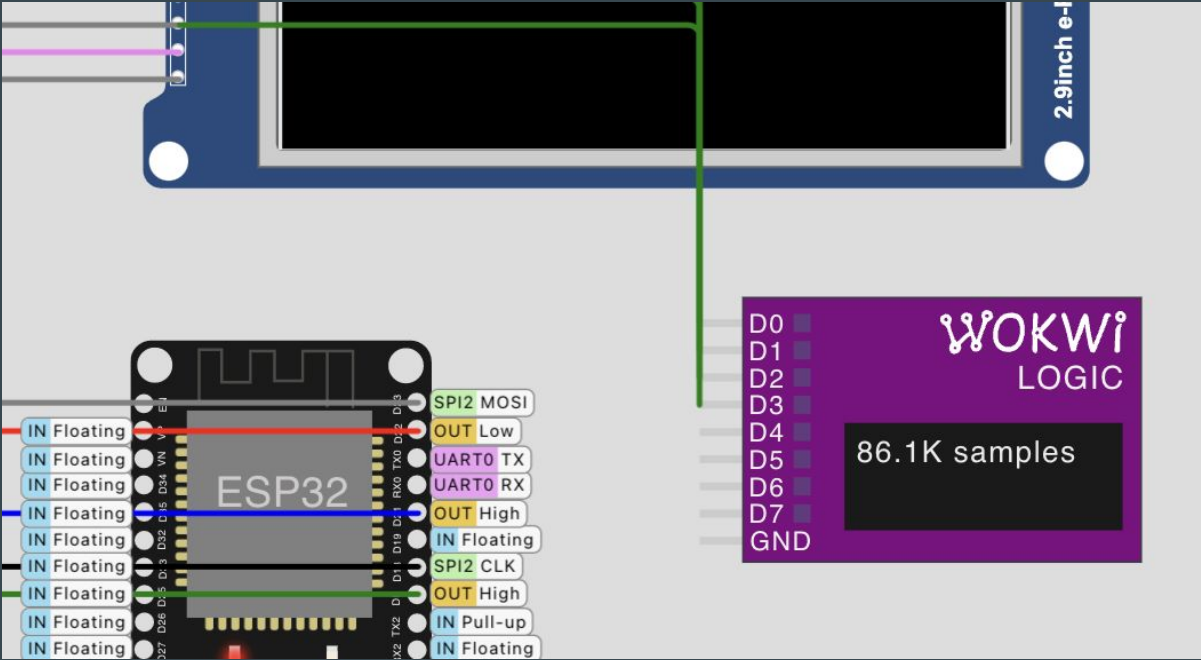
    ledc.set_pwm(0, 100, 500);
    timer.start_timer(0, 1000);
}
```

esp-gallery

Pause button - see state of GPIOs

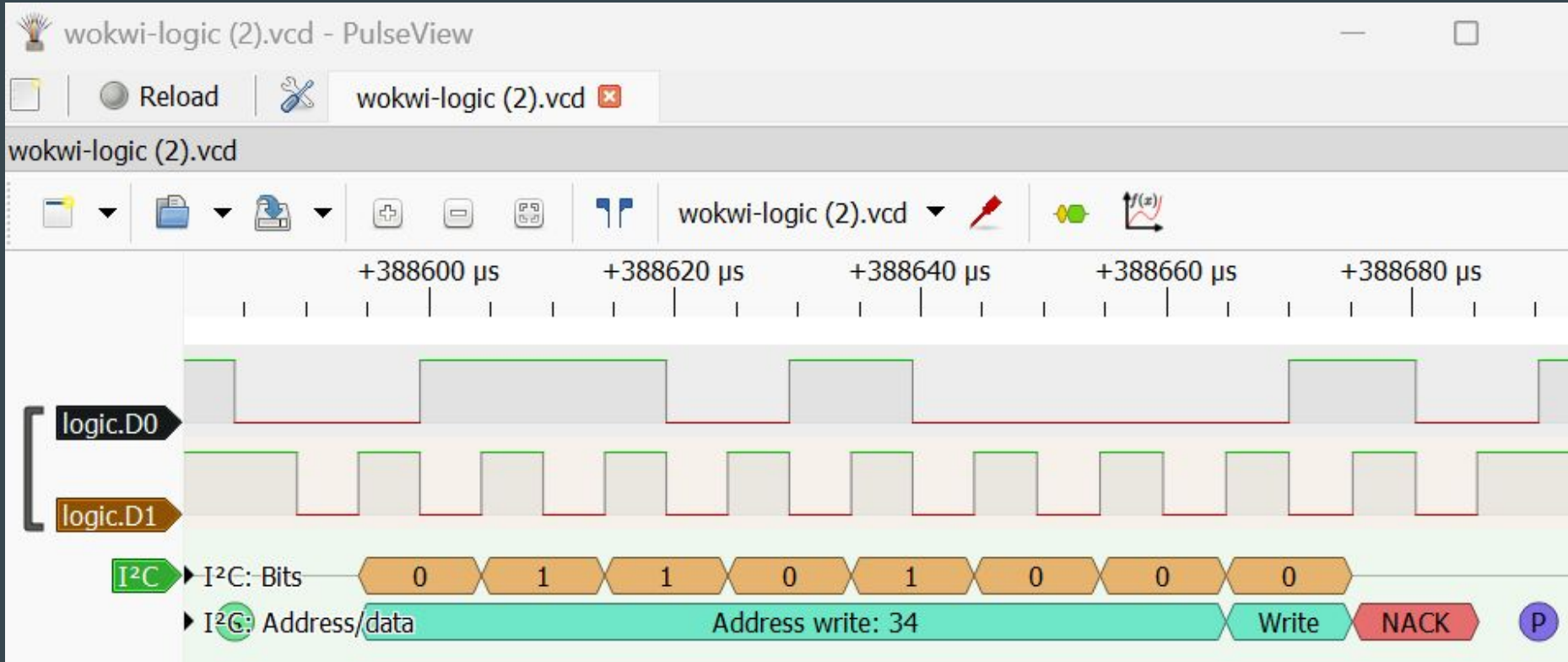


Wokwi Logic Analyzer



Hit Stop to download VCD file for PulseView

Wokwi + PulseView - examples of I2C



Create Custom chips like e-Paper

Simulate your own HW

Wokwi: Getting Started with the Wokwi Custom Chips C API

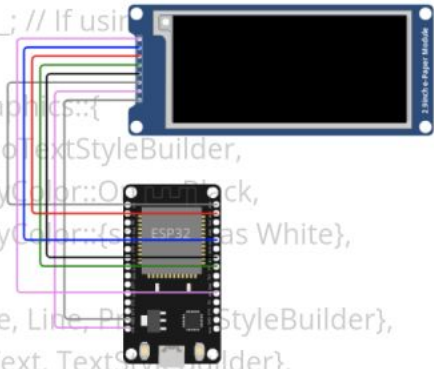
<https://docs.wokwi.com/chips-api/getting-started>

<https://wokwi.com/projects/366167936725307393>

```
main.rs
```

```
use esp_idf_sys as _; // If using the `esp-idf` toolchain, this is required to build the project.

