

Dispositivos Conectados ESP32 e a *toolchain* ESP-IDF

2025/2026

Authors: Paulo C. Bartolomeu

1 Objetivos

- Familiarização com a placa ESP32
- Instalação e familiarização com a *framework* ESP-IDF
- Criação e execução de um projeto “Hello World”

2 Kits de desenvolvimento ESP32

As aulas práticas da UC de Dispositivos Conectados suportam-se na utilização de um módulo ESP32-C6-DevKitC-1 da Espressif Systems para o desenvolvimento de soluções IoT (Internet das Coisas). Este módulo de elevado desempenho possui recursos avançados de conectividade (Wi-Fi, Bluetooth LE, Zigbee e Thread), proporcionando uma plataforma robusta para criar uma ampla gama de aplicações. O módulo ESP32-C6-DevKitC-1 é fornecido num kit base IoT constituído pelos componentes apresentados na Tabela 1:

Tabela 1: Kit Base.

Componente	Descrição
ESP32-C6-DevKitC-1	Placa de desenvolvimento
Placa branca	Placa para montagem de circuitos
DHT20	Sensor de temperatura e humidade
Cabo USB	Cabo USB-A para USB-C (1m)

Para dar suporte às aulas onde se irá trabalhar a interação com o utilizador, a meio do semestre será fornecido um kit adicional constituído pelos componentes da Tabela 2:

Tabela 2: Kit Extra.

Componente	Descrição
Display	TFT RGB 0.96" com leitor de cartões SD integrado
SD Card	Cartão SD de 16GB

2.1 ESP32-C6-DevKitC-1

O ESP32-C6-DevKitC-1 é baseada no módulo ESP32-C6-WROOM-1(U), um módulo de uso geral com 8 MB de flash SPI. A maior parte dos pinos de I/O são disponibilizados em *headers* na parte inferior da placa, facilitando a conexão de periféricos. O desenvolvimento com esta placa pode ser realizado conectando periféricos através de fios *jumper* ou montando o ESP32-C6-DevKitC-1 numa placa branca e ligando aí os periféricos. A vista isométrica da placa está representada na Figura 1.

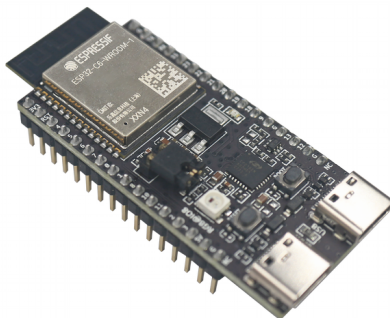


Figura 1: ESP32-C6-DevKitC-1: Vista Isométrica

O ESP32-C6-DevKitC-1 integra funções completas de Wi-Fi 6 na banda de 2,4 GHz, Bluetooth 5 e IEEE 802.15.4 (Zigbee 3.0 e Thread 1.3).

Todos os pinos *General Purpose Input/Output* (GPIO) disponíveis (exceto o barramento SPI para a *flash*) são disponibilizados nos *headers* da placa. A placa também possui um regulador de tensão *low-dropout* (LDO) de 5 V para 3,3 V, um LED de *power ON* e um conversor USB<->UART que suporta taxas de transferência até 3 Mbps.

O ESP32-C6-DevKitC-1 possui uma porta USB Type-C compatível com USB 2.0, capaz de transferências até 12 Mbps. Essa porta é usada para alimentação da placa, gravação de *firmware* e comunicação com o módulo usando protocolos USB, bem como para depuração *Joint Test Action Group* (JTAG).

A placa também possui um botão de *download* (*Boot*) e um botão de reinicialização (*Reset*), além de um LED RGB endereçável acionado pelo GPIO8. Há ainda um conector J5 usado para medição de corrente. A Figura 2 representa e identifica estes elementos na placa.