use embedded_graphics::{
    mono_font::MonoTextStyleBuilder,
    pixelcolor::BinaryColor::On as Black,
    pixelcolor::BinaryColor::Off as White,
    prelude::*,
    primitives::{Circle, Line, Point, Rectangle, StyleBuilder},
    text::{Baseline, Text, TextStyleBuilder},
};
```



The diagram illustrates the hardware connection between an ESP32 microcontroller and an e-Paper display module. The ESP32 is a small black board with a blue chip. The e-Paper display is a larger blue board with a black screen. Colored lines represent the connections between the two boards.

WOKWI

UIFlow2.M5Stack.com

UIFlow2 V2.0.5

main Save

System

- Time
- BLE UART
- BLE
- WLAN STA
- WLAN AP
- NVS
- Power
- UI

Setup

Begin initialization

Init built-in hardware

Loop

Update button

Set circle0

Update

Read Mode

```
23 circle3 = Widgets.Circle(13, 10, 10)
24 circle4 = Widgets.Circle(17, 10, 10)
25 circle5 = Widgets.Circle(21, 10, 10)
26
27
28
29 def loop():
30     global circle0, circle1, circle2, circle3, circle4, circle5
31     M5.update()
32     circle0.setVisible(True)
33
34
35 if __name__ == '__main__':
36     try:
37         setup()
38         while True:
39             loop()
40     except (Exception, KeyboardInterrupt):
41         try:
```

EZ DATA

CoreS3: m5stack

Run

Tip #4

Explore DevKits

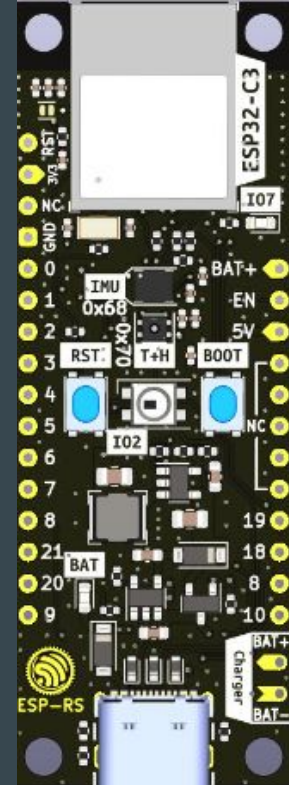
Designing Open Hardware - esp-rust-board

KiCad templates

<https://github.com/esp-rs/esp-rust-board>

ESP32-C3-DevKit-RUST-1

<https://www.espressif.com/en/products/devkits>



Open Hardware - ESP32-S3-BOX-3

<https://github.com/espressif/esp-box>

Joystick controller

https://github.com/espressif/esp-box/tree/master/examples/esp_joystick/joystick_controller



M5Stack

M5 DIAL

Programmable Controller - Smart Knob

Integrated rich peripherals

Rotary encoder

Convenient deployment

Low power design



Card-size Computer

CARDPUTER

- ESP32-S3
- 1.14" TFT screen
- 56-key keyboard
- 120mAh + 1400mAh lithium battery



m5stack.com

CoreS3 Development Kit



ESP32
S3



Multimedia
Applications



TinyML



Smart Home
Control



NANO_D++

<https://store.binaris.io/>

NANO_D++

Open Source & Open Hardware
Haptic Human Machine Interface Device



EVALUATION KIT

Coming Soon

Tip #5

Learn IDEs, have no fear from CLI

VS Code + Espressif Extension + Wokwi

```
main.rs — hello-wokwi
src > main.rs > main
45 let timer_group1: TimerGroup<TIMG1> = TimerGroup::new(
46     timer_group: peripherals.TIMG1,
47     &clocks,
48     &mut system.peripheral_clock_control,
49 );
50 let mut wdt1: Wdt<TIMG1> = timer_group1.wdt;
51 rtc.rwdt.disable();
52 wdt0.disable();
53 wdt1.disable();
54 println!("Hello world!");
55
56 loop {}
57 } fn main
58
```

VARIABLES

- Locals
 - timer_group1: {...}
 - timer_group0: {...}
 - rtc: {...}
 - clocks: {...}
- Registers
 - CPU
 - pc: 0x400d2488
 - ar0: 0x800d9bf9
 - ar1: 0x3ffdbd90
 - ar2: 0x1
- WATCH
- CALL STACK
 - [1] PAUSED ON BREAKPOINT
 - hello_wokwi::__xtensa_lx_rt_m...
 - main() main.rs 30:1
 - [2] PAUSED
- BREAKPOINTS
 - main.rs src 54
- ESP PERIPHERAL VIEW

DEBUG CONSOLE

```
>4 println!("Hello world!");
Execute debugger commands using "-exec <command>", for example "-exec i
nfo registers" will list registers in use (when GDB is the debugger)
```

main* Wokwi GDB (hello-wokwi) rust-analyzer Spaces: 4 UTF-8 LF Rust

Wokwi Simulator

WOKWI Wokwi for VS Code

Licensed to: Juraj Michálek

00:48.200 99%

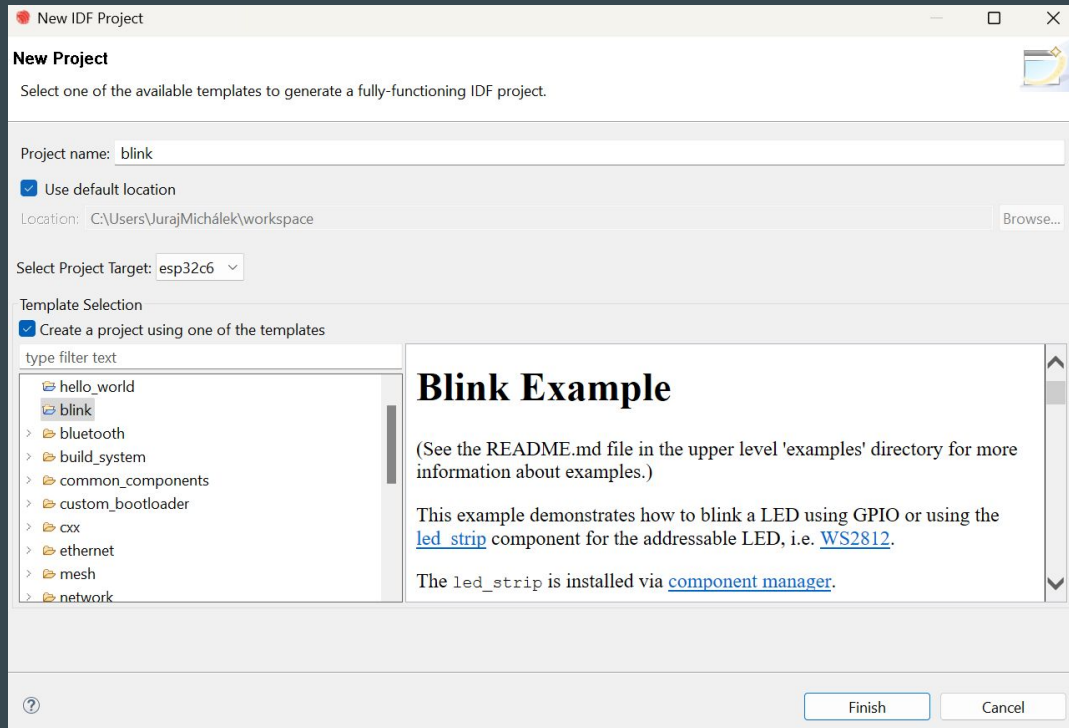
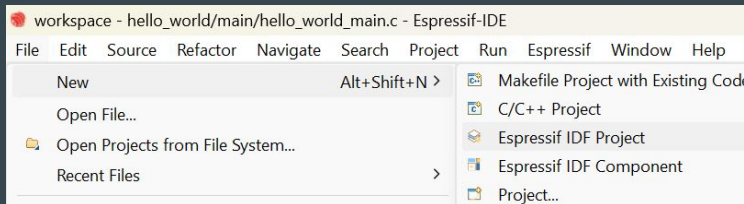
BH1750 LUX 127

Expected: 5c2fd9d3ac32c2f0e227f5f7db40a3ff5a25e28d996830f479abd09b10d3a2d3
Attempting to boot anyway...
entry 0x4086c610
W (10) clk: esp_perip_clk_init() has not been implemented yet
Not an LP core wakeup. Cause = 0
Initializing...
LP I2C initialized successfully
LP core loaded with firmware successfully
Entering deep sleep...

PROBLEMS OUTPUT AZURE: ACTIVITY LOG TERMINAL DEBUG CONSOLE

[chip-bh1750] Hello from light sensor!

Espressif IDE (based on Eclipse)



JetBrains - CLion + Wokwi simulator



The screenshot displays the CLion IDE interface with the following components:

- Code Editor:** Shows the `simple_test_main.c` file with the following code:

```
7 > #include ...
14
15 void app_main(void)
16 {
17     printf("Hello world!!\n");
18
19     /* Print chip information */
20     esp_chip_info_t chip_info; chip_info: esp_chip_info_t
21     uint32_t flash_size; flash_size: 0
22     esp_chip_info(outInfo: &chip_info); chip_info: esp_chip_info_t
23     printf("This is %s chip with %d CPU core(s), %s%s%s",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
27           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
28           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
29           app_main
```
- Wokwi Simulator:** A window titled "Wokwi Simulator" showing a virtual ESP32S3 board. It includes a "WOKwi" logo, "Wokwi for VS Code", and "Jozott Community License". The simulator shows a timer at 00:00.367 and 0% battery. The board components are listed on the right, including UART TX, UART RX, SPI, I2C, and various pins.
- Debug Console:** Shows the execution of the program. The output is:

```
Thread-1 (1) Evaluate expression (⇐) or add a watch (⌘W)
app_main simple_test_main.c:23
main_task app_startup.c:208
vPortTaskWrapper port.c:162
chip_info = (esp_chip_info_t)
  model = (esp_chip_model_t) CHIP_ESP32S3
  features = (uint32_t) 18
  revision = (uint16_t) 0
  cores = (uint8_t) 2 \002
  flash_size = (uint32_t) 0
```

Jozott00's plugin

idf.py --help

```
ESP-IDF 5.3
PS C:\Espressif\frameworks\esp-idf-master\examples\get-started\hello_world> idf.py --help
Usage: idf.py [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

ESP-IDF CLI build management tool. For commands that are not known to idf.py an attempt to execute it as a build
system target will be made. Selected target: esp32p4

Options:
  --version                Show IDF version and exit.
  --list-targets          Print list of supported targets and exit.
  -C, --project-dir PATH  Project directory.