Os *headers* do ESP32-C6-DevKitC-1 permitem acesso a sinais digitais, interfaces de comunicação série tais como UART, I2C e SPI, além de pinos analógicos para leitura de sensores. São também disponibilizados pinos dedicados a funções especiais, tais como *Pulse Width Modulation* (PWM) para controlo de motores e servos, além de entradas e saídas de propósito geral (GPIO) que podem ser programadas conforme as necessidades.

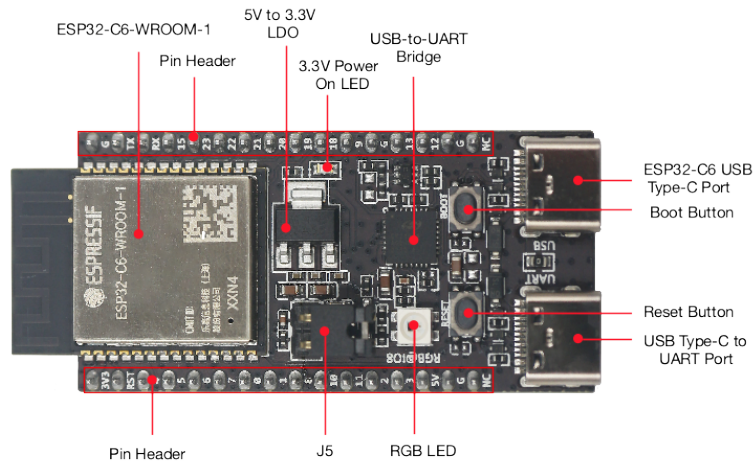


Figura 2: Elementos fundamentais do ESP32-C6-DevKitC-1

ESP32-C6-DevKitC-1

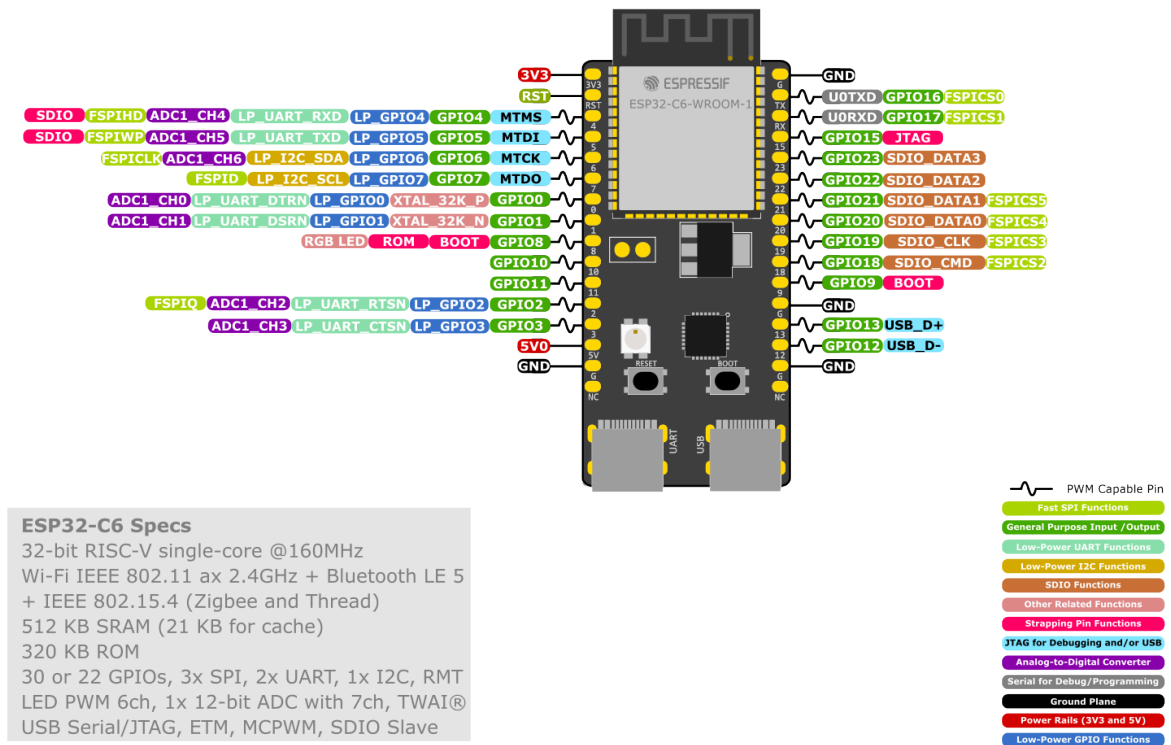


Figura 3: Especificação dos pinos do ESP32-C6-DevKitC-1

J1

No.	Name	Type ¹	Function
1	3V3	P	3.3 V power supply
2	RST	I	High: enables the chip; Low: disables the chip.
3	4	I/O/T	MTMS ³ , GPIO4, LP_GPIO4, LP_UART_RXD, ADC1_CH4, FSPiHD
4	5	I/O/T	MTDI ³ , GPIO5, LP_GPIO5, LP_UART_TXD, ADC1_CH5, FSPiWP
5	6	I/O/T	MTCK, GPIO6, LP_GPIO6, LP_I2C_SDA, ADC1_CH6, FSPiCLK
6	7	I/O/T	MTDO, GPIO7, LP_GPIO7, LP_I2C_SCL, FSPiD
7	0	I/O/T	GPIO0, XTAL_32K_P, LP_GPIO0, LP_UART_DTRN, ADC1_CH0
8	1	I/O/T	GPIO1, XTAL_32K_N, LP_GPIO1, LP_UART_DSRN, ADC1_CH1
9	8	I/O/T	GPIO8 ^{2 3}
10	10	I/O/T	GPIO10
11	11	I/O/T	GPIO11
12	2	I/O/T	GPIO2, LP_GPIO2, LP_UART_RTSN, ADC1_CH2, FSPiQ
13	3	I/O/T	GPIO3, LP_GPIO3, LP_UART_CTSN, ADC1_CH3
14	5V	P	5 V power supply
15	G	G	Ground
16	NC	-	No connection

J3

No.	Name	Type	Function
1	G	G	Ground
2	TX	I/O/T	U0TXD, GPIO16, FSPiC50
3	RX	I/O/T	U0RXD, GPIO17, FSPiC51
4	15	I/O/T	GPIO15 ³
5	23	I/O/T	GPIO23, SDIO_DATA3
6	22	I/O/T	GPIO22, SDIO_DATA2
7	21	I/O/T	GPIO21, SDIO_DATA1, FSPiC55
8	20	I/O/T	GPIO20, SDIO_DATA0, FSPiC54
9	19	I/O/T	GPIO19, SDIO_CLK, FSPiC53
10	18	I/O/T	GPIO18, SDIO_CMD, FSPiC52
11	9	I/O/T	GPIO9 ³
12	G	G	Ground
13	13	I/O/T	GPIO13, USB_D+
14	12	I/O/T	GPIO12, USB_D-
15	G	G	Ground
16	NC	-	No connection

3 Plataforma ESP-IDF

A plataforma ESP-IDF (*Espressif IoT Development Framework*) é uma opção popular para o desenvolvimento de soluções IoT baseadas no módulo ESP32-C6-DevKitC-1, fornecendo um ambiente de desenvolvimento robusto, abrangente, e alinhado com as capacidades do módulo.

O ESP-IDF é projetado especificamente para tirar proveito dos recursos do ESP32-C6, oferecendo um conjunto abrangente de *drivers*, bibliotecas e exemplos. A sua integração permite o foco do desenvolvimento no *core* das aplicações IoT, evitando as complexidades de baixo nível relacionadas com a configuração e controlo dos periféricos.

Adicionalmente, esta plataforma é *open-source* e bem documentada, contando com uma comunidade ativa de utilizadores e *developers*, o que facilita a descoberta de soluções, a resolução de problemas e a adoção de práticas comprovadas durante o ciclo de desenvolvimento.