  -B, --build-dir PATH    Build directory.
  -w, --cmake-warn-uninitialized / -n, --no-warnings
                          Enable CMake uninitialized variable warnings for CMake files inside the project
                          directory. (--no-warnings is now the default, and doesn't need to be specified.)
                          The default value can be set with the IDF_CMAKE_WARN_UNINITIALIZED environment
                          variable.
  -v, --verbose           Verbose build output.
  --preview              Enable IDF features that are still in preview.
  --ccache / --no-ccache Use ccache in build. Disabled by default. The default value can be set with the
                          IDF_CCACHE_ENABLE environment variable.
  -G, --generator [Ninja]
                          CMake generator.
  --no-hints              Disable hints on how to resolve errors and logging.
  -D, --define-cache-entry TEXT
                          Create a cmake cache entry. This option can be used at most once either
                          globally, or for one subcommand.
  -p, --port PATH        Serial port. The default value can be set with the ESPPORT environment variable.
                          This option can be used at most once either globally, or for one subcommand.
  -b, --baud INTEGER     Baud rate for flashing. It can imply monitor baud rate as well if it hasn't been
                          defined locally. The default value can be set with the ESPBAUD environment
                          variable. This option can be used at most once either globally, or for one
                          subcommand.
```

idf.py build flash monitor

idf.py uf2

Tip #6

Have more USB cables

Tip #7

Know the limits

Desktop vs. Embedded development

Memory: **GB** (TB) vs. **kB** (MB bytes of PSRAM)

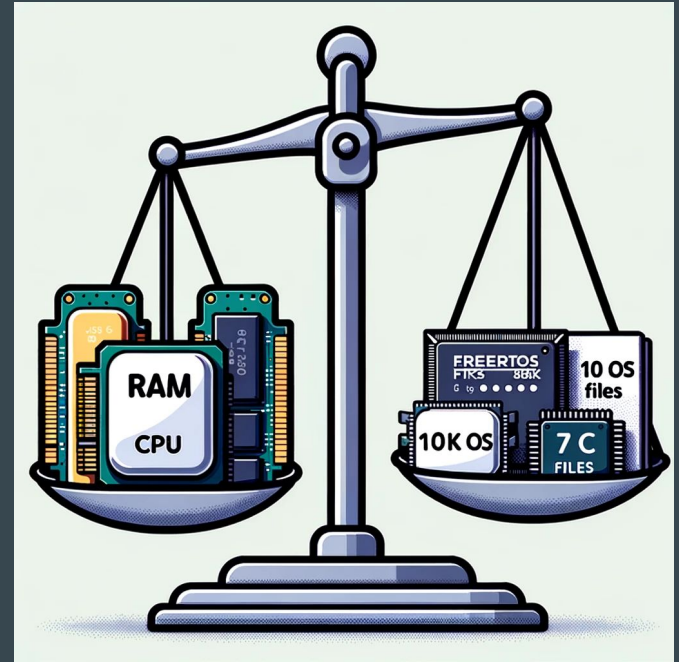
CPU Cores: **8 - 64** vs. **1 - 2**

CPU Frequencies: GHz vs. 20-240 MHz

Power consumption: 100s of Watts vs. Milliwatts

OS kernel: **61k** files vs. 10k files in ESP-IDF

(FreeRTOS kernel: **7** C files)



ESP32 Lang lab - Free heap - no PSRAM

<https://github.com/georgik/esp32-lang-lab>

	ESP-IDF C	Arduino	CircuitPython	MicroPython	Rust no_std	Rust std	ZIG + ESP-IDF
ESP32	300892	237568	113424	164064	179200	296028	300476
ESP32-S3	386744	369920	150432		332800	388016	389976
ESP32-C6	468852				440316	471068	471208

Note: ESP32, ESP32-S3 - supports PSRAM, ESP32-C6 does not support PSRAM