Outra vantagem significativa do ESP-IDF é o seu suporte nativo a recursos avançados de conectividade, tais como Wi-Fi 6, Bluetooth 5 e protocolos de rede de baixa potência, como Zigbee e Thread. Essa abordagem integrada simplifica enormemente a implementação de aplicações IoT que exigem acesso a múltiplas tecnologias de comunicação.

3.1 Instalação e configuração da plataforma ESP-IDF

O primeiro passo para iniciar a jornada com o módulo ESP32-C6-DevKitC-1 é a instalação do ESP-IDF que irá permitir editar o *firmware* da aplicação IoT, compilar o código fonte, programá-lo no módulo e realizar a respetiva depuração e teste.

Para o efeito, deve seguir rigorosamente as indicações disponibilizadas na página [Get Started para o módulo ESP32-C6 da Espressif](#) onde encontra um guia passo-a-passo de como instalar a plataforma.

NOTA 1: Assegure-se que instala os pré-requisitos devidamente.

A forma recomendada de instalação nesta disciplina é através da [Extensão para o VSCode](#) e sugere-se o uso da versão ESP-IDF 5.5.1.

No Visual Studio Code previamente instalado deverá clicar no ícon de Extensões da Barra de

Actividade (ou através do comando View -> Show Extensions - Ctrl+Shift+X), procurar a extensão “ESP-IDF” e prosseguir com a respetiva instalação. As Figuras 4, 5 e 6 resumem o processo a adoptar.