Tip #8

Choose right HW/SW setup

Chip

ESP32 



Available: 2016 - 2028 

- + ETH
- ←
- USB
- CAM
- SPI-ETH

ESP32-S3 



Available: 2022 - 2034 

- + WiFi6
- + Low-Power
- + RISC-V
-
- USB
- LCD
- CAM
- PSRAM
- SD/MMC
- Xtensa

ESP32-C6 



Available: 2023 - 2035 

ESP32-C6

Main feature: WiFi 6 support - reduced power consumption

- <https://www.youtube.com/watch?v=FA1jqZLig4s>

Second important feature:

- Low Power Core - 20 MHz



Form Factor

DevKits



ESP32-S3-BOX-3



ESP32-S3-DevKitM-1



ESP32-S3-DevKitC-1

Modules



ESP32-S3-WROOM-1



ESP32-S3-MINI-1

Chips (Soc)



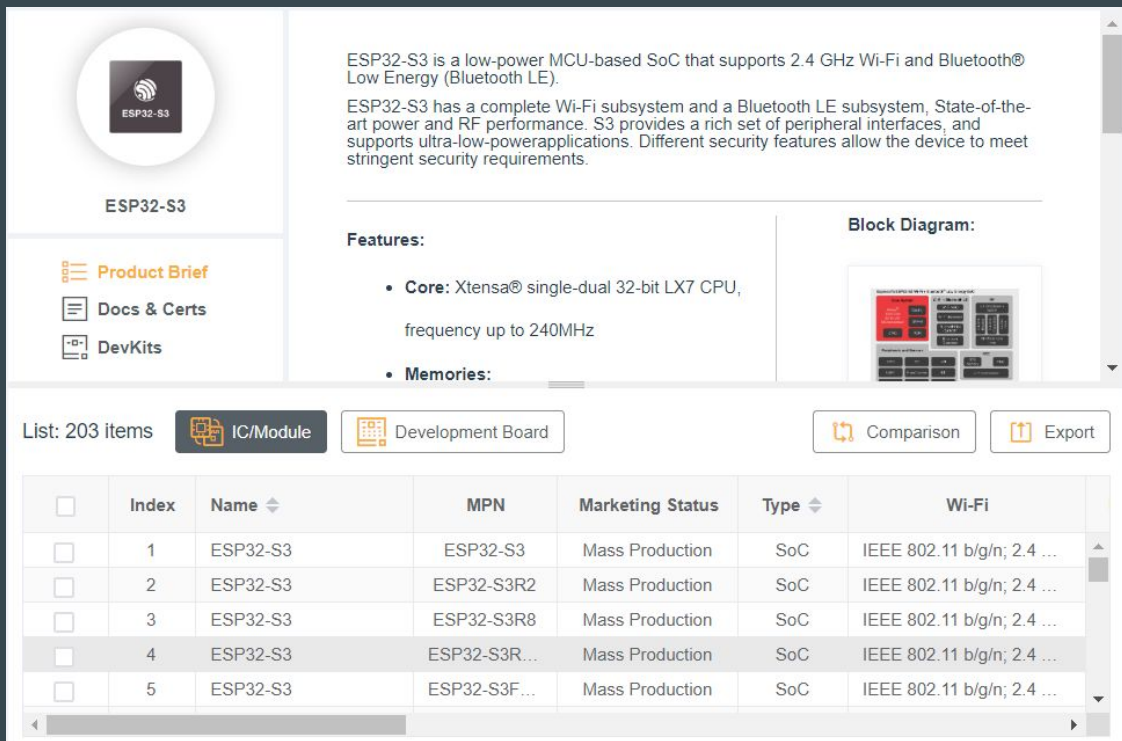
ESP32-S3



ESP32-S3-PICO-1

Many chips, many boards - quick help

<https://products.espressif.com/>



The screenshot shows the product page for the ESP32-S3. On the left, there is a navigation menu with options: Product Brief, Docs & Certs, and DevKits. The main content area includes a description of the chip as a low-power MCU-based SoC, a list of features (Core: Xtensa® single-dual 32-bit LX7 CPU, frequency up to 240MHz; Memories), and a block diagram. Below the main content, there are filters for 'IC/Module' and 'Development Board', and buttons for 'Comparison' and 'Export'. At the bottom, a table lists five different ESP32-S3 models with their respective MPN, Marketing Status, Type, and Wi-Fi capabilities.

ESP32-S3

ESP32-S3 is a low-power MCU-based SoC that supports 2.4 GHz Wi-Fi and Bluetooth® Low Energy (Bluetooth LE).

ESP32-S3 has a complete Wi-Fi subsystem and a Bluetooth LE subsystem, State-of-the-art power and RF performance. S3 provides a rich set of peripheral interfaces, and supports ultra-low-power applications. Different security features allow the device to meet stringent security requirements.

Features:

- **Core:** Xtensa® single-dual 32-bit LX7 CPU, frequency up to 240MHz
- **Memories:**

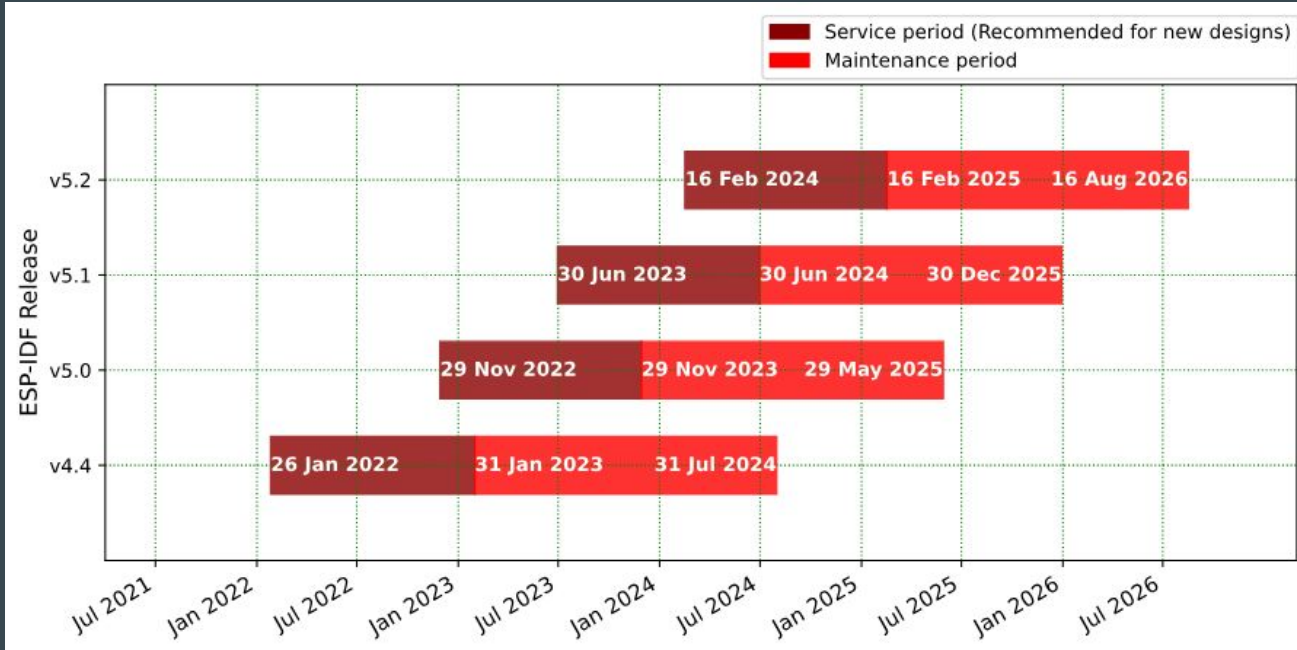
Block Diagram:

List: 203 items

IC/Module Development Board Comparison Export

<input type="checkbox"/>	Index	Name	MPN	Marketing Status	Type	Wi-Fi
<input type="checkbox"/>	1	ESP32-S3	ESP32-S3	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	2	ESP32-S3	ESP32-S3R2	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	3	ESP32-S3	ESP32-S3R8	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	4	ESP32-S3	ESP32-S3R...	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	5	ESP32-S3	ESP32-S3F...	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...

ESP-IDF Release Support Schedule



ESP-IDF Release and SoC Compatibility

Chip	v4.4	v5.0	v5.1	v5.2	v5.3
ESP32	supported	supported	supported	supported	supported
ESP32-S2	supported	supported	supported	supported	supported
ESP32-C3	supported	supported	supported	supported	supported
ESP32-S3	supported	supported	supported	supported	supported
ESP32-C2		supported	supported	supported	supported
ESP32-C6			supported	supported	supported
ESP32-H2			supported	supported	supported
ESP32-P4					supported
ESP32-C5					preview

Tip #9

GUI Designers

Embedded Wizard



GUI Solutions by TARA Systems

PROJECT EDIT ARRANGE SEARCH NAVIGATE BUILD DEBUG EXTRAS WINDOW HELP

ESP32 Default (none)

Templates Browser

Search ...

Chora

- Unit
Project module
- Class
Empty software component
- Constant
 - Component Templates
 - Resources
 - Views
 - Widgets
 - Effects
 - Event Handlers
 - Device

Mosaic Framework

- unit Core
- unit Resources
- unit Graphics
- unit Effects
- unit Views

Profile Configuration

- profile ESP32

ESP32

Name	Type	Order
ESP32	profile	15
Note4	note group	14
Note	note legend	13
Note5	note group	12
Note3	note group	11

Brick

<290,70,490,110>

Description
PlatformPackage
Application:App...

Application:App...

Prototype Application:Ap...

Level: 01, Score: 00005, Lost: 00

Information
Loading project 'BrickGame.ewp' ...
Success
Done.
Information
Prototype the class 'Application:Application' ...
Information
Prototype the class 'Application:Application' ...

Main Search Debug

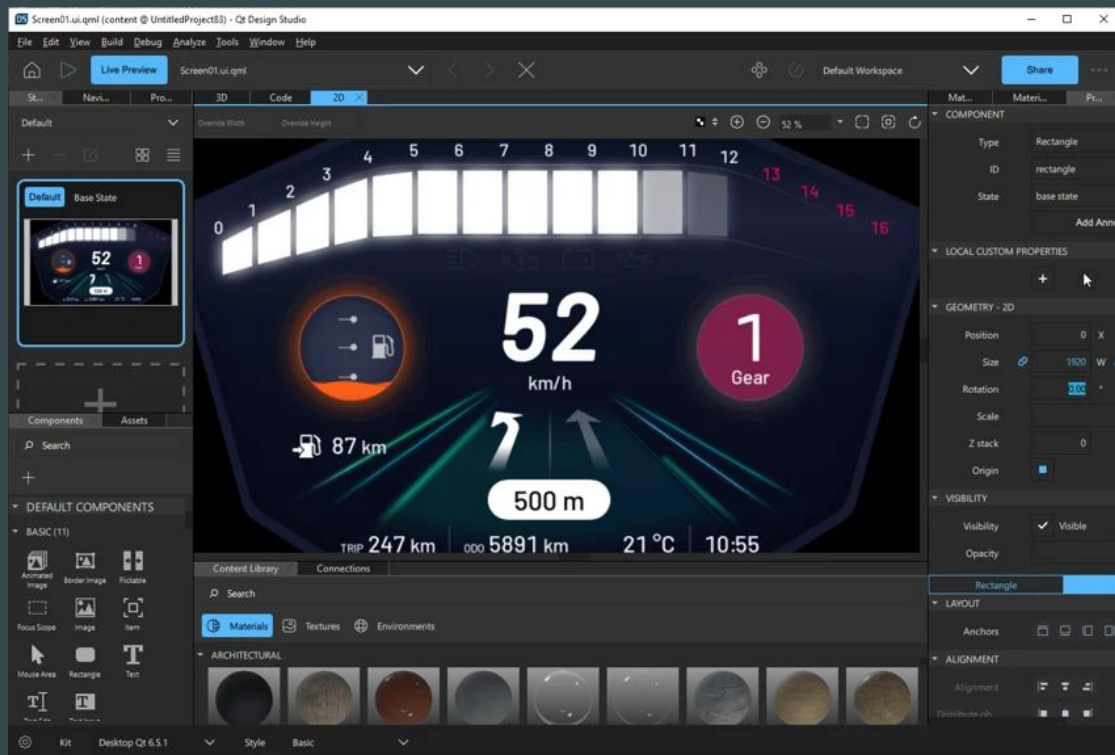
Ready

1 selected

SIZE 200 x 40

www.embedded-wizard.de

Qt for MCU



Slint UI

The image shows the Slint IDE interface. On the left is a code editor with the following content:

```
1 // Copyright © SixtyFPS GmbH <info@slint.
2 // SPDX-License-Identifier: MIT
3
4 import { DemoPalette, Page } from "common
5 import { HomePage } from "./home_page.sli
6 import { InkLevel, InkPage } from "./ink_
7 import { SettingsPage, PrinterSettings }
8 import { PrinterQueue } from "./printer_q
9
10 // re-export for the native code
11 export { PrinterQueue, PrinterSettings }
12
13 import "./fonts/NotoSans-Regular.ttf";
14 import "./fonts/NotoSans-Bold.ttf";
15
16 ▶ Show Preview
17 component SideBarIcon inherits Rectangle
18     in-out property <bool> active;
19
20     callback activate;
21
22     GridLayout {
23         padding: 0px;
24         @children
25     }
```

On the right is a preview window titled "Slint Printer Demo". It features a dark blue sidebar with icons for a printer and a scanner. The main area contains the text "Slint Printer Demo" and two large buttons: "Print" with a printer icon and "Scan" with a scanner icon. To the right of the main area, there is a "Printing Queue" section with items like "63% - PRINTING 210106-Fin" and "WAITING... Adressliste.c". Below the preview is a "Welcome" window with the text "Welcome to SlintPad" and the Slint logo.