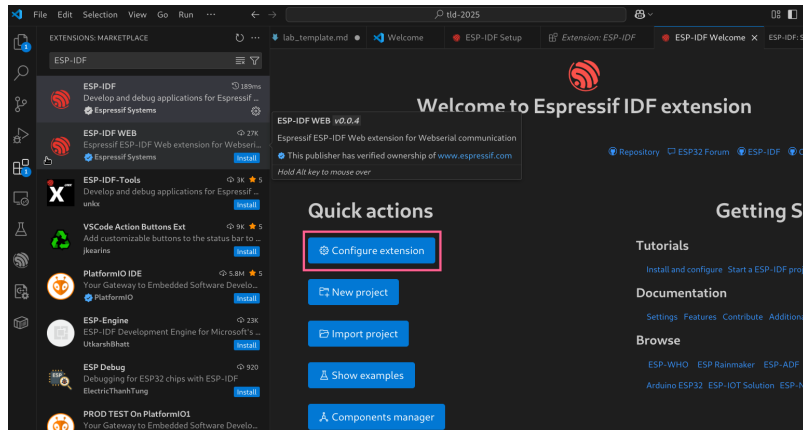


Figura 4: Configuração do ESP-IDF

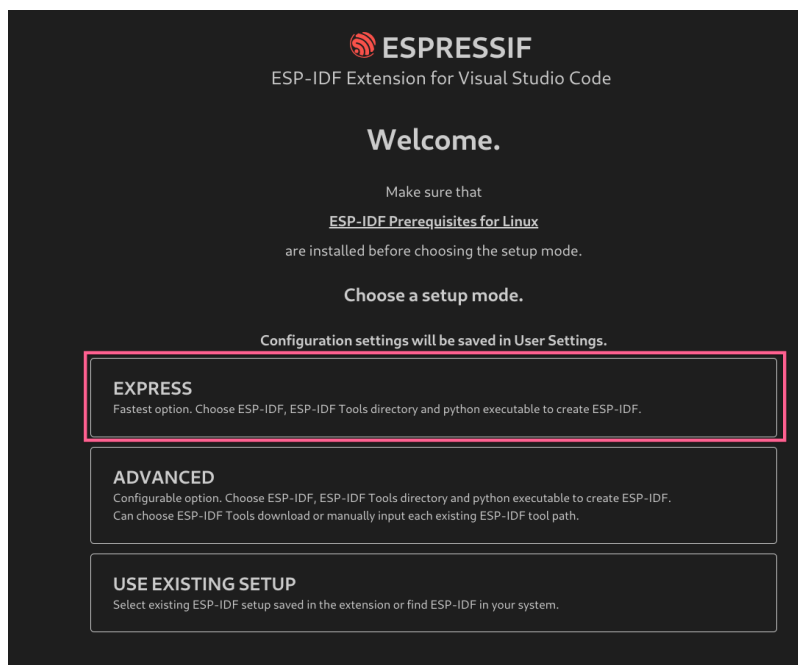


Figura 5: Configuração *Express*

Por fim, a secção do “Using the ESP-IDF Extension for VS Code” disponibiliza um pequeno tutorial sobre como usar a extensão, concretamente como interpretar os ícones da barra do fundo da janela do VS Code que são disponibilizados.

4 Aplicação “Hello World”

Depois de ter concluído a instalação do ambiente de desenvolvimento o próximo passo é a criação de uma aplicação básica para testar todo o ecossistema.

ESPRESSIF
ESP-IDF Extension for Visual Studio Code

Git version: 2.30.2

Select download server:
Github

Show all ESP-IDF tags

Select ESP-IDF version:
v5.5.1 (release version)

Enter ESP-IDF container directory:
/home/parallels/esp /v5.5.1/esp-idf

Enter ESP-IDF Tools directory (IDF_TOOLS_PATH):
/home/parallels/esp/esp-idf/tools

Select Python version:
/usr/bin/python3

Install

Figura 6: Parâmetros de configuração

Para o efeito, no menu de criação novos projetos, crie um projeto com base no exemplo “Hello World”. As Figuras 7 e 8 ilustram o processo.

NOTA 2: Caso não seleccione devidamente o *target* ESP32-C6 os binários gerados não serão compatíveis com a placa.

Após ter criado o projeto, compile-o, descarregue-o para a placa e teste-o, tentando interpretar o código e o respetivo funcionamento.

5 Acknowledgements

Originally authored by [Paulo C. Bartolomeu](#)

New Project

Project Name
hello

Enter Project directory
/home/parallels /hello

Choose ESP-IDF Target (IDF_TARGET)
esp32c6

Choose ESP-IDF Board
ESP32-C6 chip (via builtin USB-JTAG)

Choose serial port
/dev/ttyS0

Add your ESP-IDF Component directory
/home/parallels/esp/v5.5.1/esp-idf/components

[Choose Template](#)

Figura 7: Configuração base da aplicação

New Project

ESP-IDF [Create project using template hello_world](#)

Search Template By Name

Supported Targets	ESP32	ESP32-C2	ESP32-C3	ESP32-C5	ESP32-C6	ESP32-C61	ESP32-H2	ESP32-H21
get-started								
blink								
hello_world								
bluetooth								
bluedroid								
Bluedroid_Beacon								
Bluedroid_Connection								
Bluedroid_GATT_Server								
nimble								
NimBLE_Beacon								
NimBLE_Connection								
NimBLE_GATT_Server								
NimBLE_Security								
ble								
ble_throughput								
throughput_client								
throughput_server								
ble_ancs								

Hello World Example

Starts a FreeRTOS task to print "Hello World".

(See the README.md file in the upper level 'examples' directory for more information about examples.)

How to use example

Follow detailed instructions provided specifically for this example.

Select the instructions depending on Espressif chip installed on your development board:

- [ESP32 Getting Started Guide](#)
- [ESP32-S2 Getting Started Guide](#)

Example folder contents

The project **hello_world** contains one source file in C language **hello_world_main.c**. The file is located in folder **main**.

ESP-IDF projects are built using CMake. The project build configuration is contained in **CMakeLists.txt** files that provide set of directives and instructions describing the project's source files and targets (executable, library, or both).

Figura 8: Criação da aplicação a partir do template “Hello World”