Tip #10

Play with more languages and AI

Programming languages

Active support by Espressif teams

- C/C++
 - most common choice - <https://github.com/espressif/esp-idf>
- Rust
 - security and memory guaranties of Rust
 - <https://github.com/esp-rs>
 - multi-target support: Xtensa, RISC-V, plus WASM, desktops or mobile
- Arduino - Maker choice
 - Arduino IDE 2.x
 - note: check the license for production



esp-rs

Libraries, crates and examples for using Rust on Espressif SoC's

OSes - C/CPP, Rust



no_std a.k.a. bare metal with Rust - <https://github.com/esp-rs/esp-hal> (minimalistic)

ESP-IDF (OS based on FreeRTOS) - <https://github.com/esp-rs/esp-idf-hal>



Zephyr - <https://zephyrproject.org/>

- EDC22 Day 1 Talk 10: Applications of Asymmetric Multiprocessing with ESP32 Devices - including Rust on one core - <https://youtu.be/oble9ObAqxM>



NuttX - <https://nuttx.apache.org/> (as app, Linux-like OS)

SVD files: <https://github.com/espressif/svd>

Other languages and frameworks in context of ESP32

- Zig

VM based:

- CircuitPython and MicroPython - Python-like language
- DeviceScript - TypeScript language - Microsoft Research
- Lua
- Mongoose OS
- Nanoframework - C# language
- Toit
- downside: bigger VM
- upside: more robust, comes with OTA and monitoring

Accelerated learning with AI

Help with learning new prog. language

Very good results: Chat GPT 4, GitHub CoPilot

Less accurate results: Chat GPT 3.5, Gemini



Tip #11

Read books

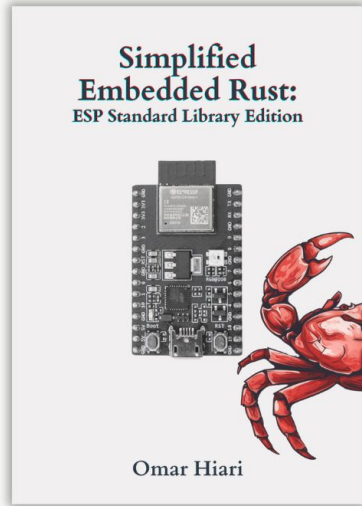
NEW BOOK RELEASES

LEARN EMBEDDED RUST WITH ESP

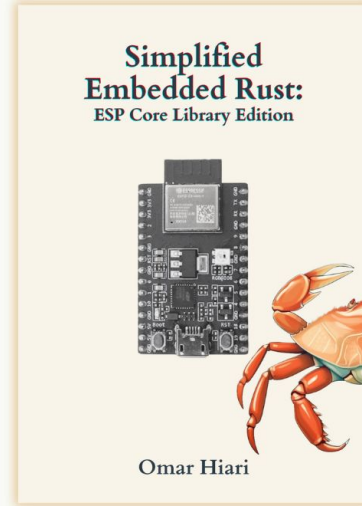
RELEASE DATE: MAY 17TH

STANDARD LIBRARY EDITION

CORE LIBRARY EDITION



[HTTPS://ME-QR.COM/L/SER-STD](https://me-qr.com/l/ser-std)



[HTTPS://ME-QR.COM/L/SER-NO-STD](https://me-qr.com/l/ser-no-std)



SPECIAL EDITION

Special edition guest-edited by

140 Pages

ESPRESSIF Prototyping With Espressif Chips

Try it with ESP Launchpad

ADF, IDF, and Other SDKs

Insights from **Espressif Engineers**

Automation With Rainmaker and Matter

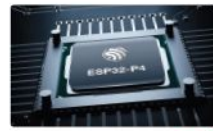
Trying Out the **ESP32-S3-BOX-3**

ESP32 and ChatGPT

In this issue

- > An Open-Source Speech Recognition Server
- > Power Duct: Rust + Embedded
- > Facial Recognition With ESP32-S3-EYE
- > Walkie-Talkie with ESP-NOW
- > Another Elektor Christmas Tree Project

and much more!



Unleashing the ESP32-P4
The Next Era of Innovative
Microcontrollers

p. 59



Acoustic Fingerprinting
Song Recognition With ESP32

p. 80



A Vision for the AIoT
Interview with Espressif CEO
Teo Swee-Ann

p. 35



9 770532 546006

Elektor Mag

Special Guest Edition

<https://www.elektor.com/products/elektor-special-esspressif-guest-edition-2023-pdf-en>

Developing IoT Projects with ESP32

<https://blog.espressif.com/book-review-developing-iot-projects-with-esp32-2nd-edition-facdef7545bb>

Developing IoT Projects with ESP32

Discover the IoT development ecosystem with ESP32 to
create production-grade smart devices

Second Edition



<packt>

Vedat Ozan Oner

MicroPython Projects

<https://www.packtpub.com/product/micropython-projects/9781789958034>

MicroPython Projects

A do-it-yourself guide for embedded developers to build a range of applications using Python



Jacob Beningo

Packt

www.packt.com

ESP32-C3 Wireless Adventures a Comprehensive Guide to IoT

<https://espressif.github.io/esp32-c3-book-en/>

ESP32-C3

Wireless Adventure:
A Comprehensive Guide to IoT

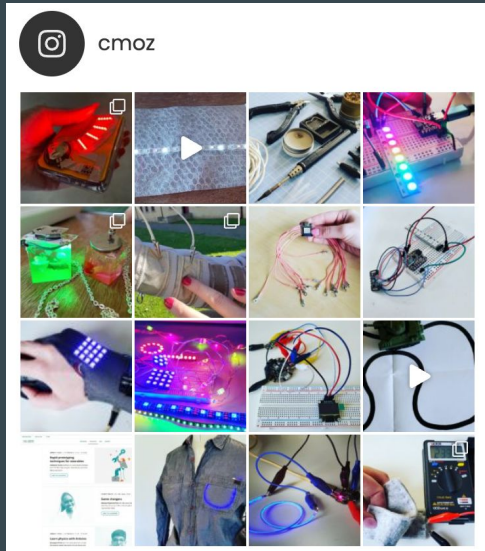
RISC-V Wi-Fi Bluetooth
ESP-IDF ESP RainMaker



Wearables

The Ultimate Guide to Informed Wearable Technology

- book: <https://packt.link/01VBv>



More books about ESP32

<https://www.espressif.com/en/ecosystem/community-engagement/books>



ESPRESSIF

Hardware

SDKs

Cloud

Solutions

Support

Ecosystem

Company

Contact



Partnership and Resource

AWS Technology Partner

Third-Party Platforms

Third-Party SDKs



Developer Zone

Espressif DevCon

Tech Blogs

ESP32 Forum



Community

Courses

Rust

Books

Videos

Projects

Tip #12

Follow us

Embedded World 2025

Meet us in Nuremberg, Germany - 11th - 13th March 2025




embeddedworld

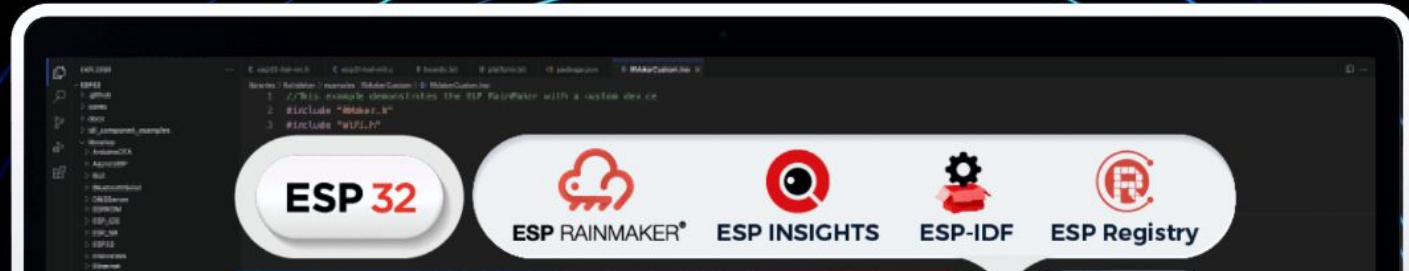
Exhibition&Conference

<https://devcon.espressif.com/>

Sep 3-5, 13:00-18:00 CEST

Espressif DevCon24

 Add to Calendar >



Visit us in Brno

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe

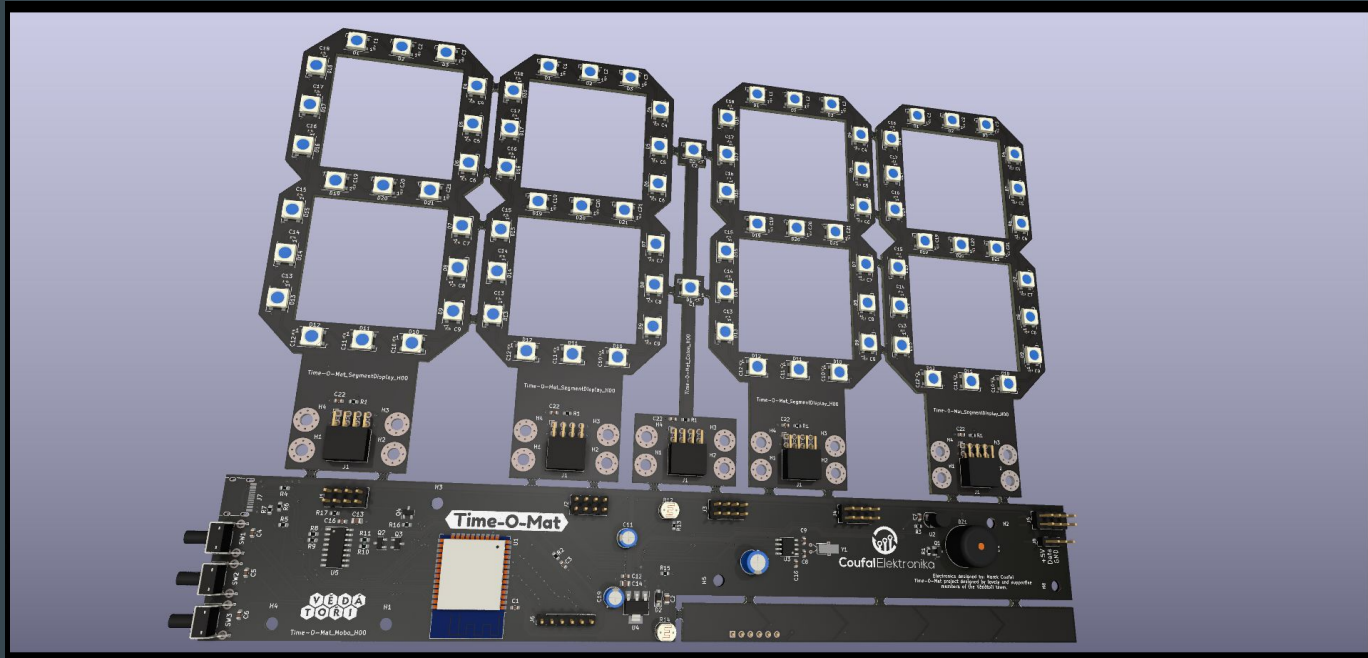


Tip #13

Create and explore

Time-O-Mat - built at summer camp

<https://github.com/vedatori/Time-O-Mat>



Grafana



<https://grafana.com/blog/2020/06/17/how-to-monitor-a-sourdough-starter-with-grafana/>

<https://github.com/grafana/diy-iot> - Arduino now. We're not Rust yet :)

Tip #14

Have fun